

A Dissertation

entitled

The impact of learning analytics and badges in providing immediate detailed
feedback through dashboard on students' performance

by

Anil Varma Penumatsa

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the
Masters of Science Degree in Engineering

Dr. Vijayakumar Devabhaktuni, Committee Chair

Dr. Jason Stroud, Committee Co-Chair

Dr. Mansoor Alam, Committee Member

Dr. Ahmad Y. Javaid, Committee Member

Dr. Cyndee Gruden, Dean
College of Graduate Studies

The University of Toledo

December 2018

Copyright 2018, Anil Varma Penumatsa

This document is copyrighted material. Under copyright law, no parts of this document may be reproduced without the expressed permission of the author.

An Abstract of

The impact of learning analytics and badges in providing immediate detailed
feedback through dashboard on students' performance

by

Anil Varma Penumatsa

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the
Masters of Science Degree in Engineering

The University of Toledo
December 2018

The topic of instructional feedback to residents is an under-researched area in medical education. As the demand for feedback in educational information on the progress of learners continues to grow, the need for delivering effective feedback quickly and accurately becomes of great importance. Over the years, the paper-based techniques have been used for feedback. This method delivers feedback in indefinite time. Meanwhile, the residents undergo further evaluations and they often provide an incomplete picture of the students' whole performance. Therefore, paper evaluations are frustrating and place a heavy burden on instructors to track students' progress within the training program. Hence, the web-based instant feedback techniques play a prominent role to deliver the progress to the students.

This thesis proposes a two-tier cloud-based web model-view-controller (MVC) architecture as a proof-of-concept, which allows the instructors to submit the feedback and the students to check their progress instantaneously in the web browser. This is accomplished by developing an MVC architecture with a web-based called "Centralized Anesthesia Resident Achievement Tracking System (CARATS)". It is an online platform for medical faculty members to submit the feedback on multiple areas of resident's performance including daily performance, oral presentations, and simulation sessions. Additionally, this system compiles case numbers as logged by the residents

into the ACGME system. All data entered into the system are displayed on a dashboard in a pertinent graphical format for both the residents and instructors.

The web-based system offers several advantages over traditional feedback mechanisms. First, it allows the instructors to easily submit daily feedback on the resident's performance and makes such data immediately available to the students via the dashboard so that corrective measures can be taken immediately, rather than a month later. Second, by aggregating and integrating performance in multiple areas of residency in one place and providing an insightful overall score, it allows the students' and instructors to more easily track the progress throughout the training.

Acknowledgments

Firstly, I would like to thank my advisor Dr. Vijayakumar Devabhaktuni, for all of his support, patience, motivation, and appreciation over the course of my graduate studies. I would also like to thank my co-advisor Dr. Jason Stroud for his technical advice, continuous supervision, participation, and inputs. I would further like to thank Dr. Ahmad Y. Javaid and Dr. Mansoor Alam for agreeing to serve as members of my thesis committee. Secondly, I would like to thank The University of Toledo's Department of Electrical Engineering and Computer Science for providing me with financial support in the forms of graduate assistantships and tuition waivers. I would also like to express my appreciation for the University of Toledo Medical Campus for the research portion. In addition, I would like to thank my colleague Osama Hussein for the continuous support during thesis writing. My current and former colleagues of labs 2042 are also greatly appreciated for their guidance and support.

Contents

Abstract	iii
Acknowledgments	v
Contents	vi
List of Tables	ix
List of Figures	x
List of Abbreviations	xiii
1 Introduction	1
1.1 Definition of feedback in clinical education	2
1.2 Feedback Components	2
1.3 Core competencies in medical education	3
1.4 Flaws in current feedback mechanism	5
1.5 Statement of the problem	5
1.6 Proposed method	6
1.7 Thesis Organization	7
2 Literature Review	8
2.1 Usage of web tools	8
2.2 Usage of Analytics	11
2.2.1 Process of analytic applications	12

2.2.2	Dashboard	12
2.3	Usage of Badges	14
2.4	Conclusion	16
3	Architecture Overview	17
3.1	Objectives	18
3.2	Proposed Architecture	18
3.3	Decomposition Overview	20
3.3.1	Client Overview	20
3.3.2	Services Overview	21
3.3.3	Database Overview	22
3.4	Technology Stack	22
3.4.1	AngularJS	23
3.4.2	Spring	24
3.4.3	Microsoft SQL Server	26
3.4.4	Hibernate	28
3.4.5	Bootstrap	29
3.5	Data flow in CARATS Architecture	30
3.6	Client Architecture	32
3.7	Server Architecture	38
3.8	Database	57
3.9	Cloud Deployment	68
4	Centralized Anesthesia Resident Achievement Tracking System	74
4.1	Application Overview	75
4.1.1	User Roles and Security	78
4.1.2	Password protection	82
4.1.3	User profile	84

4.2	Inputs	85
4.2.1	Simulation Evaluation	87
4.2.2	Grand Rounds	90
4.2.3	Case Log	91
4.2.4	Daily Feedback	93
4.3	Outputs	95
4.3.1	Admin pages	95
4.3.1.1	Summary page details	95
4.3.2	Residents' dashboard	98
4.3.2.1	Graphically interfaced individual feedback	100
4.3.2.2	Badges	112
4.3.2.3	Residency Score	115
5	Results & Discussions	117
5.1	Daily Feedback	117
5.2	Dashboard	118
5.3	Residency Score	119
5.4	Badges	120
5.5	Overall performance of CARATS	122
6	Conclusion Remarks & Future work	124
6.1	Future work	125
	References	126

List of Tables

3.1	The screen sizes implemented in the application	37
3.2	Error code series and their information	47
3.3	The error codes & their information	48
3.4	Syntax to clear session objects used in the application	53
3.5	Syntax used to implement transactions in the application	54
4.1	Bar graph value representation of simulation evaluation	104
4.2	Bar graph values representation of grand rounds	107
4.3	ACGME standards of Case logs	111
4.4	Weight-age for resident score.	115

List of Figures

2-1	OFES feedback web form	9
2-2	The online feedback system developed by Rahmann and Amit	11
2-3	Dashboard developed by Linda Corring and Paula De Barba	14
3-1	CARATS proposed architecture	19
3-2	Data flow in CARATS	31
3-3	Flow of data in client archietecture	33
3-4	The responsive design of CARATS works on these devices	34
3-5	Flow of data in server archietecture	38
3-6	DAO hibernate archietecture	49
3-7	DAO hibernate Transaction session states	51
3-8	DAO hibernate Transaction	53
3-9	Application query flow	55
3-10	Normalization flow	63
3-11	Database flow	64
3-12	Cloud Component layout	69
3-13	AWS archietecuture flow	70
4-1	CARATS Home page	76
4-2	CARATS collaborators expansion pop up	77
4-3	CARATS Login page	77
4-4	User administration page	80

4-5	Roles structure	81
4-6	Access denied page	81
4-7	Navigation bar for different users	82
4-8	Password change page	83
4-9	User account details for admin	84
4-10	User profile details	86
4-11	Add simulation type form	88
4-12	Simulation evaluation form	89
4-13	Grand rounds form	92
4-14	Case log file upload	93
4-15	Daily feedback form	94
4-16	Sample results page navigation link	95
4-17	Sample column visibility popup	97
4-18	Presentaion and simulation summary tables	99
4-19	Daily feedback and case log tables	99
4-20	Resident's dashboard	100
4-21	Spider graph representation of simulation evaluations	102
4-22	Listed representation of simulation evaluations	103
4-23	Bar graph representation of simulation evaluations	103
4-24	Spider graph representation of grand rounds	106
4-25	Listed representation of presentation evaluations	106
4-26	Bar graph representation of grand rounds	107
4-27	Checklist representation of case logs	109
4-28	Bar graph representation of case log	110
4-29	Stock graph representation of daily feedback	113
4-30	Resident badges	114
4-31	Residency score	116

5-1	Bar graph representation of residents' rating on daily feedback	118
5-2	Bar graph representation of residents' rating on dashboard	119
5-3	Bar graph representation of residents' rating on residency score	120
5-4	Bar graph representation of residents' rating on badge	121
5-5	Bar graph representation of residents' rating on overall performance of CARATS	123

List of Abbreviations

3-D	Three Dimensional
ACGME	Accreditation Council For Graduate Medical Education
AMI	Amazon Machine Image
API	Application Programming Interface
AUTH	Authentication
B-Rep	Boundary Representation
CAD	Computer-Aided Design
CARATS	Centralized Anesthesia Residents Achievement Tracking System'
CASE	Comprehensive Anesthesia Simulation Environment
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DAO	Data Access Object
DDL	Data Definition Language
DML	Data Manipulation Language
DNS	Domain Name System
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
ER	Entity Relationship
FEA	Finite Element Analysis

FTP File Transfer Protocol

GIF Graphics Interchange Format

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

JDBC Java Database Connectivity

JS JavaScript

JSON JavaScript Object Notation

LHS Left-hand Side

MEMS Micro-Electro-Mechanical Systems

MPS Multi Pilot Simulator

MVC Model View Controller

NB Number of Badges

NCL Number of Case Log

OFES Online Feedback System

POJO Plain Old Java Objects

REST REpresentation State Transfer

RHS Right-hand Side

SDF Score of Daily Feedback

SDK Software Development Kit

SGR Score of Grand Rounds

SMTP Simple Mail Transfer Protocol

SPA	Single Page Application
SSE	Score of Simulation Evaluation
STL	Stereolithography file format
WAR	Web application ARchive
WebGL	Web Graphics Library
XML	Extensible Markup Language

Chapter 1

Introduction

The topic of instructional feedback to residents is an under-researched area in medical education. Feedback in general education is defined as information on the progress of learners provided through various channels [1]. As the demand for feedback in education continues to grow, the need for delivering effective feedback quickly and accurately becomes of great importance. Using an effective feedback mechanism is crucial in clinical learning situations because it motivates residents to correct their mistakes. Many educational institutions provide feedback via paper evaluations which reach to residents after a couple of weeks. Meanwhile, the resident undergoes further evaluations before analyzing their current performance . Moreover, the paper evaluations often provide an incomplete picture of student's whole performance. Paper evaluations are frustrating and place a heavy burden on instructors to track resident's progress within the training program. The greatest challenges in the area of medical education are (1) delivering the feedback to residents with consistency, in a timely manner, and (2) tracking of residents' medical evaluations progress to the instructors. Overcoming these challenges can potentially increase residents' motivation and improves prediction of residents' success rate.

1.1 Definition of feedback in clinical education

In clinical education, feedback is acknowledged as an essential tool in learning. Judging from the number of publications and their correlated topics, feedback is crucial [2 - 5]. For example, Hattie and Timberley stated in their review of 196 studies on feedback in the classroom that feedback is as powerful as quality and quantity of instruction to the residents [6]. Moreno described effective feedback as important to improve knowledge and skill acquisition [7]. The concept of feedback is used in various fields and its definition is different in each field. The definition of feedback in electronics is “the return of a partial fraction of output of a machine or system as an input to the source to adjust” [8]. The social definition of feedback is “the control of the behavior of an object by an error at a given time with the specific goal,” whereas in clinical education, it is the information shared with the resident concerning the correctness or accuracy [9]. In short, the definition of the feedback for purpose of this study is a scorecard of resident’s performance.

1.2 Feedback Components

Feedback is instrumental in the learning experience of students and is often given as a part of the assessment. These assessments can enable learners to consolidate their strengths and identify their weaknesses. Their strengths and weaknesses direct them to the necessary actions to achieve the learning outcomes. Feedback is an essential component in all learning contexts [10]. It serves a variety of purposes including evaluation of students’ achievements, development of students’ competencies, and understanding and elevation of students’ motivation and confidence. It can be perceived as information conveyed to the students to assess themselves. In order to be effective, feedback on assessment needs to possess a number of qualities: it needs to be timely, constructive, motivational, personal, manageable, and directly related to assessment

criteria and learning outcomes [11]. A feedback strategy should address as many of these attributes as possible to promote learning. The provision of effective and high-quality feedback has been identified as a key element of quality teaching, and this view is well supported by meta-analytic studies [12]. The ideal criteria for effective feedback in this thesis include goal-referenced, user-friendly, timely or ongoing and consistent. These are defined as follows.

Goal-referenced : Setting objectives is the key to learning. The residents should receive performance review on their objectives. The feedback mechanism is crucial to remind the residents and lead them to achieve their objectives.

User-friendly : User-friendliness helps the resident in understanding the results delivered during their curriculum. Misunderstanding the feedback will lead to residents confusion and thus mistaken actions.

Timely or Ongoing : The sooner the users get feedback, the better they learn. Timely feedback helps residents avoid committing mistakes. The evaluations are often coming after weeks hence the residents are not receiving the information to rectify their mistakes in time.

Consistent : To be useful, the feedback must be consistent all the time. The residents can assess themselves only by stable and accurate information.

1.3 Core competencies in medical education

Educators in clinical education are tasked with providing accurate, constructive, and timely feedback on resident's performance. Previously, the only available feedback tool was objective data based on the in-training examination, a once-yearly standardized exam. Effective feedback to residents is important to acquire new knowledge and skills, especially in light of the shift toward competency-based education[13]. In the United States of America, resident assessment follows the developed model of Ac-

creditation Council For Graduate Medical Education (ACGME). In July 2002, they mandated that six core competencies be incorporated into residency training. The ACGME model is developed to measure the resident's performance during the training program.

The six core competencies are as follows

1. Practice-based learning and improvement: Practice-based learning and improvement involve investigation and evaluation of their own patient care.
2. Patient care: Patient care that is compassionate, appropriate and effective.
3. Systems-based practice: Systems-based practice, exhibits an awareness of, and responsiveness to, a larger context and system of health care.
4. Medical knowledge: Medical knowledge about established and evolving biomedical, clinical and cognate sciences.
5. Interpersonal and communication skills: Interpersonal communication skills that result in effective information exchange.
6. Professionalism: Professionalism, as manifested through a commitment to carrying out professional responsibilities. [13, 14]

The resident's performance is evaluated on these six core competencies. The evaluations might be derived from simulation exercise or grand rounds presentation. There are then presented to the residents as aggregated data, many times. Based on the evaluations from the attending physicians or instructors, the committee decides the resident qualification to graduate.

1.4 Flaws in current feedback mechanism

Today, as we become more data-driven, the need for timely, meaningful, and consistent feedback has become more critical. The current feedback mechanism that most of the universities following are paper evaluations. Providing such feedback is often difficult due to time constraints. Furthermore, this feedback is often generalized and comes later in the training cycle, when there is little time to fix deficiencies. However paper evaluations are not able to deliver actionable, timely, user-friendly and consistent feedback. This paper evaluation mechanism has many flaws:

User-friendly : The resident should observe all evaluations to assess themselves on performance. Time consumed for this activity through paper evaluations is high. Therefore, the residents sometimes are inadvisedly interpolating the data. So, the evaluation results must be represented in an understandable way to the resident.

Timely or Ongoing : Paper evaluations require a large number of steps that need human intervention. Additionally the results become available at a later point of time. The paper evaluations are gathered together and stored at the secure location. Depending on the availability of the resource, these evaluations are distributed. The time to reach feedback often takes weeks or months. By that time residents' have undergone further evaluations, which can decrease or degrade their performance.

Consistent : The availability of the evaluations through paper restricts the time factor. The instructors are either provided short time or excess time, which leads to inconsistent data. Finally, there is no system or administrator to make sure the evaluations are performed only once by every individual.

1.5 Statement of the problem

Motivating and predicting the resident's success in clinical education has long been a concern for instructors. To achieve this, many of the publications and research stud-

ies have stated that effective feedback plays an important role and has the potential to improve learning and performance [15 - 17]. Medical residents believe that they are not provided with feedback regularly from the instructors [15]. To illustrate this, Sender-Liberman et al. found that, although 90% of attending surgeons reported that they gave feedback successfully, only 17% of their residents agreed with this assertion [16]. The feedback usually constitutes one-way transmission of information from the instructor to the resident. Feedback is not just telling, it is a process whereby residents obtain information and work to eliminate the differences between standards and the actual work [17]. The dominant message from the study is that the delivery medium of the feedback is problematic, both for the residents and instructors. In the clinical education, residents are required to evaluate or review themselves every session. The addition of web tools to deliver the feedback has shown decent improvements. Further research needs to be conducted to increase motivation and to predict the resident's success by implementing an effective feedback mechanism.

1.6 Proposed method

Instructors in the world of clinical education are critically in need of new tools and strategies to eliminate the risk of students' academic failure and to increase their motivational levels. Motivation is a psychological process that influences the form, direction, intensity, and duration of work-related behavior [18]. There is no significant model in market to increase motivation, but the current market is following the digital badges system to some extent. For risk of resident's academic failure, many theories have stated that use of analytics can increase resident's decision-making skills [19]. The decision-making skills help resident to concentrate more on performance. Analytic techniques utilize large data sets to provide residents with actionable information. It helps them to determine the best action to improve their performance.

By utilizing the analytical techniques, early warning systems can be constructed to predict students at-risk of academic failure [20]. Based on the research, Wang and his colleagues proposed in 2002 that learning management systems such as Blackboard or Desire2Learn systems may be able to provide an early indicator of medical residents performance [21]. More recently, John Campbell and his colleagues argued that analytic can offer new insights and identify at-risk or course failure [22]. The outcomes of this research could be fundamental in creating an effective feedback mechanism by using analytical techniques and implementing the concept of badges.

1.7 Thesis Organization

This thesis unfolds as follows:

Chapter 2 provides a literature review that presents usage of web tools, analytics and badges in feedback.

Chapter 3 presents a overall view of the proposed model.

Chapter 4 draws conclusive remarks on the results. It also discusses future research directions in which the research can be extended.

Moreover, the thesis includes an appendix that contains java source code along with respective comments.

Chapter 2

Literature Review

The purpose of this literature review is to detail usage of web tools, analytics, and implementation of badges in feedback mechanisms.

2.1 Usage of web tools

The world has been witnessing a rapid increase in the use of technology in every field. Because of their ease of use, web tools in particular offer the opportunity to share powerful information and easily collaborate [23]. If they are effectively deployed, they can offer a way to enhance residents' performance in clinical activities. Also, they provide a new dimension to deliver medical evaluation results to the residents. Graham Walton found in a research that students' highly ranked the usage of mobile technologies as resources for their studies [24]. At the time of review, mobile technologies were laptops and mobile phones. Residents are now using mobile technologies more than ever. The notion of “anytime, anyplace” recently became cheaper by the mobile and personal technologies [23].

There are several studies conducted on web tools inclusion in the education system Hatziapostolou and Paraskakis [25] introduced a web-based tool called the Online FEedback System (OFES) to conduct a study on communicating feedback and involving students in the feedback process. They included asynchronous discussions,

quizzes, announcements, documents and lecture notes in the OFES tool. They made it accessible to a class of 46 students in 2005 and a class of 34 students in 2006. Their observation from the database records stated that 100% students accessed the tool and most of the students re-visited it before the exams. The students considered OFES tool as a valuable resource before exams and it is warmly welcomed. While the OFES tool is considered successful in drawing the attention of students, it is failed in effectively portraying the student's scores through the coursework.

PRACTICAL 1 FEEDBACK FORM SETUP

INSTRUCTOR INFORMATION:
 Name: Email:

ASSIGNMENT DETAILS:
 Title: Number: Session: FALL Year:

ASSESSMENT CRITERIA:

1.	<input type="text"/>	Weight: <input type="text"/>
2.	<input type="text"/>	Weight: <input type="text"/>
3.	<input type="text"/>	Weight: <input type="text"/>

☐ Enable individual weight of criteria

OPTIONAL PARAMETERS:

<input type="checkbox"/> Enable automatic late submission penalties	<input type="checkbox"/> Allow students to view class performance statistics
<input type="checkbox"/> Required to submit to Plagiarism Detection System	<input type="checkbox"/> Enable motivational images (view examples)

Figure 2-1: OFES feedback web form

Another study by Sana et al. [26] strengthened the findings of Hatzipostolou and Paraskakis research that students' acceptance of web tools is successful. They developed a tool for the automated generation of feedback. They categorized the users into admin, student, and faculty depending upon their roles and accordingly

granted access to them. Each categorized account was provided access with certain limitations. The faculty account were given access to view comments or complaints contributed by students; students' account were given access to submit comments on feedback forms or to contact faculty/admin; and the admin account was provided access to register/edit/update a student account, to register faculty account and their display, and to generate feedback report. They observed that students more willingly participated in using the tool and it increased the students' performance. On the other hand, they also noticed some disadvantages. There were problems with computerized timing, flaws in security and limitations in internet access.

In both studies, web tools played a key role especially in promoting interaction, delivering education and providing communication between instructor and student.



Online Feedback System

SELF BETTER™

[Home](#)
[About Us](#)
[Contact us](#)
[Admin Login](#)

Student Feedback Form

Class: CSE
FINAL
DEPARTMENT: CSE
SEC:A

Sr no.	Evaluation Parameter	Grading	Always (5)	Usually (4)	Very Often (3)	Sometimes (2)	Never (1)
1.Planning & Organization							
			ES	AI	SE	NS	P&S
1	Does the Teacher comes on time?		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
2	Teacher is well planned?		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
3	Are aims and objective made clear?		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
4	Subjects Matter organized in a logical manner?		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
5	Teacher comes well prepared?		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
2.Teaching - Learning Progress							
1	Teacher Speaks Clearly and Audibly		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
2	Teacher writes/Draws Clearly		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
3	Teacher Provides Real Time Example		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
4	Teacher Provides Ability To Bring Conceptual Clarity And Promotion of Thinking Ability		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
5	Teacher uses Audio Visuals Aids To Teach		<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>

Figure 2-2: The online feedback system developed by Rahmann and Amit

2.2 Usage of Analytics

Learning is the product of interaction in several stages through the coursework [27]. Although the use of web tools promotes the interaction, efficiency in delivering the education, and providing communication between instructor and user, there is a concern about the feedback portrayal to residents or users. Most researchers believe the use of analytics to deliver feedback creates a greater impact on residents' learning

of clinical activities. Also, they asserted that analytic research is a driver to increase effectiveness and efficiency of learners and teachers [28]. Personal analytic applications help residents to improve self-knowledge by providing tools for the review and analysis of their history [29].

2.2.1 Process of analytic applications

Personal analytics are carried out in four stages, namely, awareness, reflection, sense-making, and impact [30]. Each stage has its own importance. Awareness deals with data; usually represented as activity streams, tabular overviews, or in other forms of visual representations. The reflection stage focuses on raising questions and analyzes relation to the user. The sense-making stage is associated with users response to raised questions and in the creation of new insights. The final stage, impact, is to aggregate information about the quality of data delivery and effectiveness to the user. After the impact stage, the user's response is transferred again to the first stage, awareness, and the process is resumed. The impact stage determines the effectiveness of the data presented to the user and predicts user's progress towards the goal.

2.2.2 Dashboard

In recent years, dashboards have been developed to support analytic applications. These are data visualization tools that display the status and key performance metrics to the user. They consolidate and arrange the numbers, graphical metrics, and scorecards on a single template. They are considered as one of the best resource to the resident for the review on performance and the department to compare between the residents. Dashboards help in course activity (awareness), teaching practice (reflection), following residents at risk (sense-making) and finally motivating the resident (impact). Interactive dashboards facilitate decision-making capabilities in resident's

performance. The decision dashboard is the display of the needed information to achieve one or more objectives. The information is consolidated and arranged on a single screen so it can be monitored at a glance. For example, Dr. Dolan developed a computerized interactive clinical decision dashboard [31]. Their results supported the idea that the decision dashboard format can be used to serve in critical decision-making situations. The majority of their participants considered the dashboard effective and accurate.

In another study, Corring and Barba asserted the use of a dashboard application in delivering feedback to students [32]. They developed students' dashboard to present scores. They observed the actions of the students after receiving the feedback for its influence on performance during coursework. They recruited undergraduate students and presented the designed dashboard by relating the entire class scores. They requested the recruited participants to complete the survey at the start of the semester. Their team conducted it to establish motivations relating to the subject as well as their personal learning goals. They again interviewed participants during the seventh week. After the interaction, they observed the following advantages including the ease of understanding the scores; the student's ability to plan new strategies after the interpretation of the data; and the benefit of presenting all assessment and learning activities in the consolidated view.

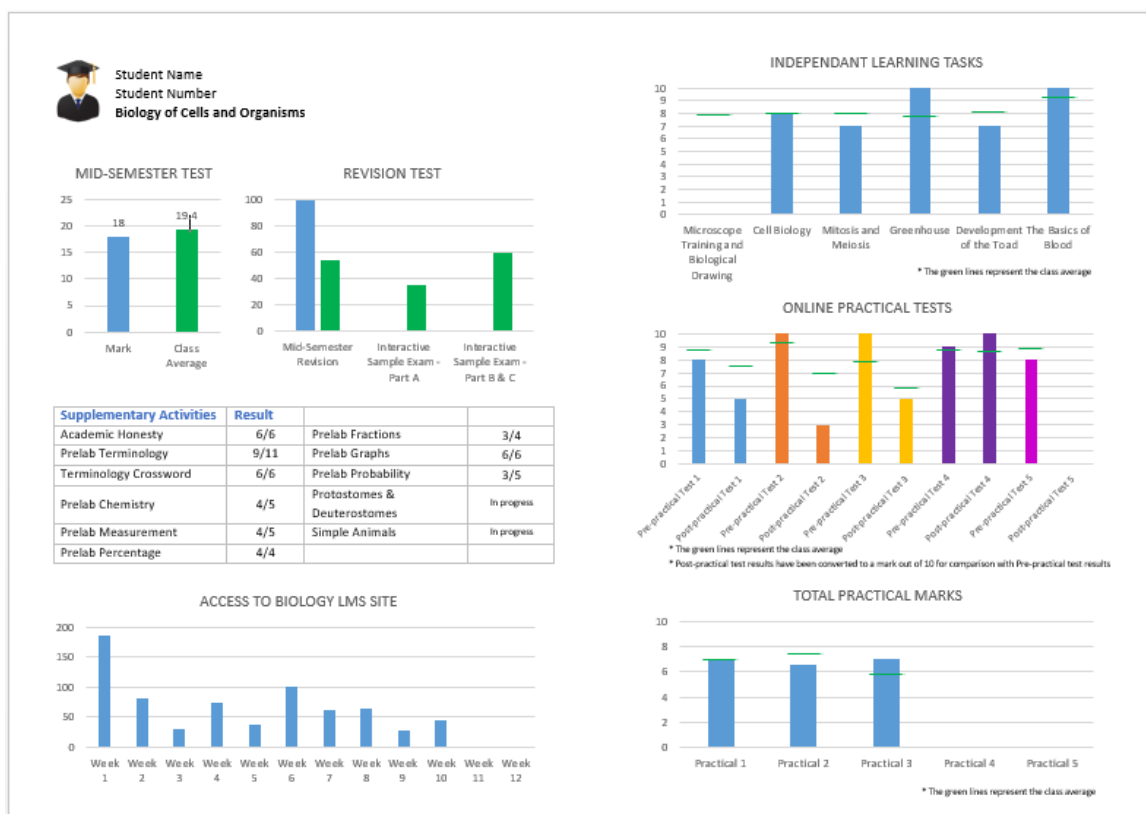


Figure 2-3: Dashboard developed by Linda Corring and Paula De Barba

2.3 Usage of Badges

The integration of dashboards with the web tools has increased the ease of understanding scores and decision making and has the benefit of presenting results in a consolidated view. However, there is a need to work on the motivational aspects which is one of the important factors in the effective feedback mechanism. Researchers hypothesized that high motivation and the accurate feedback could lead to a higher success rate [33]. The Stephen and Sean conducted a survey and observed that highly motivated observers performed less accurately, but more confidently than those in the low motivation condition [33]. To increase the motivation, different organizations are

introducing digital badges. Badge advocates claim that badges can be offered as an alternative assessment to increase the learner motivation while maintaining high-quality feedback [34]. Some researchers stated that digital badges are going to be an exceptional innovation in the education system [35]. The institutions issue digital badges to reflect the learners' achievements or the milestones along educational pathway. This concept succeeded in video gaming and has been gaining attraction in education in recent years. Digital badges can provide evidence of skills and achievements in an effective manner better than the conservative or the traditional grades and degrees. This current form of distributing badges for achievements can become the biggest asset in the education system for the current generation that has been growing up with technology. One example of the use of badges to increase motivation is seen in certain digital badges have the potential to become a way to summarize a residents' resume in job searches.

The American Museum of Natural History conducted a survey on distributing digital badges on web tools [34]. Their particular interest was the impact of badges on student's and staff's learning capabilities. They designed 38 unique digital badges; few of them are shared across program while the others are mostly mutually exclusive. They observed from the surveys that the educators reported a mild to strong interest in using digital badges, whereas at the initial stage they displayed a very little interest. Most of the students believed the badges were more useful for the long-term courses. Other than the students' interest in badges, they discovered students enjoyed the program more than before; it increased motivation, corrected students towards learning trajectory, and encouraged students to stay in focus on their objectives. Most of the students felt pride after receiving the badges for their work. Some students felt that badges have the potential to serve as their special achievements or proof of effective learning in the education program during the job interviews. They also believed the inclusion of digital badges increased the motivation towards subject.

2.4 Conclusion

From all of these studies, the researchers' conclusions and their proposed models for student's feedback is explored in various sections. To the best of my knowledge, there is no available model or application for residents' medical evaluations. To increase residents' motivation, an effective feedback mechanism that portrays evaluations and has the ability to deliver constructive and manageable feedback in real time is needed. Moreover, the application should predict the residents' outcomes during the program.

Chapter 3

Architecture Overview

The results of traditional evaluations are presented to the residents in a piecemeal format, lacking any coherence or sense of connectivity. In an effort to improve both the elicitation of data and delivery of feedback, we designed and developed a web-based computer interface system for tracking multiple facets of resident education and achievement known as Centralized Anesthesia Resident Achievement Tracking System (CARATS). CARATS is an online platform for medical faculty members to submit feedback on multiple areas of a resident's performance, including daily performance, oral presentations, and simulation sessions. Additionally, the system compiles case numbers as logged by residents into the ACGME system. All data entered into the system are displayed on a dashboard in a pertinent graphical format for both the residents and teachers. The data is used in two novel ways. First, the data used to compute a residency score, akin to credit scores, meant to provide an overall picture of how the resident is performing and progressing toward graduation. Second, the data used to provide the basis for achievements, for which the residents earn badges or digital tokens like we see in fitness contests, video gaming, and other areas. The proposed web-based system offers several advantages over traditional feedback mechanisms. First, it allows teachers to easily submit daily feedback on residents' performance and makes such data immediately available to the residents

via a dashboard so that corrective measures can be taken immediately. Second, by aggregating and integrating performance in multiple areas of residency in one place and providing an insightful overall score, it allows residents and teachers to easily track progress throughout training. Third, through gamification, it motivates medical residents toward achievements. Finally, CARATS is dynamic and generalized, and allows development of additional feedback forms, and can be tailored to the specific needs of a residency program.

3.1 Objectives

We started the development of CARATS by keeping the following five objectives in mind: 1) to allow faculty to easily submit daily feedback, 2) to make data immediately available to residents via dashboard, 3) to aggregate and integrate performance in multiple areas of residency in one place and provide an overall score, 4) to motivate residents toward achievements through gamification, and 5) to create a dynamic system that allows development of additional feedback forms and can be tailored to the specific needs of a residency program.

3.2 Proposed Architecture

This section describes the architecture of CARATS. The CARATS website follows a two-tier application architectural style that involves client and server applications. This architecture is also known as client server application. In this architecture, the presentation and business logic runs on client, and server handles the database. The two-tier architecture offers high performance because of physical closeness of client and server. Implementation of two-tier architecture brings the following advantages: [36]

1. Less components to integrate and interface help to avoid loading libraries,
2. A single provider can offer majority of integration guarantees,
3. The components/ modules have little overlap,
4. Less communication overhead between the applications,
5. Smaller, faster applications because of their physical closeness,
6. Less development environments to master the application.

CARATS follows model-view-controller architecture. In translation, this architecture provides a client application that is used for rendering purposes and contains computation logic. In contrast, the server application maintains the database storage. As discussed, the proposed architecture is divided into two coupled applications as shown in figure 3-1.

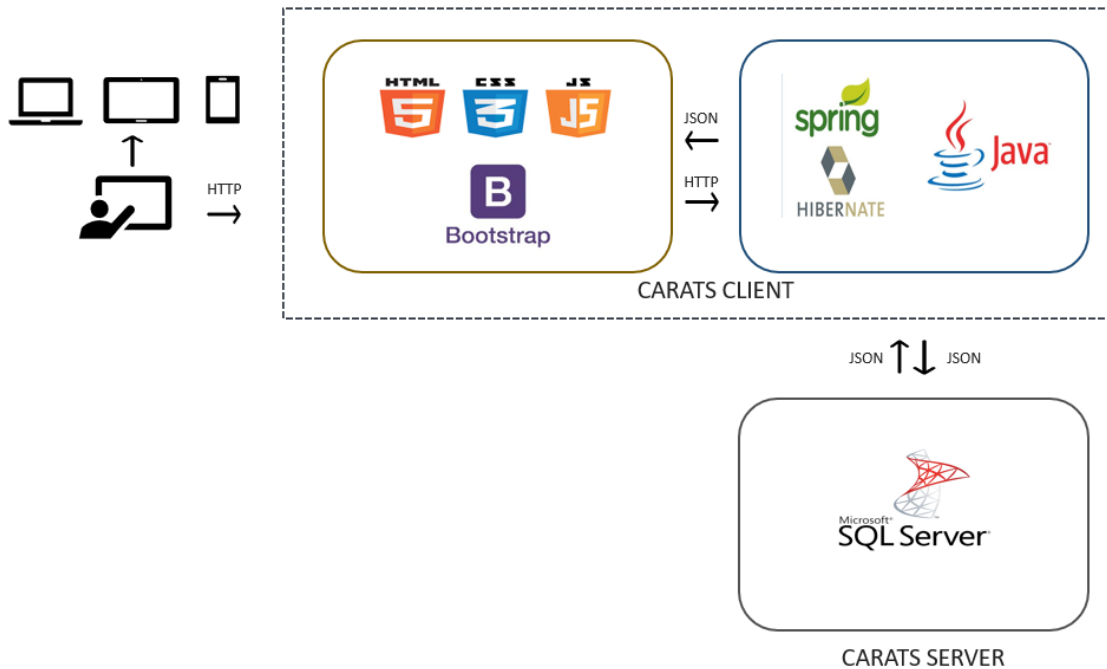


Figure 3-1: CARATS proposed architecture

3.3 Decomposition Overview

The overview of proposed architecture functional decomposition is explained in this section. All component applications in CARATS are coupled together and interact in a synchronous manner.

3.3.1 Client Overview

The CARATS client is an angular and Spring JSP web SPA that follows MVC pattern. It uses bootstrap and data tables for rendering the pictorial representations. The user can only access the CARATS client in the proposed architecture, and its capabilities are listed below as follows:

1. Retrieving and highlighting the important information to residents on their individual dashboards,
2. Creating customized analysis reports by using the application,
3. Motivating the residents with innovative achievement badges,
4. Achieving less communication overhead between the applications,
5. Informing the performance growth of the residents' performance using the daily feedback stock chart visuals,
6. Predicting the residents' status by mathematical calculations and presenting through residency score scorecard,
7. Notifying residents the number of cases completed during the coursework through the bar graphs and listed cards,
8. Providing the summary of all simulation, grand rounds evaluations results to the residents,

9. Reporting all kind of residents' evaluations in detailed table format to the medical examiners,
10. Communicating the progress status in each evaluation in percentage computations,
11. Importing residents case log details to the application in Excel spreadsheet format.

3.3.2 Services Overview

CARATS services is a Java microservices component and acts as the controller in the proposed architecture. It uses the standard HTTPS protocols to transfer data in JSON format. The configured application services can switch persistence providers without changing the code. To facilitate the maintenance of applications, it supports third-party libraries and a carefully chosen JDK versions. The APIs used in the application are intuitive and can hold across many versions. There are no circular dependencies between packages, which helps to maintain the clean code structure. The responsibilities of the proposed architecture include the following:

1. Synchronously facilitating the client-server interaction,
2. Exchanging the JSON data,
3. Authenticating and maintaining user sessions,
4. Verifying of roles before accessing the controllers using the roles structure,
5. Searching, sorting, and filtering operations on data,
6. Computing the mathematical calculations before sending to the client in order to show the numbers effectively,

7. Protecting the application against malicious attempts for manipulation or data retrieval,
8. Validating the client-server handshake requests.

3.3.3 Database Overview

CARATS uses Microsoft SQL server storage to store and retrieve data. The SQL server database stores all the data like user login information, evaluations data, and so on. The following are the capabilities and responsibilities of the database:

1. Setting up development and production environments,
2. Assigning the permissions based on the level of database access to the user,
3. Performing backup and recovery,
4. Query performance tuning at the physical server level and query tuning,
5. Authenticating and maintaining user sessions while working on the database,
6. Setting up the authorized handshake of data between server and database,
7. Utilizing the concept of views for faster retrieval of data.

3.4 Technology Stack

This section explains the technologies used in the CARATS client-server application. As proposed in the architecture, this application was developed using JAVA, HTML, JS, and CSS. Details of programming interfaces used in the client-server architecture are explained in subsections below.

3.4.1 AngularJS

AngularJS is a JavaScript-based front-end web application framework developed to address challenges in single page applications. It is a declarative programme used to create user interfaces and connect components. It helps to simplify the development of client-side model-view-controller (MVC) and model-view-view-model (MVVM) architectures. AngularJS framework reads HTML page customized tag attributes and interprets attributes as directives to bind input or output parts of the page. The uses of AngularJS programming interface are described below as follows:

1. MVC: AngularJS manages MVC components and serves as the pipeline that connects with minimal coding.
2. Interface: AngularJS uses HTML declarative language which is intuitive and less convoluted than defining the interface in JavaScript language.
3. POJO Data models: Angular data models follow plain old JavaScript objects (POJO) and do not require getter and setter functions. Properties can be changed directly in these models code looks cleaner. Data models are responsible for data persistence and server syncing.
4. Directives: Directives manipulate the DOM to simulate and help to invent their own HTML elements. They act as a standalone reusable elements.
5. Filters flexibility: Filters filter the data, including formatting decimal places on a number, reversing the order of an array, or in pagination. Filters are similar to directives; they are also standalone functions. Filters help to create a sortable HTML table without writing any JavaScript.
6. Less code: All the above points up till now mean that less code and AngularJS helps to write their own MVC pipeline. For example, since directives are sepa-

rate from application code, they can be written by a parallel team with minimal integration issues.

7. DOM manipulations: Traditionally, the view modifies the DOM to add behavior, manipulate, and present data. In Angular, DOM manipulation code should be inside directives. But Angular sees the view as just another HTML page like a placeholder for data and the DOM manipulation code placed inside the directives but not in the view. MVC is all about presenting data into views, and not having to think about manipulation. This made angular web app development easy.
8. Service providers: Services involve with the MVC of application, and they provide an outward API to expose data. Most of the time it maintains an offline data by syncing with the server to push and pull data. Services are designed in AngularJS to be standalone objects.
9. Unit testing: Angular is linked by Dependency Injection (DI), manages controllers and scopes. Because all controllers pass information depending on DI, Angulars unit tests can measure the output and behavior. Angular is equipped with a mock HTTP provider to inject fake server responses into controllers. [37]

3.4.2 Spring

Spring framework makes it easy to create Java enterprise applications with the flexibility to create architectures depending on an applications' needs. Spring framework applications can choose modules from a vast range integrated with the framework. It provides foundational support, including messaging, transactional data and persistence, and web. The framework is developed around DispatcherServlet which forwards requests to handlers. The new mechanism Spring developed in 3.0 version

helps to create RESTful websites and applications through the `@PathVariable` annotation. The uses of the Spring framework programming interface are described below as follows:

1. Clear separation of roles: There is a specialized object for every role like Controller, validator, form object, model object, DispatcherServlet, and so on.
2. Adaptability, nonintrusiveness, and flexibility: The Spring framework has many controller method signatures the application needs. A few of the parameter annotations are `@RequestParam`, `@RequestHeader`, and `@PathVariable` for a given scenario. Annotations help the framework adaptability and flexibility..
3. Reusable business code: One of the most important advantages of the framework is avoiding duplication of code. The framework allows the application to use existing business objects as command or form objects instead of copying them to extend a base class.
4. Binding and validation: The framework offers customizable binding and validation like type mismatches as application-level validation errors, localized date, number binding, and manual parsing String-only form objects and converting them to business objects.
5. Handler mapping and view resolution: Handler mapping and view resolution in the framework are customizable, and their strategies range from simple to sophisticated URL-based configuration.
6. Model transfer. The model transfer is flexible and its name/value map supports easy integration with any view technology.
7. Spring tag library: Spring tag library is a simple yet powerful JSP tag library. It provides support for data binding and themes features. The custom tags in the framework allow maximum flexibility in terms of markup code.

8. Configuration: Spring framework has a powerful and straightforward configuration. This configuration includes easy referencing between contexts like web controllers to objects. [38]

3.4.3 Microsoft SQL Server

The Microsoft SQL Server is a software product designed by Microsoft to store and retrieve data as requested by other applications. The applications might be on the same computer or another computer in a network. The SQL Server follows the relational database management system principles. The store and retrieve operations can be invoked on the SQL Server through Tabular Data Stream (TDS), which is an application layer protocol used to exchange data between a database server and other software applications. The SQL Server developed in a row-based table structure that relates data in different tables. It avoids the need to store data in multiple places within a database. The relational database model provides referential integrity and other integrity constraints to maintain data accuracy. The uses of the Microsoft SQL Server framework programming interface are described below as follows:

1. Reliability: In random situations, the client machine may crash while writing data. This will cause the back-end database to also crash and become corrupt. But in the SQL Server there is an intermediate stage created called intelligent data manager, which helps ensure that the clients do not talk directly with the tables. The intermediate system reads and writes data to and from the tables. If a client machine crashes, the intelligent data manager realizes the situations and avoids committing the partially transmitted data. Therefore, the database continues to perform the upcoming transactions. The system will also maintain an automatic transaction log for future references. If a backup needs to be restored the transaction log helps to restore all transactions up to the crash.

2. **Data Integrity:** SQL Server data integrity is enhanced by the application of “triggers”. Triggers are applied when the transactions or data inserted, updated or deleted. By using triggers, the table level data can be stored and these records help in audit processes because they cannot be forgotten.
3. **Performance:** Every form, report or a query will have access to the certain number of tables across the network from the server to the clients machine. Suppose one of the tables contains 50,000 records. The query looks for a single record, the client pulls all 50,000 records for that single record. It collects 50,000 records over the network, and then 49,999 of these records are thrown away. Contrast this with SQL Server, filtering takes place on the server, because of that only 1 record is transmitted. This usually decreases the time required to pull data, and the amount of data transmitted across the network decreases. The SQL server improve the performance by overcoming the key parameter: time constraint.
4. **Network Traffic:** As can be seen from the previous section, traffic across the network is greatly decreased. This improves network reliability and also improves the performance of the network for the client application.
5. **Scalability:** A file server system is designed for a small number of workgroups and is approximately scalable to 10 concurrent clients. The increase in the number of clients will lead to the rapid performance degradation. SQL Server architecture supports hundreds or even thousands of concurrent users without notable performance degradation.
6. **Excellent Data Recovery Support:** Microsoft SQL Server has a number of features for data restoration and recovery when power losses or improper shutdowns occur. Every individual table can't be backed up or restored, but the complete

database restoration can be made available through the use of log files, caching, and backups. [39, 40]

3.4.4 Hibernate

The Hibernate object-relational mapping tool to the relational database for the Java language is developed to overcome the of limitations of JDBC. Its primary feature is mapping from Java classes to data tables and data types. It also provides data query and data retrieval. It is an open-source framework. It uses HQL and SQL to perform powerful object-relational mapping and to query the database. It is one of the best tools for ORM mappings in Java and can cut down a lot of complexity. It also eliminates the defects from application. The uses of the Hibernate framework programming interface are described below as follows:

1. Object/Relational Mapping: Hibernate helps business logic to access and manipulate database entities through Java objects. Hibernate takes care of aspects such as automatic primary key generation, transaction management, and administrating database connections, and so on, which eventually helps to speed up the performance.
2. JPA provider: Hibernate supports the Java Persistence API (JPA), which consists of specifications like accessing, persisting, and managing data.
3. High performance and scalability: Hibernate achieves high performance by supporting lazy initializations, different fetching strategies, and so on. Additionally, it scales well in any environment.
4. Easy to maintain: Hibernate generates SQL at initialization time and it does not require any special database tables or fields. The data transaction time is quick and the maintenance is easy when compared with JDBC.

5. Layered Architecture: Hibernate has a layered architecture, so there are no restrictions to use every piece of code. The applications can use those features enough for the project.
6. Database Independent: Hibernate is independent of the database engine. It is offered with a list of Hibernate Dialects to connect with the different databases. [41, 42]

3.4.5 Bootstrap

Bootstrap was developed at Twitter as a framework to encourage consistency across their internal tools. Previously, the Twitter team used various libraries for interface development led to inconsistencies and a higher maintenance. Eventually, to maintain consistency, they developed the Bootstrap framework. The Bootstrap framework is a leading responsive, mobile-first framework. Responsive design helps websites that automatically adjust according to the screen size. It helps to adjust from navigation, the organization of content, buttons, and image scalability. The responsive websites are designed to prioritize the information different device users need. It is a free and open-source front-end framework. It holds customizable HTML- and CSS- based design templates with typography, forms, buttons, navigation, and so on. It also supports JavaScript extensions and concerns only with front-end development.

1. Customization: The websites developed using Bootstrap do not look like a Bootstrap website-meaning, the customization of the website completely depends on developers. Some frameworks offer completely coded from scratch and other dropped-in templates, but Bootstrap falls in the middle category. Lots of the work is done in advance, either with the grid system or customizable components. Additional JavaScript effects can also be altered or extended.
2. Grid approach: To stay clean and logical on the smallest screens, Bootstrap

forces designers to build websites with small screens in mind. Bootstraps grid-based layout helps developers to focus on it. Its grid approach played a significant role in its success. Developers are given a choice up to 12 columns in a layout, which are organized in layout classes extra small, small, medium, and large for all types of screens.

3. Base styling: Bootstrap supports styling for all fundamental HTML elements. Elements like typography, code, forms, images, icons, headings, lists, tables, and buttons can be extended using the Bootstrap framework.
4. Components: Bootstrap prestyled most of the components. Some of the components are dropdowns, button groups, navigation bars, breadcrumbs, labels & badges, alerts, progress bars and many others.
5. Javascript plugins: The Bootstrap packages made components like drop-down menu interactive with the bundled JavaScript plugins. It also provided JavaScript plugins to the sliders, tabs, accordions, and so on. Adding these functionalities takes only a few additional lines of code. [43, 44, 45]

3.5 Data flow in CARATS Architecture

The overview of the steps involved in the data flow across the application is shown in figure 3-2. The process is explained in the basic note below and will be further described with detailed descriptions in forthcoming sections.

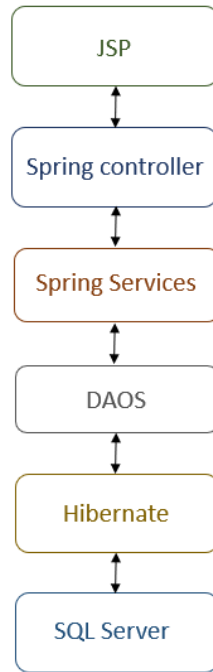


Figure 3-2: Data flow in CARATS

1. Javascript sends data in JSON format with a GET request to a CARATS Controller back end.
2. CARATS Controller doesn't process any data, it just transfers to the CARATS service.
3. To implement the logic, the CARATS service needs additional data from the database. So it transfers the protocol to ServicesDAO to establish a connection with the database.
4. CARATS ServicesDAO then calls the CARATS repository to process the query and to retrieve required data. The data goes back to CARATS ServicesDAO for further process.
5. CARATS ServicesDAO returns the data to the CARATS service to finish its

logic and handles the result to CARATS Controller.

6. The CARATS Controller then transforms the data into JSON and returns it to the front end

3.6 Client Architecture

In this client architecture, the user requests either by typing a URL, clicking a link, or submitting a form. It goes to the server through the POST or GET requests. The response to these requests will become either the retrieving the file or output of the environment. Both will be constructed into an HTTP response to pass information to further stages. When the response comes back to the client or browser, the Javascript will start processing it. After finishing the process, the information will be interpreted by HTML and CSS to display content to the user. Once it displays the content, the user starts interacting with the interface, which enables Javascript to perform a few more operations. Initially while loading the page, it will be packed with CSS files, JavaScript files, images, and other content.

Javascript runs according to the user interactions with the displayed content through the HTML and CSS. When something changes in the page, the Javascript also helps in refreshing it. The user interactions will only advance to the server when the user requests a URL or clicks a link. But there is another scenario where the requests will be sent to the server. When the page has an AJAX request with GET and POST parameters, it will be forwarded to the server through the Javascript. These AJAX requests are asynchronous, and a callback function is registered to deal with the response. The AJAX response will not trigger a page reload as it will be handled by the callback function. An AJAX request has a URL to relay the current location, data of the form submission or other data, and callback function for the response.

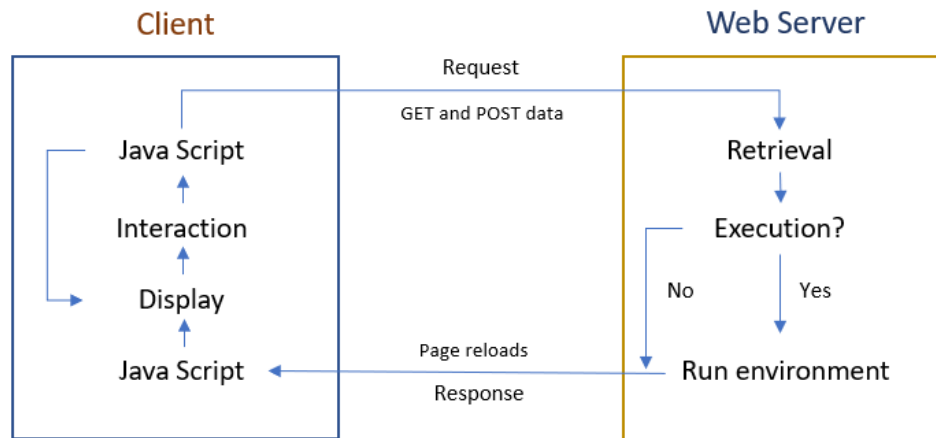


Figure 3-3: Flow of data in client architecture

The client architecture is developed in a way to avoid cross-browser compatibility issues. The transfer of data happens in JSON (Javascript object notation) format. JSON is a lightweight data-interchange format, and it is based on a subset of Javascript. It is a completely independent data-interchange language but uses C, Java, Javascript, Python, and many others. The JSON is developed on two structures: a collection of name/ value of pairs and ordered list of values. The ways of transferring data types using the JSON format explained below as follows:

1. The object begins with (left brace) and ends with (right brace) and each name/pair is followed by : (colon) and it is separated by , (comma).
2. An array begins with [(left bracket) and ends with] (right bracket) and the values are separated by , (comma).
3. Values can be a number, or true or false or null, or string in double quotes.
4. A string very much like a C or Java string is a sequence of zero or more Unicode characters enclosed in double quotes. A group of characters makes a string.

5. Except that the octal and hexadecimal formats, the regular number used in a C or Java programs are used in JSON.
6. Whitespace is allowed between any pair of tokens. [46, 47]

Responsive Web design: It is an approach to design and develop a website that will respond according to the users behavior and environment based on screen size, platform, and orientation. The type of devices CARATS application supports is shown in figure 3-4.



Figure 3-4: The responsive design of CARATS works on these devices

To implement responsive web design in the application, it use a mix of flexible grids and layouts, images, and an intelligent use of CSS media. If the user requests a website from a tablet or laptop, the website automatically switches its design to accommodate for resolution, image size, and scripting abilities. As the website is deployed for users irrespective of place, the application will not consider the settings like VPN on their devices.

Responsive sites use fluid grids for rendering in different screens. All page elements in the application are sized by proportion, rather than pixels. So if the application has three columns, it works on the principle of how wide they should be in relation to the other columns than individually. The columns widths are defined in the ratio

like first column half the page, the second column 30%, and the third column 20%. Media content like images is also resized using relativity. The images stay within their relative design element.

Issues to overcome for responsive design: This section describes the problems rectified for better performance of the application:

1. Mouse vs. touch: To design for mobile devices, the applications need to overcome the issue of mouse versus touch. On regular desktops, the user handles navigation and selects items with a mouse. On a smartphone or tablet, the user mostly uses fingers for the same purposes. The web application must consider touch into a mouse click. The CARATS application uses in-built plugins to differentiate and work accordingly.
2. Graphics and download speed: The next major issue is graphics and files download speed on mobile devices. The application is optimized in a way that file download speed is similar on all devices. The graphics used for analytics are high end with low configurations, so it works same across platforms.

To overcome responsive design issues and to render according to the screen's fluid grids and media queries are used in the application. A detailed explanation provided below:

Fluid Grids: The usage of the fluid grid is a key behind responsive design. The fluid grid is getting popular nowadays by moving liquid layout aside. The liquid layout creates fixed-width layouts and page designs. Whereas the fluid grid considers page design is fixed, but the width layout varies according to the layouts. The fluid grids follow the traditional liquid layout. Instead of designing a layout based on pixels, it does so in terms of proportions. This helps when pixels squeezed into a mobile device

or stretched to a larger screen computer or laptop, all of the elements will resize width in relation to one another.

To calculate the proportions, the screen is divided by its content. Let's say the screen is a width of 900px and the content is 300px the system calculates the fluid layout in relation width using the below equation. The result is 30 percent. So the content will be given 30 percent of the pixel width across all screens.

$$\text{Fluid layout in relation width} = \frac{\text{Content width}}{\text{screen pixel size}}$$

Fluid layouts can take us only up to a certain limit in responsive design. If the screen is too small, then the content will be portrayed in an unreadable manner. To resolve these kinds of issues the media queries are used in the application.

Media Queries: The solution to implement an effective responsive design for smaller screens is CSS3 media queries, which is having a decent support from many modern browsers. The application uses the conditional CSS styles to implement the responsive design.

For application purposes, we used the max-width media feature, which allowed us to apply specific CSS styles. If the browser window drops below a particular width, then conditional CSS styles become active. The sample conditional CSS code for a screen size 400px is shown below:

```
@media screen and (max-width: 400px) {  
    .content {  
        float: left;  
    }  
    .ut_icons {
```

```

        display: none;
    }
    // and so on ...
}

```

In current industrial situations, every device is manufactured with different screen sizes. The application is designed for mobile, tablet, and computer (both desktop and laptops) by assuming the pixel sizes, which are shown in table 3-1. The mobile maximum screen size is considered or limited to 479px. Whenever the screen size exceeds more than the mobile pixels limit, the application will consider the device as a tablet. The process goes on by assuming the device screen sizes as mentioned in the table.

Table 3.1: The screen sizes implemented in the application

Device	Max-width
Mobile	479px
Tablet	767px
Laptop	991px
Desktop	991px

3.7 Server Architecture

The server architecture processing flow from receiving the request until returning the response is shown in the 3-5 figure. The straight lines rounded with numbers one to eight is the actual flow of data from front-end requests to the response to it. The dotted lines are positioned in the architecture to represent the flow that quietly helps to further process the data. The flow of the data is explained below in eight steps, which are noted in the architecture as well.

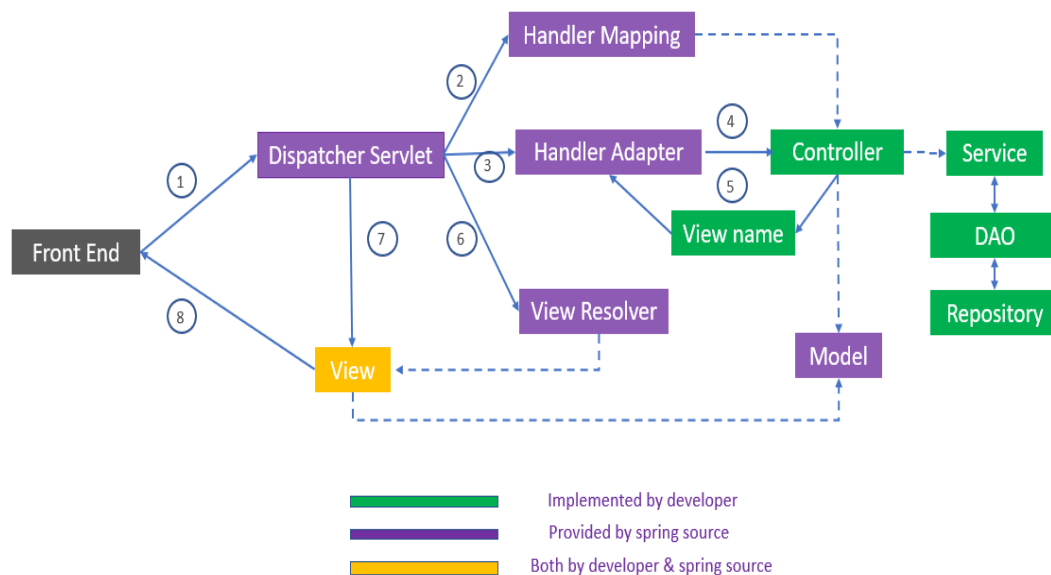


Figure 3-5: Flow of data in server architecture

1. DispatcherServlet receives the incoming request to the server.
2. DispatcherServlet selects an appropriate HandlerMapping and then sends the task. The HandlerMapping decodes the approached URL and maps the controller depending on that information. After mapping the controller, the HandlerMapping will return the mapped controller response to the DispatcherServlet.

3. After receiving the mapped controller information, the `DispatcherServlet` dispatches the execution task of the business logic of controller to `HandlerAdapter`.
4. `HandlerAdapter` will then calls the business logic process of the controller according to the information provided by the `DispatcherServlet`.
5. The controller performs the business logic and sets the processed results in Model. The controller then returns the logical name of the view to the `HandlerAdapter` for further processing.
6. `DispatcherServlet` forwards the task of determining the view according to the view name to the next block `ViewResolver`. The primary role of `ViewResolver` is to return the view mapped to the view name to the `DispatcherServlet`.
7. The rendering process will then be transferred to returned view by the `DispatcherServlet`.
8. View returns the response to the front end after rendering the model data. [48, 49]

The details of the individual blocks in the server architecture is explained below.

DispatcherServlet: `DispatcherServlet` is also can be called as the front controller. It is an actual servlet, which is declared in the `web.xml` file while configuring the application. It uses special beans to render the appropriate views and to process requests. The special Beans are configured in the web application context. The `DispatcherServlet` consists of `LocaleResolver`, which is bound to the request to enable elements, `ThemeResolver` is bound to the request to let the elements similar to views, and `MultipartResolver` is to inspect the requested multipart files. [50]

HandlerMapping: HandlerMapping defines a mapping between requests and handler objects. It is included as beans in the framework or can be implemented by the developers. It is always wrapped in an execution chain instance, and its implementations support the mapped interceptors. The HandlerMapping is very powerful in the framework, and even it makes it possible to write a custom mapping based on session state, cookie state, and other variables. [51]

HandlerAdapter: The HandlerAdapter acts as an interface that handles the HTTP requests. It maps a method to a specific URL and is used in conjunction with HandlerMapping. The Servlet has the information about invoking a method but it cannot invoke directly, so it takes help of HandlerAdapter to do that. HandlerAdapter also involves in returning the response as model and view to the Servlet. The HandlerAdapter interface is not intended for application developers but it is available for the handlers who want to develop their own web workflow. [52]

Controller: The controller serves as a specialized component to autodetect the implemented classes through the classpath scanning. Based on the RequestMapping annotations, it is used in combination with annotated handler methods. Controller operates on HttpServletRequest and HttpServletResponse objects like an HttpServlet. It processes the request and the obtained result is returned as a model and view object which the DispatcherServlet will render. Sometimes the null value might be sent to DispatcherServlet, which states that this object completed request processing and no model and view to render. [53, 54]

View: Views are labeled by a view name and resolved by a view resolver. The application supports more than one resolver. View resolvers enable the application to render models without binding to a specific view technology. It provides the mapping between actual view and view names. An AbstractCachingViewResolver takes care

of caching views, `UrlBasedViewResolver` affects the direct resolution of symbolic view names to URL, and `XmlViewResolver` accepts configuration file written in XML. These are some of the view resolver examples and their benefits. [55, 56]

The following are the configurable properties of requested URL into a view name:

1. `prefix`: Prepended string to the generated view name. The default is "".
2. `suffix`: Appended string to the generated view name. The default is "".
3. `separator`: The string separates URI parts. The default is "/".
4. `stripLeadingSlash`: Boolean specifies whether beginning slash should be removed. The default boolean value is true.
5. `stripTrailingSlash`: Boolean specifies whether ending slash should be removed. The default boolean value is true.
6. `stripExtension`: Boolean specifies whether file extension should be removed. The default boolean value is true.

Model: Model is designed for adding attributes, and it acts like a holder. `ModelMap` is an extension to the model and it has the ability to store attributes. It also has the ability to pass the collection of values. The `ModelAndView` is the final interface to pass values to a view. It transfers values to both model and view in one return value. When incoming requests occur, the `@ModelAttribute` annotated methods are called before any controller handler method. Before the execution of the handler methods, the data will be added to the `java.util.map`. [57, 58]

Services: The services used in the application are Restful services. They offer suitable actions like GET, POST, PUT, DELETE, and so on, caching, redirection and forwarding, and finally security like encryption and authentication. The restful

services support backward compatibility; they are scalable and securable services with evolvable APIs, and they also have the spectrum of stateless to stateful services. The following are the common annotations used in the application:

1. **@RestController**: **@RestController** is an advanced version of the controller. It holds both **@Controller** and **@ResponseBody** annotations, which simplifies the whole controller implementation. The usage of the **@RestController** eliminates the requirement of **@ResponseBody**.
2. **@RequestMapping**: This annotation decodes information like method type GET/-POST and defines the URL of the resource. There are three parameters Headers, Value, and Method, which are defined on top of the body to describe the functionality. The value contains the path information; the headers carry the authentication and format of the data; and the method will define the request type GET, POST, and so on.
3. **@RequestParam**: This annotation helps to find the required parameter defined in the URL. It decodes the parameter and transfers it to the body to perform the business logic. Along with the declaration of the annotation, the type of parameter to search must be defined next to it. It helps the application to mark the particular data. If the parameter name and targeted name are the same, the value element can be skipped. Methods can have many **@RequestParam** annotations as there are no constraints on it.
4. **@RequestBody**: This annotation represents the body of the request. It maps the **HttpRequest** body to transfer Java object by enabling the automatic deserialization. The application deserializes JSON into a Java type automatically sent from the client-side controller. Based on **Accept** header present in the request, the application uses the **HTTP** message converters to convert the deserialize request body into the domain object. [59]

Representation State Transfer (REST) is an architecture style designed for distributed systems. The application uses this design to further strengthen the determining the controller and carrying the data to the client through the architecture. REST is a uniform interface and it is not related to HTTP, but it commonly associates with it. The principals of the REST are described below:

1. Resources: It exposes the directory structure Uniform Resource Identifier (URI).
2. Representations: It transfers JSON or XML to represent attributes and data objects.
3. Messages: It uses HTTP methods explicitly.
4. Stateless: Between requests, the interaction's information stores no client context on the server. State dependencies restrict the scalability and they are limited. The client-side only holds a session state but not the server side. [60]

The application uses HTTP methods to map create, retrieve, update, and delete CRUD operations to HTTP requests. The requests are never cached, and they do not remain in the browser history. The bookmarking of the requests is not possible and there are no restrictions on data length. The following are the HTTP methods used in the application:

1. GET: GET request helps to retrieve information, and they are safe and idempotent. It means the results are same regardless of how many times the system repeats with the same parameters. They cannot be critical to the operation of the system as the user doesn't expect them. They might have side effects, but they are negligible. The GET requests can be either partial or conditional. When dealing with the sensitive data, the GET requests cannot be used. They are only good to request the nonsensitive data.

2. POST: This is a request method designed to accept the data enclosed in the body of the request message. It is used for storing the data and sometimes when uploading a file or while submitting a completed web form. It sends a representation of new entity of data to the server. Along with the POST request, additional headers will be sent Content-Type: like application/x-www-form-urlencoded and Content-length: provides the length of the URL-encoded form data.
3. PUT: PUT request is a store entity at a URI. It can be used for creating a new entity or for updating an existing one. Similar to GET request, PUT request is also idempotent. The idempotent nature is the major difference between the PUT versus a POST request. In case only a subset amount of data are requested to replace through PUT request, the system will replace rest of the data with either null or empty. The PUT request works like if the Request-URI doesn't exist it will create one or else it modifies original. It also allows the update or creates a resource with the same URL object.
4. PATCH: PATCH request is designed to update only the specified fields of an entity at a URI. This request is neither safe nor idempotent because its operation cannot ensure the entire resource has been updated. For instance, the PATCH request will be handy if the application needs to update only one field in the entire entity. The usage of PUT here replace the other entity fields with either null or empty. This method affects the resource located by the Request-URI. It may create some side effects to the other resources like creating a new resource or modifying the existing one. The PATCH will reduce the workload to change the other fields in the entity.
5. DELETE: DELETE request helps to remove a resource either asynchronously or a long-running request. It requests that the server delete the resource by

identifying the Request-URI. The DELETE operations are idempotent. Once the clients request the resource to be deleted, the repetitive of the same request won't affect the system. Since the file has been removed already, if the request is repeated the system just responds with the error file not found.

Exceptions defined in the application: To track errors, we implemented the exception handling in the application. Exception handling is the mechanism that helps to handle runtime malfunctions to prevent the abrupt termination of the program. Programming error, hardware failures, file not found, and resource exhaustion are a few exceptions that can happen the program executes. The application is built with methods to store the exception error codes and their location in the database. This helps to find the error location and resolve. Most of the exception handlers generated error codes are restricted to display on the client web pages. Exceptions can occur at runtime known as runtime exceptions and at compile time known as compile-time exceptions. Predefined and user-defined are the two types of exceptions present in the application.

Predefined Exception: Predefined exceptions are supplied as a part of JDK. These exceptions are developed for global or universal problems. A few universal problems are as follows:

1. Division by zero.
2. Invalid bounds of the array.
3. Invalid formats of a number.

User-Defined Exceptions: User-Defined Exceptions are developed by Java programmer. It is supplied as a part of the project to deal with common specific Problems. A few problems are as follows:

1. Entering negative ages for human beings.

2. Entering negative salaries for employees.
3. Trying to deposit the negative amounts.

In Java, exception handling is done by using five key concepts. They are try, catch, throw, throws, and finally. Among the five, we used only two try and catch in the application along with the custom exception handlers.

Try: Try is a predefined block. It can be loaded with a block of statements that will cause problems at runtime. The problematic statements are highly recommended to write in this block, which helps during the exceptions. If an execution error occurs in the try block, the program abnormally terminates the current execution and entries into appropriate catch block. Once the request went to the catch block, control never comes back to try block. Every try block must be followed with a catch block. One try block can have many catch blocks. It is preferred to have at least one catch block. Multiple catch blocks are used for generating user-friendly error messages. The nested/inner try blocks can be used in the application.

Catch: The catch block is also a predefined block that will be executed when an exception occurs in the try block. Only one catch block is allowed to execute when multiple blocks are present. The catch can accommodate try and catch blocks in it. The catch block can handle all type of exceptions like `ArrayIndexOutOfBoundsException` or `ArithmeticException` or `NullPointerException`. If there are no problems in executing the try block then the catch block or blocks associated with that try block, will be ignored. Catch blocks execute for a specific type of exception like `catch (ArithmeticException e)` that handles `ArithmeticException`.

The application architecture is defined with 40 standard status codes. They are used to convey the results of a clients request. The error codes are divided into the five categories. The status information and their series code are presented in the below table 3-2.

Table 3.2: Error code series and their information

Series	What for
1XX	informational
2XX	success
3XX	redirection
4XX	client error
5XX	server error

- Informational: Protocol-level information will be displayed.
- Success: Indicates request accepted successfully.
- Redirection: In order to complete the request client must take some additional action.
- Client Error: Error status codes points the finger at clients.
- Server Error: Error status codes points the finger at the server. [61]

The table below summarizes the HTTP methods recommended return values when used in combination with the resource URIs.

Table 3.3: The error codes & their information

HTTP Verb	Error codes
POST (Create)	404 (Not Found), 409 (Conflict) if resource already exists
GET (Read)	200 (OK), single field. 404 (Not Found), if ID not found or invalid
PUT (Update/Replace)	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid
PATCH (Update/Modify)	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid
DELETE (Delete)	200 (OK). 404 (Not Found), if ID not found or invalid.

DAO: The application allows switching between persistence technologies without worrying about exceptions that are technology specific. This framework provides helpful translation from technology-specific exceptions. The framework exceptions wrap the other technology exceptions like SQLException, so there is never any risk of losing the important information. Application DAO is a Java component follows data access object (DAO) pattern and uses Hibernate- like services to establish sessions and make calls to the database. The advantages of DAO are as follows:

1. It hides all the data access implementation details and enables transparency by doing that.
2. It allows to clean-up the resources automatically.
3. If the database provider is changed it allows easy migration.
4. It reduces architecture and code complexity by centralizing the separate data access layer.

Hibernate technology is used in the application for DAO pattern. The architecture that allows transactions between the server and the database is shown in figure 3-6.

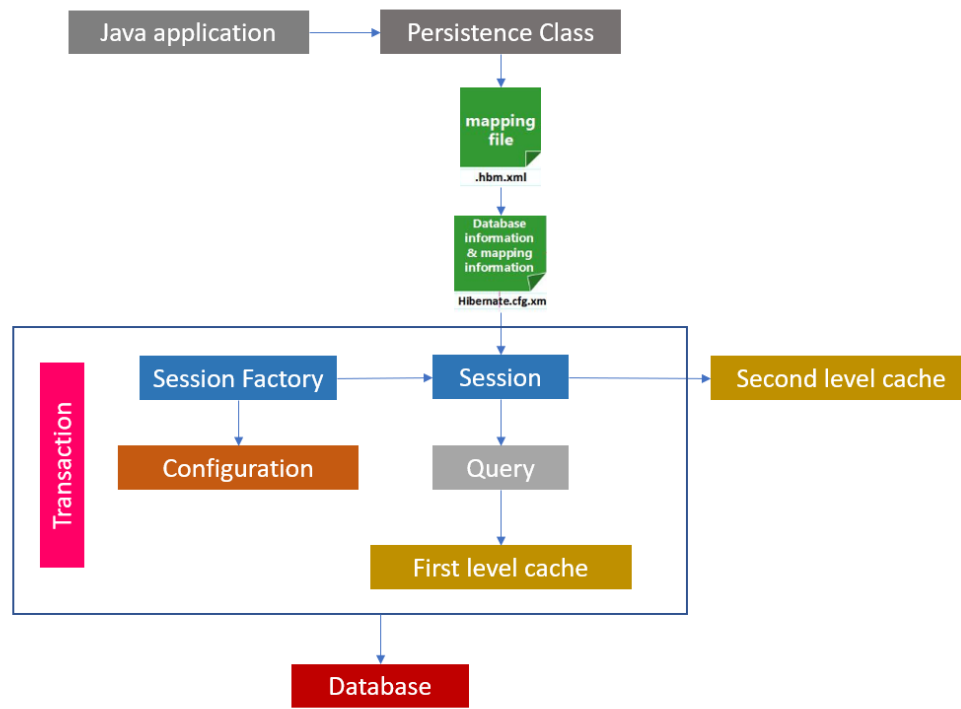


Figure 3-6: DAO hibernate architecture

1. Configuration: The configuration details are stored in hibernate.properties or hibernate.cfg.xml files. The files are annotated with the @configuration, which helps to load the files when a request is placed. The system uses these details to establish a stable connection with the database. It uses Session factory to work with both the database and the Java Application. The entire set of mappings to an SQL database of a Java types application are represented in these files. There is an additional feature in the hibernate properties to store all development, testing, and production database details in a single file.
2. Session Factory: It is a factory for Session objects and created by providing

the Configuration object. The configuration object contains the database related property details. Session factory pulls these details either from the hibernate.properties file. The application requests the Session factory for a session object to initiate the connection. If the application refers to development, testing, and production databases, then it needs to create one Session factory per database.

3. Session: It represents the interaction between the application and the database at any point in time. This is represented by the org.hibernate.Session class. The SessionFactory bean will have the retrieval information for the session instance. Session main functions are to offer, create, read, and operations. At any given point in time, the instance will be in the following states:

- Transient: The object has just been instantiated using a new operator and not associated with a Hibernate session.
- Persistent: Associated with a Hibernate session and detects changes made to an object. It synchronizes with the database until the work completes.
- Detached: Object has been in the persistent state, but its session has been closed. It can be reattached to a new session at a later point in time.

Before saving and updating the row data in the database table, every entity object passes through three states of the object as per as given in the following figure 3-7.



Figure 3-7: Hibernate Transaction session states

4. Query: It allows the application to query the database. This framework provides two different techniques NamedQuery and CriteriaAPI to query the database. Both can be used for one or more stored objects. The details of NamedQuery and CriteriaAPI are explained below:

- NamedQuery: Named query in hibernate is a JPQL or SQL expression with pre-defined queries. This can be defined either in mapping file or in an entity class. In this application, it is defined in an entity class. @NamedQuery and @NamedQueries annotations are used for HQL or JPQL expressions. Annotations like @NamedNativeQuery and @NamedNativeQueries are used for native SQL expression. The application also supports the queries via stored procedures and functions. The stored procedures and functions return a result set as the first out-parameter. The Named queries help while executing the same queries multiple times in more than one application. If the configuration is written in the mapping file, the developer needs to use getNamedQuery() given by session interface. The calling list() is used for getting the Query reference to execute queries.
- Criteria: The Criteria API in this framework allows the application to build up a criteria query object programmatically. The org.hibernate.Criteria interface defines the methods available depending upon the objects. The

hibernate interface used in this architecture contains several overloaded `createCriteria()` methods. The application uses an entity name or object's class to create criteria methods. When the application is executing the criteria query, it returns instances of the persistent state object's class. The criteria query also returns all objects that correspond to the class. The Criteria API can be used to apply restrictions in your queries to selectively retrieve objects the way the application could retrieve only residents with 70% of residency scores. This framework also allows the application to do more complicated queries with the help of Criteria API.

5. First-level cache: While interacting with the database the First-level cache is used as the default cache for Hibernate Session object and there is no option to change into disable status. However, hibernate allows for the deletion of selected objects from the cache or to empty the cache completely through some methods. The cached memory created or stored for a session is not visible to other sessions. Once the session is closed, all objects stored in the cache memory will be cleared. The First-level cache is also called as a session cache. It caches objects within the current session. All-Session object requests to the database pass through the first-level cache. Until the next Session object is live, the first-level cache is available with the session object. This feature helps in not firing the query again and again within the same session. Different methods used to clear the session objects in the application are shown in table 3-4.

Table 3.4: Syntax to clear session objects used in the application

Syntax	What for
evict()	to remove a single object from the first level cache.
clear()	to clear the cache i.e delete all the objects from the cache.
contains()	to check if an object is present in the cache or not.

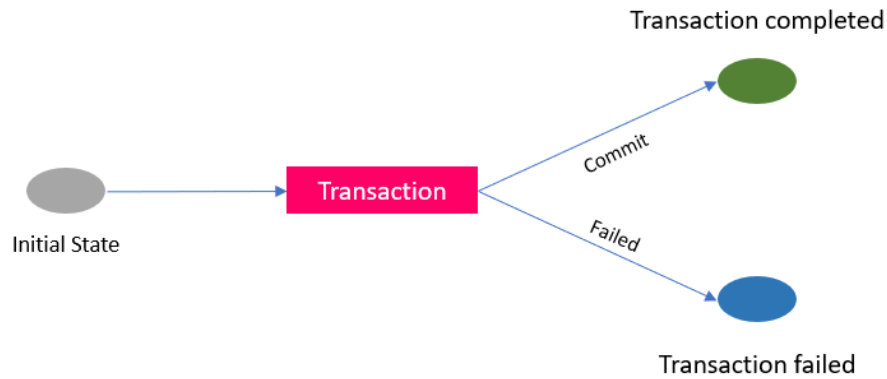


Figure 3-8: Hibernate Transaction

- Transaction: Transaction enables the application to achieve consistency in data transfer. It also helps to roll back the data in case something goes unexpectedly. The role of the transaction is stated in figure 3-8. The commit is a statement and a state that informs the database that the transaction is finished and to proceed with the storing process. The failed state describes something unexpected or error in code that occurred in the system, so don't store the data transferred

after the last commit. The syntax used to implement the transactions in the application are shown in the table 3-4.

Table 3.5: Syntax used to implement transactions in the application

Syntax	What for
begin()	to start a new transaction.
commit()	to end the unit of work unless in FlushMode..
rollback()	to force the transaction to rollback.
registerSynchronization()	to register a user synchronization callback for the transaction.
wasCommitted()	to check if the transaction is committed successfully.
wasRolledBack()	to check if the transaction is rolled-back successfully.

7. Persistent objects: Persistent objects are configured using annotation @Entity.

They are plain old Java objects called as POJOs created to represent rows in the table in the database.

- Default constructors are created to persist all Java classes.
- All classes are created with an ID to allow easy identification of the objects within Hibernate and the database. This property helps to map the primary key of a table.
- All attributes are declared private and their getter and setter methods are defined in JavaBean style.
- The central feature of Hibernate, proxies which depends on the persistent class implemented. It acts like an interface that declares all public methods.

8. Second-level cache: It helps to store objects across sessions. Application explicitly enables this cache and it requires to provide the cache provider for a second-level cache. EhCache is one of the external cache providers. As there is no need of second cache in this application, this is not enabled or borrowed from providers. [62, 63, 64, 65, 66]

Application query flow: The application query flow to query the database and return results presented in a flow chart in figure 3-9 and the steps followed are explained below:

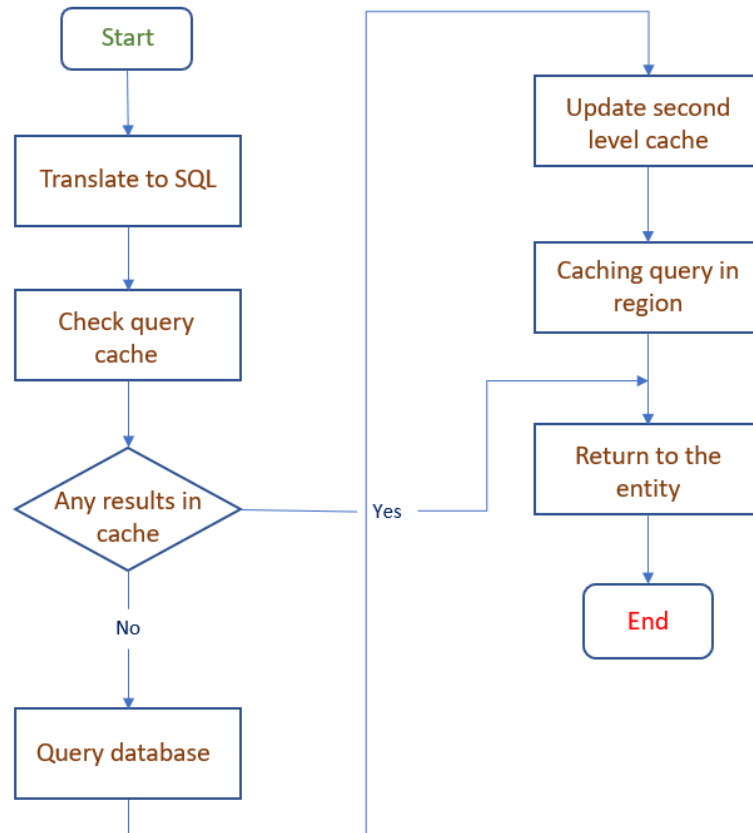


Figure 3-9: Application query flow

1. The DAO starts the process when the application services send a request for data from the database.
2. In the first step, the HQL written query is translated into a SQL server query, as the database used in the application is Microsoft SQL server. The DAO architecture is designed in such a way that a small change in the configuration file adjusts the database language. According to that configuration change, the application automatically converts the HQL or Criteria-based queries.
3. After translating the query, the system checks for the required data in the first-level cache.
4. Depending the availability of the data in the first-level cache, further steps will be decided. If the data are available in the system, then it returns to the entity or proceeds to query the database. While proceeding to the next step, the system changes its current status from a transient state to persistent state.
5. The systems execute the translated query in this stage in the server database according to the environments. As discussed in above sections, the server databases can be a development or testing or production environments.
6. The result from the executed query is then transferred to the second-level cache. As mentioned the current application is not borrowing any third-party cache system, so the process moves to the region cache step.
7. The system clears the existing session entity data in the first-level cache and starts a new session. The returned new query results will be stored in the first level cache.
8. The system returns the entity to the server services and then closes the session by changing the state to detached. The first level cache data will not be cleared until the next session requests to clear. [67, 68]

3.8 Database

A Database is a collection of related data stored for the purposes of easy access, management and updating. The database can be based on software or hardware. Database Management System (DBMS) is a software developed to create and manipulate a database based on users' input. It allows the user to analyze data easily. It is provided with an interface to create a database, store and update the data, create data tables in the database, and more. It also protects and secures the databases. In case of multiple user requests, it maintains the data consistency.

Database architecture in this application was developed focusing on the design, development, implementation, and maintenance of residents' evaluations data. DBMS can be centralized or decentralized or hierarchical, depending upon its architecture.

- Centralized: All the data will be stored at one location.
- Decentralized: Multiple copies of a single database are stored at different locations.
- Hierarchical: Data are organized into a tree-like structure.

The operation of DBMS will depend on the tier of architecture used for the application. In 1-tier architecture, the user can store data directly in the database. This setup helps programmers to directly communicate with the database for a quick response. 2-tier architecture has an application layer between the user and the DBMS. It is responsible to communicate the requests and the responses. In 3-tier architecture, there is an additional presentation or GUI layer combined to 2-tier. For the end user, the presentation layer is the DBMS. Additionally, the end user will have no idea about the application layer and the database. The CARATS application is using the 2-tier architecture, so only application layer is presented in between the database and the user.

The application database is developed by following the Relational Database Management system (RDBMS) Normalization techniques. The Database Normalization is a technique that helps in the effective organization of the data. It is a systematic approach to decompose tables in order to eliminate data redundancy. Additionally, undesirable characteristics like Insertion, Update and Deletion Anomalies will be handled in a better way. Normalization is used in the application for mainly two purposes:

- To eliminate useless data.
- To ensure the data are logically stored.

An anomaly in the database is a variation that might occur when update, insert, and delete operations differ in some way from what is considered normal. In databases update, insert, and/or delete operations need to be as straightforward and as efficient as possible. The anomalies and problems that occur with them are explained below as follows:

- Update Anomaly: It exists when one or more instances of duplicated data are updated. If the residents' evaluation is updated twice or thrice, then it disturbs the aggregation of the data for analytics.
- Delete Anomaly: It exists when certain attributes are lost because of the deletion of other attributes. Each resident's evaluation results are interlinked with a unique ID across all tables. If that unique ID is deleted instead of one evaluation result, then all the records connected to the residents will no longer be visible under them.
- Insert Anomaly: It occurs when certain attributes cannot be inserted into the database without the presence of other attributes. All evaluation results are

distinguished with the help of a unique id. If that id is repeated in an instance then it ignores the insert operation. [69]

To make the operations as efficient as possible the system utilizing the four normalization techniques. If the data tables are created without following the normalization techniques then it will have data redundancy. The data redundancy makes it difficult to handle the database and it will consume more memory space. It not only causes data loss but also creates a frequent Insertion, Updation, and Deletion anomalies. The four normalization techniques followed in the application are explained below:

1st Normalization form: The 1st Normal form accepts the table design in a way that it can easily be extended and easier to retrieve data from it whenever required.

Rules for the First Norm:

1. Rule 1: Single Valued Attributes: Each column of the table should be single-valued. It means they should not contain multiple values.
2. Rule 2: Attribute Domain should not change: Column of the table should maintain the same format of entire data in it. It means if the column is designed for a date of birth, it cannot be filled with text.
3. Rule 3: Unique name for Attributes/Columns: To avoid confusion or navigating to an unintended column, each table should maintain different names for all columns.
4. Rule 4: Order doesn't matter: Because each row is treated as same as other ones, the order of inserting the data doesn't matter.

Second Normal Form: To satisfy the Second Normal Form, the table must have two conditions:

1. The table should follow all the rules in the First Normal Form.
2. The table should have no Partial Dependency.

The partial dependency exists when at least one column depends on only one key among the composite key. The composite key is the group primary keys which help to retrieve unique data according to the situation. For example, the table is created to store the user login details. The form is developed with additional fields like profession, kind of access along with username and password information. The profession of the user related to the user but not on the type of access they required. Sometimes the admin access is given to the staff who takes care of creating new users into the system. So the role is partial depends on the user but not on the type of user. These details must be placed in different tables to decrease the data redundancy. The Second Normal Form is followed in the application to remove the data repetitiveness.

Third Normal Form: The below are the two conditions that need to be followed by a table to implement the Third Normal Form:

1. The table should follow all the conditions in the Second Normal form.
2. The table should not have a Transitive Dependency.

Definitions help to understand the transition dependency:

- **Candidate Key:** A candidate key is a key used for identification of row in a table by using two or more columns.
- **Prime Attributes:** The attributes are called as prime attributes when there exists at least one of the possible candidate key relation.
- **Non-Prime Attributes:** Attributes which does not exist in any of the possible candidate keys of the relation are called non-prime attributes.

A Transitive Dependency occurs when rather than depending upon the prime attributes or primary key the non-prime attribute depends on other non-prime attributes. For example, the maximum limit to the daily feedback score is one hundred but whereas the presentation and simulation evaluations are considered each field is given with range from poor to excellent. The theme of calculating the performance is different in this cases. Here, the daily feedback score and both evaluations depend on the user performance but they are mutually exclusive. In this type of cases, the Third Normal Form is used in the system to eliminate the transitive dependency.

Boyce-Codd Normal Form (BCNF): BCNF is an extension to the third normal form. It is also considered as 3.5 Normal Form. The table should follow all the below conditions to satisfy the Boyce-Codd Normal Form:

1. The table should be in the Third Normal Form.
2. And, for any dependency $A \rightarrow B$ in a table, A must be a super key. It means, that for a dependency like $A \rightarrow B$, If B is a prime attribute or column then A cannot be a non-prime attribute.

Fourth Normal Form: The conditions to satisfy the Fourth Normal Form are:

1. The table should be in the BCNF.
2. The table should not have any Multi-valued Dependency.

Multi-valued dependency: If the following conditions exist in a relation(table), then the table is having a multi-valued dependency.

- First, the table should have more than three columns.
- Second, for a dependency like $A \twoheadrightarrow B$, for a single value of A multiple values of B exists.

- Third, for a relation $R(A, B, C)$, the relation between A, B and B, C should be independent of each other.

To avoid the multi-valued dependency, the table must maintain independent non-prime keys. If there is any relation between non-prime keys, then retrieval of data will become hard. To avoid the multi-valued dependency in the application, two tables are created for the user login credential details. The users will have the username, password, and their roles information. As there are three roles created in the system (explained in further sections), the user information table might have three rows of data. Because of this repeat data which is called as a multi-valued dependency, during data retrieval the system will pull additional unwanted data. One for the user credentials and another one for the roles information. They both tied together with the foreign key.

Normalisation Stages followed in the application: The process of applying Normalisation Stages is performed by a series of tests on a relation. These tests determine whether it satisfies or violates the requirements. If a test fails then the relation is decomposed into fraction relations to meet the individual normalization forms. Usually, the higher the normal form test success indicates the fewer vulnerabilities in the relation which causes the anomalies. All the first three norms are based on the functional dependencies. The below are the steps involved to implement the normalization forms in the application.

- Step 1: The data source is selected and converted into an unnormalized table. It means the table is created with multiple values or complex data structures stored with a single attribute. It is difficult to maintain data changes in this current form.
- Step 2: The table in unnormalized form is transformed to first normal form.

The complex data structures or multivalued columns are converted to columns with atomic values. This will help the table to not having repeating columns.

- Step 3: In this stage, the table is further transformed to the second normal form. The columns are categorized and it is organized in a way that each column is dependent on a primary key of the table.
- Step 4: The third normal form, non-transitive dependency will be executed in this stage. It makes sure that all of its columns in the table are not transitively depends on the primary key.



Figure 3-10: Normalization flow

The flow of the implementation of Normalization forms is shown in figure 3-10. After the implementation of the third normal form, the data may still subject to the anomalies. In that cases, the further transformations need to be performed on the tables. The next levels of normalization techniques are normally required for complex database structures. As the CARATS database is not having complex structures, the further forms are not implemented.

Microsoft SQL Server query execution: The understanding of the Microsoft SQL Server query execution and the process of retrieving, updating the tables is shown in figure 3-11 and explained below.

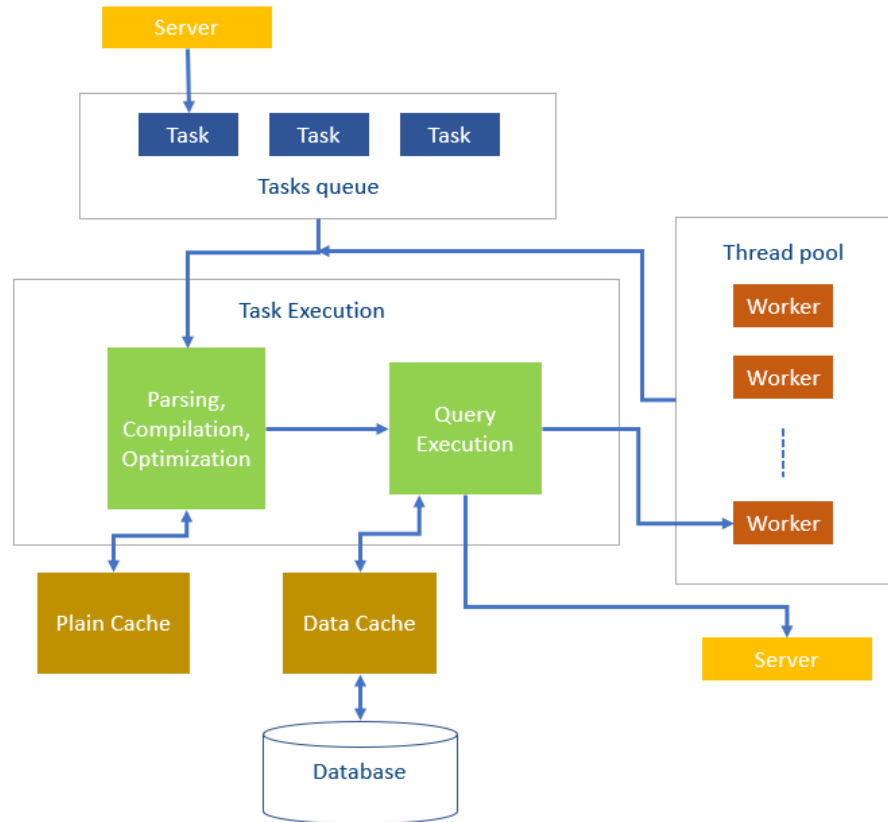


Figure 3-11: Database flow

1. The application layer requests the database either to retrieve data or to manipulate data.
2. Before assigning the request to the task execution, the system checks for the available workers. When a worker is available, it sends request and idle worker information to task execution block.

3. In task execution, the system according to the given requirement it parses, or compiles or optimizes data. The plan is stored temporarily on plan cache.
4. Once the plan is finalized to execute the query, the operator sends a request to the database for data through the buffer pool. Data is temporarily stored in Data cache.
5. After accumulating the required data, the system executes the task or query. The gathered information will then transferred to the application layer. The worker status is changed to idle and passed to the thread pool section.

The details of the blocks in the Database flow is explained below.

Request: In Microsoft SQL server DBMS, the only way the application layer can interact with the database is by sending requests through commands. The communication protocols used between the database and the application is called Tabular Data Stream (TDS). Every request will be processed or transferred through the TDS. Below are the few request forms used in the application.

1. Batch Request: The batch request is used to execute a bunch of requests. This requests will contain just T-SQL text having no parameters. The T-SQL batch may have locally declared parameters. Commands like SqlCommand.ExecuteReader(), ExecuteNonQuery(), ExecuteScalar(), and ExecuteXmlReader() are used to invoke the T-SQL batch.
2. Remote Procedure Call Request: This kind of request will have any number of parameters and a procedure identifier is used to invoke. The client or application layer will send an RPC (remote procedure call) message data stream to the DBMS. The application layer can send a numerical or text value with parameters, but it should not send mixing both RPC and parameters. Additionally, the application layer can send more than one RPC messages at one time.

3. Bulk Load Request: Like a bcp.exe utility, the Bulk load is request used for bulk insert operations. It is the only request that starts executing the task before completing the TDS protocol. It starts the process and then consumes the data stream to insert.

The syntax `sys.dm_exec_requests` can be used any time of a point to check the status of requests.

Tasks: Task runs SQL statements or stored procedures with either a single SQL statement or multiple SQL statements that run sequentially. A task will be built on the requests, but not on the type of request. For example, the entire Batch request is considered as a task, not individual statements. The individual statements will not create new tasks. Sometimes a task can be subdivided to run parallel statements. The syntax `sys.dm_os_tasks` is used to check the status of the tasks. When a request reaches the DBMS, the task is created and it will be in a pending state. At this stage, the application layer will have no idea about the request. The engine must assign a worker to start executing first.

Workers: Initially, at server start up, a certain number of servers are configured in the system and later can be increased depending upon the demand. They are the thread pool of SQL Server and only workers can execute code. Workers wait for the tasks to hit into the system. When they did each worker will take up only one task and executes it. The status of the worker will be on BUSY until the executing task is complete. For a SQL batch request, the worker executes the entire batch statement after statement. It means the execution of each statement must be completed before the next one starts. The status of workers can be seen by querying `sys.dm_os_workers`.

Parsing and Compilation: Once the worker picks up the task from the server then the first thing the system does is to understand the request. The statements in

the task will be parsed and by using the interpreted language the system prepares an abstract tree structure. The entire batch of a request or a single statement is parsed and compiled. If an error occurs at this stage, the system will terminate the request with a compilation error message. The error message will be transferred to the server to rectify the problems in the statement. The system changes the task status to complete and the worker will be assigned to the next task. When complex queries reach to the system like select with too many joins query plan will be created. Query plan eases the whole process of executing the complex statements. The complex queries are avoided in the application by using effective where clause conditions. So the query plan does not take place in this application.

Optimization: In the process of execution of SQL query, the next most important stage of the request is optimization. The optimization allows the system to choose the best data access path from all possible alternatives. Consider the system trying to access a simple query with join between two tables. There will be four possible ways to access data when each table has an additional index. The Join statements can be nested, loop, hash, and merger which increases the possibilities more. And this number of possibilities increase exponentially as the complexity of the query increases. As the SQL server charges according to the usage, the complex query increases possibilities which increases the cost. To avoid overuse, the queries are optimally constructed by simplifying as much as possible in the system. The CPU consumption and Memory usage are also reduced by simplifying the complexity of queries. After exploring all these alternatives once a query plan is created, it is cached for future reuse. Future alike requests can skip the optimization phase and can find in SQL Server internal cache.

Execution: Once the query is optimized and query plan selected, the request starts executing. The query plan is translated into execution tree. Each node is called as

an operator. All operators work on three methods: `open()`, `next()`, and `close()`. The operators' loop starts with `open()`, continues the steps with `next()`, and finally closes by calling `close()`. Queries that use parallelism use a special operator called an Exchange operator which helps intermediate operations like filtering data, sorting rows and so on. These exchange operations are implemented at the client level to reduce the workload on the database. The JavaScript libraries have been used in the client section. This execution process not only applies to queries, but also insert, delete, and update modifications. The same execution plans are used for the modifications: `open()` for start, `next()` for steps, and `close()` for closing. These operators are designed to work for both regular loops and nested loops. Few operators consume all data before starting the process. This application has avoided these operators by moving a few operations to the client side like sorting.

Results: Once all operators in the execution complete their process then the execution status will be changed to completed. The results will return back to the client program. The last operator means the top one transfers data into network buffers. Later the data will be sent to the client. The network flow control protocols are used in the system. Suspension of data will happen when the client is not actively consuming the result. During this process, the network flow control will terminate the session. The output parameters of the query can only be checked after the results are consumed on the client side. [66, 67]

3.9 Cloud Deployment

This section discusses the cloud deployment of the proposed CARATS architecture. The client and server applications that come together to form CARATS are deployed on a single Amazon Web Services (AWS) Elastic Beanstalk instance. The

CARATS uses the Apache Tomcat server to serve the client application. All the components in the CARATS architecture reside on Amazon web services. As shown in Figure 3-12, the overview of the AWS components are described below:

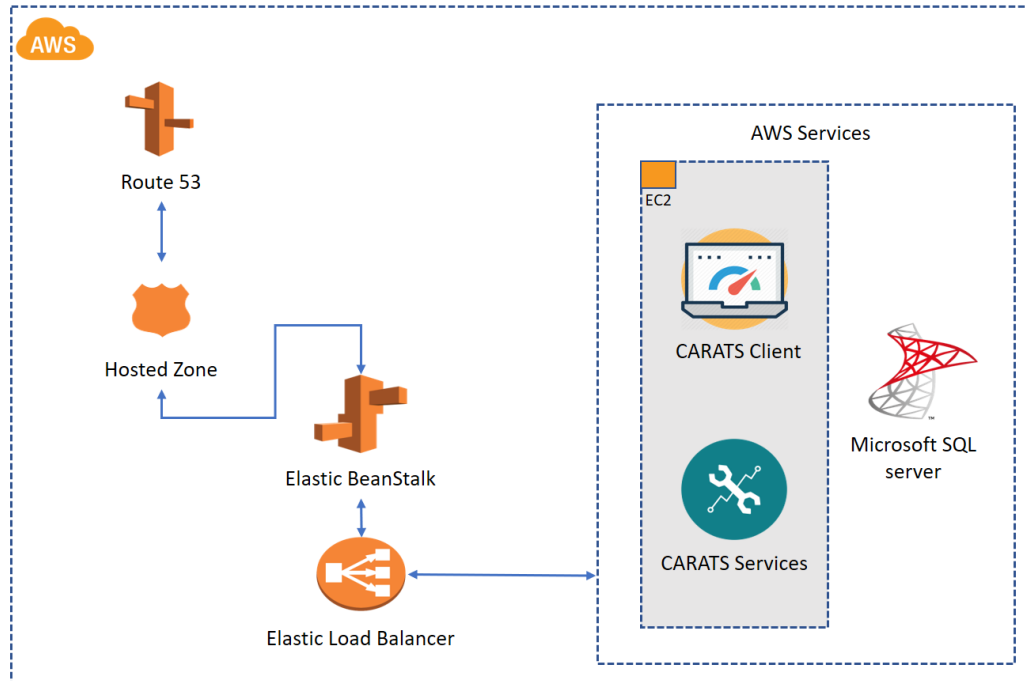


Figure 3-12: Cloud Component layout

Route 53 is a Domain Name System (DNS) web service that helps to route once the user requests for a browser Uniform Resource Locator (URL) to hosted zone. [70]

Hosted Zone contains information about how traffic of the specified domain would be routed on the internet.

Elastic Load Balancing is used based on configured algorithm routes incoming traffic to multiple targets. [71]

Application Load Balancer is used for HTTP port forwarding. For distributed architectures, it also provides advanced request routing.

Elastic IP address associated with an EC2 instance for reconfiguration. This helps when the EC2 instance is resized or changed. [72]

Availability Zones indicates geographically separated areas. CARATS uses availability zone in US East region.

Security Group restricts the access to the hosted components on AWS. All user HTTP and HTTPS requests are allowed by CARATS application security group.

EC2 corresponds to an AMI instance. It hosts the complete application architecture. CARATS uses EC2 with Windows Server with Microsoft SQL server.

MS SQL Data Store corresponds to MS SQL Server installation on EC2.

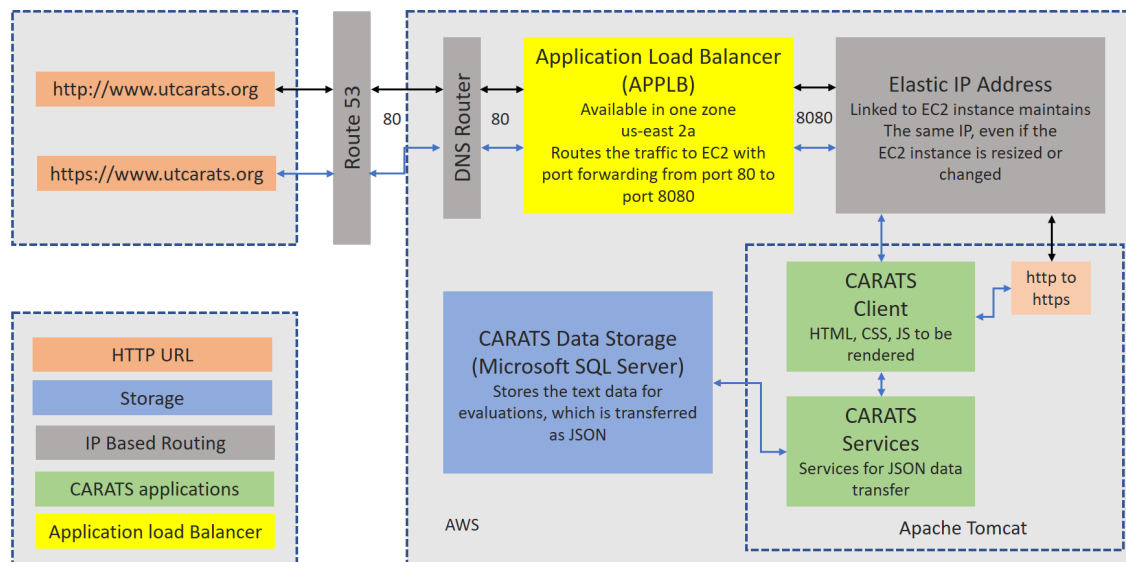


Figure 3-13: AWS architecture flow

Figure 3-13 demonstrates the complete flow of an HTTP/HTTPS request. The possible request flows are explained by two cases which are described below as follows:

CASE 1 (HTTP Request, shown in black): When the user requests the web URL `http://www.utcarats.org` from the browser (computer, tab or mobile), the request forwarded to Route53. The Route53 navigates it to `utcarats.org` hosted zone where the request is intercepted by DNS router. The DNS router then forwards it to Application Load Balancer. The load balancer based on defined algorithms either forwards or redirects to multiple servers. It has too many purposes, but it is used only forwarding or redirecting in the application. According to the defined algorithm, the load balancer redirects the request from port 80 to 8080 of the Elastic IP Address. The EC2 instance is allocated with elastic IP, on which the CARATS suite of applications is deployed. The Apache Tomcat is configured on EC2 instance to run at port 8080. The CARATS two-tier architecture application is deployed on this Tomcat web server. Once the request is reached to web container on EC2, it will be handled by the application. To secure the data the HTTP request is rerouted to HTTPS request. Thus, the request is rerouted back to the browser as an HTTPS request. The request is denoted by `https://www.utcarats.org`. The flow changes after it reach the Elastic IP address again. The incremental flow for HTTPS request is described in CASE 2.

CASE 2 (HTTPS Request, shown in blue): In this case, once the request reaches the Elastic IP Address bypasses the Protocol Routing. Therefore, it will be forwarded to the CARATS client. To retrieve or send data, the application client is connected to the services component. The services component then forms response in JSON and sends back to the user through the client.

The following are the detailed explanation of the components used for the deployment of the CARATS in Amazon web services.

1. Route 53: Route 53 is a DNS, web service that follows a series of steps to turn the human-readable web address into a machine-readable IP address. In this case the web url utcarats.org into a physical machine readable address . The recursive server returns the records from the IP address to the browser. The browser then opens a connection to the web server and receives the website.
2. Hosted Zone: Hosted Zone holds the data about the traffic routed for the CARATS domain.
3. Elastic BeanStalk: Based on a configured algorithm the Elastic Load Balancing routes the incoming traffic to multiple targets and it also handles the application details of capacity provisioning, load balancing, scaling, and application health monitoring.
4. Application Load Balancer: Application load balancer is the point of contact for the clients or users. It distributes incoming traffic to the multiple availability zones. CARATS application load balancer deployed only in Ohio zone as the majority users of the website resided in Toledo. It transfers all the requests from clients or users to the Ohio zone using the configured protocol and port number. There are no rules or conditions designed for the CARATS application load balancer for the reason that the single availability zone. The CARATS application is deployed in us-east-2a zone.
5. Elastic Load Balancer: The Elastic IP address is a static version 4 internet protocol (IPV4) uses packet switched networks. It is a connectionless protocol that does not guarantee delivery nor proper sequencing in delivery. We can mask the failure of the connection of an instance by continues remapping the address. The AWS does not support the IPV6.
6. Security Groups: Security group operates as a virtual firewall for instance to

control traffic both inbound and outbound. The AWS allows five different security groups to assign for any instance. Basic security groups created for the CARATS instance to make it available to all users. The CARATS instance security groups allow HTTP and HTTPS requests from all clients or users.

7. EC2 instance: Amazon Elastic Compute Cloud (EC2) provides virtual computing environment in AWS cloud. The virtual cloud computing is the delivery of computing power, storage, and other IT services via the internet. Using AWS EC2 virtual computing environment eliminates the upfront investment in hardware infrastructure. It allows faster development and deployment. The AWS can provide securities and configurations for the multiple numbers of servers for the cross independent applications. The EC2 allows various configurations like CPU, storage capacity, memory and network capacity.
8. Microsoft SQL Server: AWS offers the Microsoft SQL Server with a number of versions and editions. There are two possible ways Amazon Elastic Compute Cloud (Amazon EC2) or Amazon Relational Database Service (Amazon RDS) to deploy Microsoft SQL Server. The CARATS website database system created on Amazon RDS. The AWS supporting all the Microsoft SQL Server implications. The database created with db.t2.micro instance class, SSD storage type, general license public model, 20Gib capacity and mysql-5.6 version. The AWS offering security groups to establish the connection for development and client-server configurations. Five security groups are created for CARATS database, in which four of them for development purpose and one to establish connection with the client application. To save the CARATS user data the database backup enabled for every seven days. [73]

Chapter 4

Centralized Anesthesia Resident Achievement Tracking System

The tool is written in Java with back-end Microsoft PL/SQL and front end services Angular, JavaScript and Bootstrap. The project runs on Tomcat web server which is accessible from any web browser. It is deployed to supplement the existing paper evaluations of residents at a medical college in Ohio.

Creating data sharing networks requires a balance of privacy. In order to maintain the confidentiality of the residents while still allowing the data to be mined for helpful observations, we created three roles in CARATS; evaluators, residents, and grand round's audience. The evaluators, usually medical instructors, are granted access to simulation evaluations, grand rounds, daily feedback, and case log pages. The instructors are also provided access to the admin page, where they can create a new resident or instructor account. The instructors can submit forms and observe residents' scores in a table format. They are only restricted from accessing the residents' dashboard. The residents' account is provided access to the dashboard and grand rounds. The evaluation results appeared on the dashboard in a graphical representation. To make the grand round evaluations available to anyone attending a grand rounds presentation, the global login credential account is created. The global credential grants

access to the grand rounds evaluations page but is restricted from other pages. The detailed information and how to operate the application are explained in following sections.

4.1 Application Overview

The first time the application is requested using a web URL, the Tomcat web server loads the necessary components which initiate all the modules in the client-server architecture. The Spring MVC home page controller renders the welcome page as shown in figure 4-1. The homepage is decorated with the slideshow of UTMC pictures, welcome note, important links, health science contact information, tweets block to share information with the residents, and the collaborators to the project. The pictures of the collaborators are expandable pop-up's for their educational background and research achievements shown as figure 4-2. Blackboard, contact information, faculty and administration staff re-routing links are provided under the important links section. All these details can be seen in figure 4-1.

As no user session exists while loading the welcome page, the navigation bar will be loaded on top of the page with an evaluation dropdown on the left and a login link on the extreme right-hand side. When the user clicks the login link, the CARATS client-server application navigates the user to a secured Login page as shown in Figure 4-3.

When the user provides the credentials “Username” and “Password”, and clicks the login button, the service request will be passed to the dispatcher servlet and then to the controller. The controller validates the credentials and returns the response. Depending on the response from the controller, the user is navigated to different pages. Once the application received the success response, it saves the user session information like authentication ID in the browser. Additionally, it hides the Login

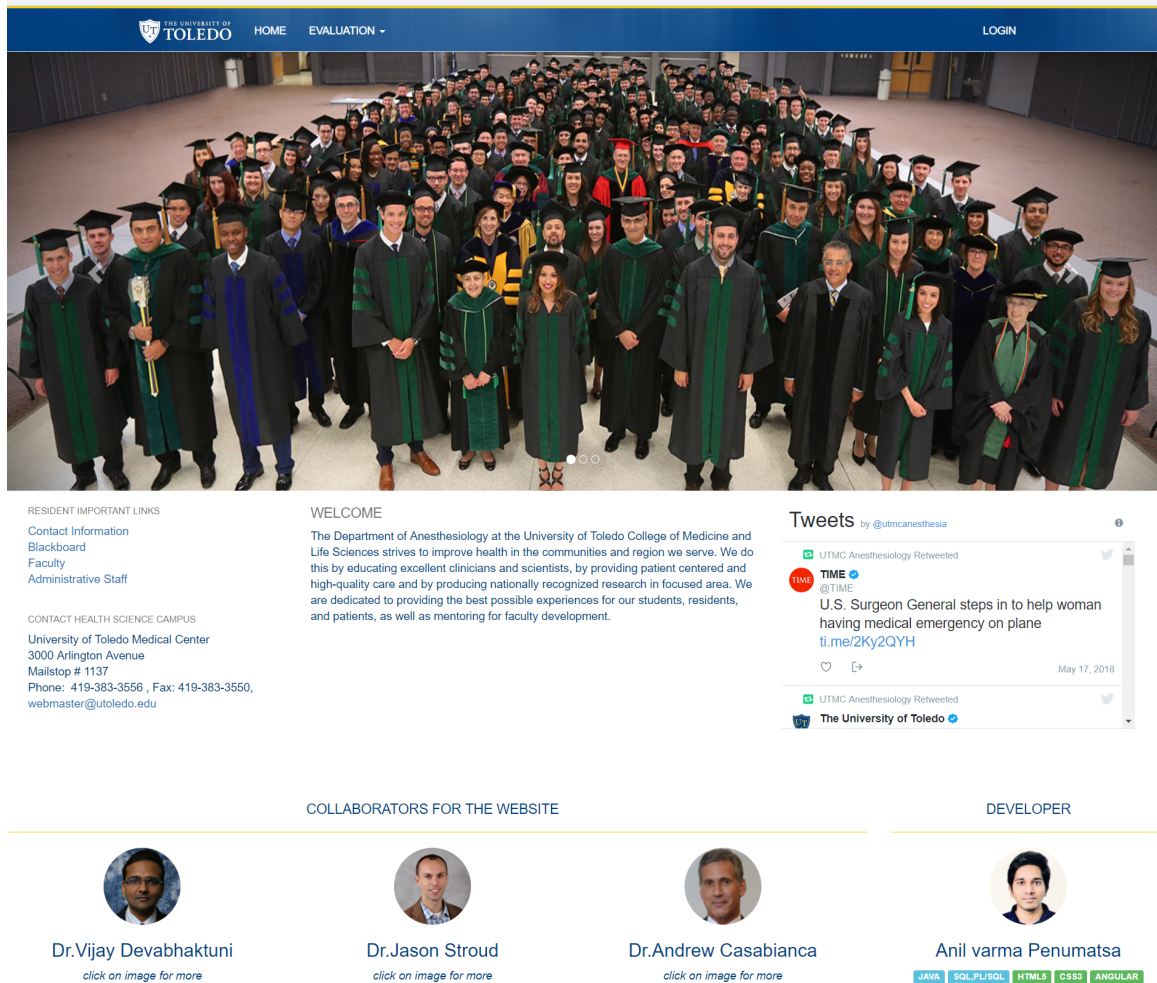


Figure 4-1: CARATS Home page

More About Dr. Vijay Devabhaktuni



Dr. Vijay Devabhaktuni USA

BE (EEE) & MSc (Physics) From BITS Pilani
PhD (Electronics) From Carleton University
Canada Prestige Professor Of Anesthesiology Dept
P.E. Of APEGA Canada, Sr. Member Of IEEE

Vijay is a Professor in the EECS Department of the College of Engineering at The University of Toledo. During 2012-2016, he served as the Director of the College of Engineering for Interdisciplinary Research Initiatives. Since January of 2017, he is the Senior Director of Research Development & Innovation in the Anesthesiology Department of the College of Medicine and Life Sciences. His R&D interests include applied electromagnetics, biomedical applications, computer aided design (CAD), device modeling, human machine teaming and human effectiveness, infrastructure monitoring, machine learning (eg. ANN), medical education exploiting simulation, optimization methods, RF/microwave design, virtual reality, wireless networking, and other emerging themes. Dr. Devabhaktuni secured funding close to \$5M, authored 230 technical papers and advising 14 grad students. To date, he graduated 50 MS and PhD thesis students. Dr. Papadimos and Vijay founded UTEMRIC framework.

Close

More About Dr. Jason Stroud



Dr. Jason Stroud USA

Assistant Professor Of Anesthesiology

Dr. Jason Stroud is an assistant professor of anesthesiology at the University of Toledo Medical Center. He completed his medical school and residency training at the University of Toledo and completed fellowship training in pediatric anesthesia at The University of Michigan. Dr. Stroud's research interests include the use of high-fidelity simulation in anesthesia training, novel approaches to anesthesia education, and human factors in anesthesia care. In addition to research in simulation, Dr. Stroud is the simulation curriculum director for the anesthesia residency program at The University of Toledo and is involved in advocacy for young physicians at the state and national level.

Close

Figure 4-2: CARATS collaborators expansion pop up

The image shows a login page for the University of Toledo's CARATS (Centralized Anesthesia Residency Achievement Tracking Software). The page is overlaid on a background image of a tall, ornate stone clock tower. The login form is white with a purple header. It contains two input fields for 'user name' and 'password', each with a small icon to its right. Below these fields is a blue 'LOGIN' button.

University Of Toledo
Centralized Anesthesia Residency
Achievement Tracking Software

user name

password

LOGIN

Figure 4-3: CARATS Login page

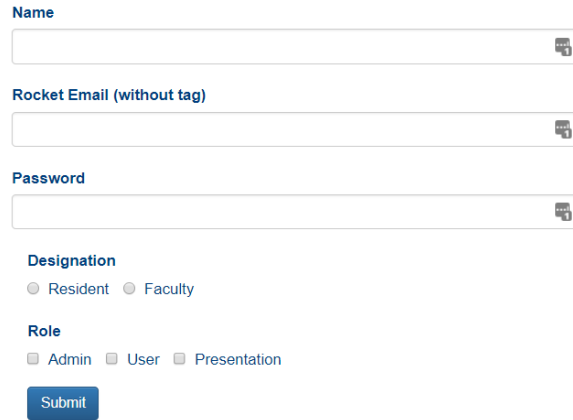
dialog, and to indicate a successful login it loads up the application navbar with customized links. At this point of time, the login link in the navigation bar is replaced with customized user profile drop down with few user-related pages. The user logout session link is visible by expanding the user profile drop-down. The user can log out at any point of time from the application. Logging out of the application terminates the current user session, thereby, user session stored in the browser will not be used for authentication purpose.

4.1.1 User Roles and Security

Among the JDBC, In Memory, and LDAP authentication services, the application is configured with the JDBC authentication. The JDBC authentication allows the application to validate credentials with the database. The application follows a list of steps to authenticate the user. While authenticating, the application pulls context information, like a list of roles for the user, after the verification of the password for the username. The application establishes the security context for the user. Both username and password are combined into UsernamePasswordAuthenticationToken. When the user requests a page protected by an access control mechanism, checks the required permissions for re-routing against the established security context information. The servlet filter is configured in the application to protect the application URLs' and help them in redirecting the user to the login form when roles are not matched. Servlet filter ensures that the authentication of the user before re-navigating the URL request. In addition, the servlet filter follows the web security services which is HTTP basic authentication configured in the application. The http element, a web-related namespace functionality, is responsible for creating FilterChainProxy to secure all URL's. Passwords are encoded with password-encoder element using a secure hashing algorithm. The HTTP authentication is introduced to implement a request level authentication in the CARATS application.

For further strengthening the security of the application, the Cross Site Request Forgery (CSRF) tokens are implemented. In some cases, there is a chance that the active user data stored in cookies unintentionally transferred to hackers. This happens when the user actively participating in the application and the malicious websites at the same time. So the malicious website takes the advantage of the weakly protected website to gather private data and misuses it. In order to protect the cross transformation of data, the CSRF tokens are introduced into the system. Therefore, every GET and POST requests are equipped with CSRF token in the header.

To protect the privacy of the data, certain roles are created in the system. We classified the roles depends on the category of user requirements. According to the requirements, we designed three roles, namely Role_admin, Role_user, and Role_pres. The users with Role_admin access can submit forms on residents performance in simulation evaluation, presentation evaluation, and daily feedback and they can also upload the case log data into the system. The users with admin access can also create a new user account by accessing the user administration page. The user administration page is shown in figure 4-4. The admin user is required to fill the fields in the page username, password, rocket email id without the tag (used for identifying the residents in the research), designation of the new user, and kind of role the new user eligible or needed to create a new user. The selection of the role depends on the kind of the user. There are three kinds of the users, namely resident, lecturer, and presentation audience implemented on the system. The admin users were not given access to the resident's dashboard but they can see the summarized data in other pages explained in further sections. The Role_user users are not authorized to access to the simulation evaluation, daily feedback, and case log upload pages. They are authorized or given access to presentation evaluations and their individual dashboards. To protect the privacy of residents scores, user with Role_user access can only navigate to their



The form is titled 'Add Users' and contains several input fields and radio buttons. It is organized into sections: 'Name', 'Rocket Email (without tag)', 'Password', 'Designation', and 'Role'. Each section has a corresponding input field or radio buttons. The 'Name' field is a text input with a small icon on the right. The 'Rocket Email (without tag)' field is a text input with a small icon on the right. The 'Password' field is a text input with a small icon on the right. The 'Designation' section has two radio buttons: 'Resident' and 'Faculty'. The 'Role' section has three checkboxes: 'Admin', 'User', and 'Presentation'. A 'Submit' button is located at the bottom of the form.

Name

Rocket Email (without tag)

Password

Designation

☐ Resident ☐ Faculty

Role

☐ Admin ☐ User ☐ Presentation

Submit

Figure 4-4: User administration page

own individual dashboards. The dashboards do not have any comparisons or other residents' performance metrics. The Role-pres is created for certain individuals who are involved in the presentation evaluations. The presentation role is introduced into the system to make the presentation evaluations available to the audience. The admin user can create global credentials before the presentation to obtain the feedback from all the audience. The complete role structure is shown in figure 4-5.

To implement the roles structure in the application, authentication is configured in two different stages. One is at HTML page level and the other one at the spring configuration level. The syntax's `hasAnyRole` and `hasRole` are used to execute the authorization of pages. The `hasAnyRole` is used for the pages needed for multiple access and `hasRole` for the single access pages. When the user is not assigned with the role to access the page, the application will re-route to the user denied page with the home page navigation link. The access denied page is shown in figure 4-6.

The navigation bars are customized according to the type of users. The admin account users can see the simulation and presentation evaluations under the evalua-

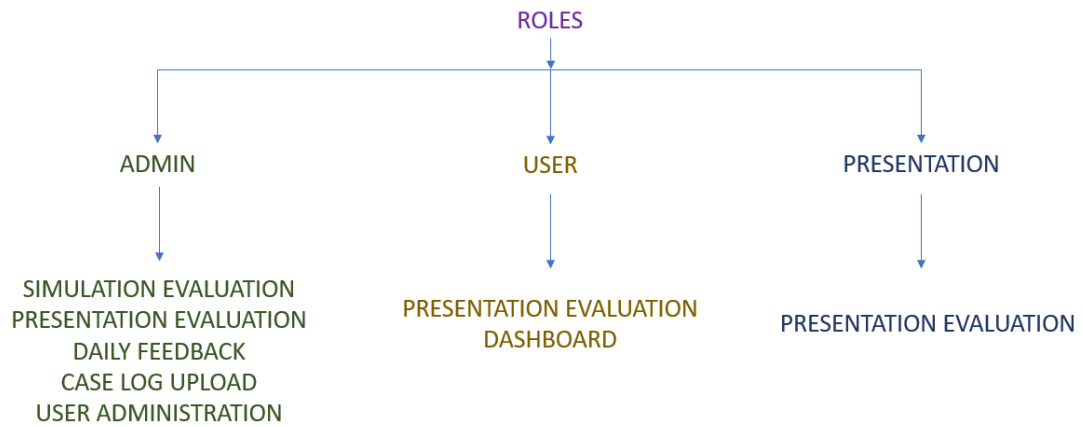


Figure 4-5: Roles structure

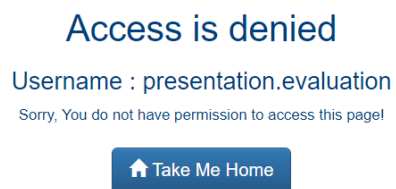


Figure 4-6: Access denied page

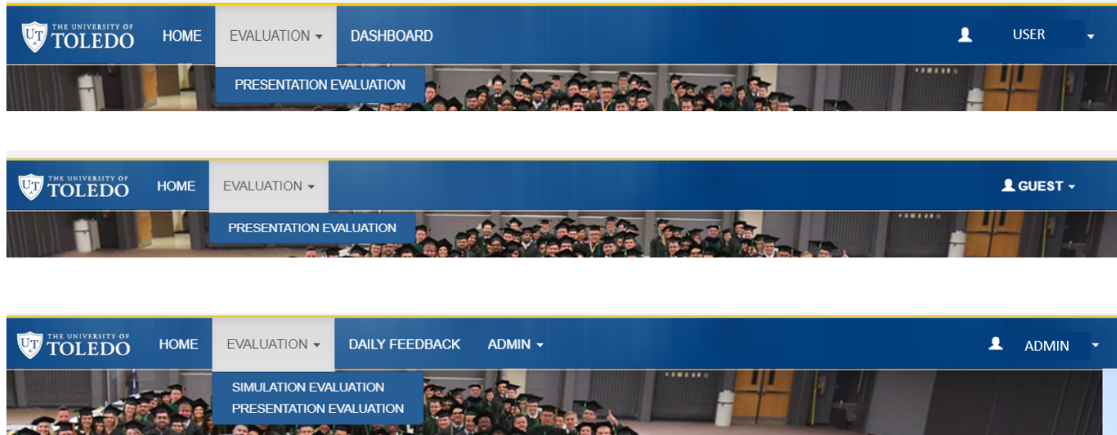


Figure 4-7: Navigation bar for different users

tion drop-down, user administration and case log upload under the admin drop-down, daily feedback link and to the right-hand side a customized profile drop-down. The JSTL libraries are practiced in the application to restrict the navigation bar with unwanted links. The JSTL conditional tags used for executing the user role conditions and it displays the content only if the expressions come true. The presentation users' navigation bar is displayed with presentation evaluation link under evaluations drop-down. The Role_user navigation bar has the same evaluation drop-down with presentation evaluation and the dashboard links. The user profile drop-down is common for every user in the application. The three different navigation bars are shown in figure 4-7 .

4.1.2 Password protection

Under user profile drop-down in the navigation bar, the update password page is created to change the customized password provided by the admin user. The admin account users will create a password for the first time as discussed in the previous section. After receiving the account details from the admin, the new user must

User Name

Guest

Rocket ID

presentation.evaluation

password

Submit

Password must contain the following:

- A lowercase letter
- A capital (uppercase) letter
- A number
- Minimum 8 characters

Figure 4-8: Password change page

change the password. The update password page is developed with pre-populated fields username and rocket ID which cannot be changed during the process. The application also does not allow the users to change the username or the rocket ID created by the admin. Once the account is created, even the admin is not given access to change the username or rocket ID. The only option is to delete the current account and create a new one for that user. Several rules are created to further strengthen the password. They will be highlighted in red color under the submit button when the user attempts to change the password field. The rules are one lower case letter, one upper case letter, one number, and a minimum password length of 8 characters. Once the password fulfills the rules one by one, they turn green. The submit button will work only if the password fulfills all conditions. The update password page is shown in figure 4-8.

A few spring MVC offered and few additional attributes are created and stored

User name	Enabled	Rocket Email	Action
Guest	1	presentation.evaluation	Delete

Figure 4-9: User account details for admin

in the database to implement the login effectively. Among all these attributes, only a couple of them will be displayed in the users summary page. Once the admin creates the account, the page will be navigated or re-routed to the summary page to cross check the details. The **username**, **rocket id**, and **enabled** are the three attributes displayed in tabular format. The **enabled** attribute describes the status of the account where one denotes to an active account and zero to an inactive account. The **enabled** attribute is developed to add an additional functionality in future to the application. It is created for the purpose to disable and re-enable an account for a certain period of time during the course. This feature helps when the user chooses to take a break during their education. If the account details are misspelled or unnecessarily created by mistake, the added delete feature under the action table header helps the admin to undo the account. After the admin click on the delete button, a delete confirmation dialog will prompt on the screen with the username to avoid unexpected button hit. The user summary page is shown in figure 4-9.

4.1.3 User profile

The motive to develop a user profile page is to decorate the dashboard with personal information. The only user information that has been provided to the application when admin create a new user is the name of the individual. The profile page

allows the application to gather more information about the resident or the lecturers. The first and last name, personal email (not university email), contact number, gender, address (three fields), and date of joining are the fields presented to the users to submit as shown in figure 4-10. There are no mandatory fields on this page, the user can submit with any number of entries. Additionally, this page is constructed with the pre-populated theme. The fields will be occupied with the previous or last submitted data. Let's suppose the user submitted the form with the first and last name on one of their visits. On their next visit to the page, they can see the fields filled with the last submitted data. All these fields are presented in such a way that they can be edited with any format. This page will also help in future work like dashboard as an abstract to the professional resume. In that case, it helps to fill up the personal information and brings the completeness to the resume.

4.2 Inputs

The data goes into the system through four individual forms: **simulation evaluation**, **grand rounds evaluation**, **daily feedback**, and **case log upload**. Each form is designed based on the type of evaluation. The simulation evaluation is designed to submit the simulation results and the page access is given only to the admin account. Most of the admin accounts created in the application are lecturers or medical examiners. The **grand rounds evaluation** page intention is to transfer the feedback generated or submitted by the group of audience in grand rounds. This page access is given to all account holders irrespective of their roles. The **daily feedback** is designed similar to the simulation evaluations but with different criteria. The **case log upload** is designed completely different compared to the other forms. Admin account holder will upload an excel spreadsheet with residents performance into the system. All the forms are developed with an aim to automate the process

First Name

Last Name

Email

Contact Number

Sex ☐ Male ☐ Female

Address1

Address2

Address3

Date of join

Figure 4-10: User profile details

of delivering the residents' feedback and to provide easy access to the forms for the instructors. The application allows the instructors to easily submit evaluations and makes the data immediately available to the residents. The admin account users are provided with access to submit all four forms in the application. Residents and grand rounds audience can only submit the grand rounds. Most of the evaluations' fields are given with five options ranging from one bad to five exceptional. The following subsections introduce the detailed description of the application input forms:

1. Simulation evaluation
2. Grand rounds
3. Case log
4. Daily feedback

4.2.1 Simulation Evaluation

A simulation is an instrumental design to replicate the real-time patient experiences with guided artificial practice [75]. In the early stages of clinical education, the residents used to practice on patient bodies. As patients came to realize that residents were practicing, they often began getting nervous [76]. Since 1960, medical schools have been using simulation centers to train residents in technical and non-technical skills. The Simulator One, designed at that time is a computer controlled mannequin patient simulator which is notable as one of the first Multi Pilot Simulator (MPS) [77]. The Comprehensive Anesthesia Simulation Environment (CASE) was the first commercial system introduced and utilized [78]. In simulation centers, most simulations involve mannequins and apparatuses to train the residents. The evaluation of these simulations is of the utmost important factor in medical education because it is

Create New Simulation Form



Simulation Name

Add

Close

Figure 4-11: Add simulation type form

now included as part of board certification. Therefore, it is important for residents to receive responsive and effective feedback on simulation sessions during their training.

The simulation evaluation gathers the residents' and evaluators' basic information to relate the data, and various other fields to determine the residents' performance. The residents' name, rank, and the type of simulation fields are used in the application to relate the data to the resident. The evaluator name which is, by default, the logged-in username and the date of simulation submitted are gathered for instructors analysis. This information helps to examine the residents' performance under them at any point of time. The simulation evaluations explore various performance fields such as the resident's medical knowledge, technical skills, teamwork, leadership, and professionalism. Each field is presented with five mutually exclusive options, the lecturers can rate the residents' performance from poor to excellent. The individual comments section is placed next to each field for instructors to provide suggestions or committed mistakes or appreciation to the residents. The additional comment section is also provided to the instructors to submit the overall performance observations.

To avoid the misspelling of words, the residents' name, their rank, and type of simulation are created as the clickable drop-down menu. It allows the instructors

Select Resident

Ryan Birdsall ▼

Resident Rank

Open this rank menu ▼

Simulation Name +

Testing ▼

Date Of Simulation

06/11/2018

Medical Knowledge * ☐ Poor ☐ Fair ☐ Satisfactory ☐ Good ☐ Excellent

comments

Technical Skills * ☐ Poor ☐ Fair ☐ Satisfactory ☐ Good ☐ Excellent

comments

Team Work * ☐ Poor ☐ Fair ☐ Satisfactory ☐ Good ☐ Excellent

comments

Leadership * ☐ Poor ☐ Fair ☐ Satisfactory ☐ Good ☐ Excellent

comments

Professionalism * ☐ Poor ☐ Fair ☐ Satisfactory ☐ Good ☐ Excellent

comments

Additional Comments


comments 

Figure 4-12: Simulation evaluation form

to choose one value from the pre-defined list comes from the database. There is a plus symbol placed next to the simulation name field to add a new simulation type. If the evaluator is not able to find the name of the simulation, they can add it by using this feature. The **add simulation form** is shown in figure 4-11. The resident's performance and other mandatory fields are marked with a star symbol. The comment sections in the simulation form are optional fields. If the evaluator finds it is important to convey a mistake or appreciation, they will use the comment fields. After submission, the results are posted on the resident's dashboard and the evaluators' simulation results page. The simulation evaluation submission form is shown in figure 4-12.

4.2.2 Grand Rounds

The purpose of the grand rounds presentations is to present the medical problems of a patient and the treatment to an audience of doctors, medical students and residents, and it is one of the principal methods of instruction in teaching hospitals [79]. It helps doctors to improve their speaking skills so they can communicate clearly with patients and with other doctors. Additionally, it encourages the residents to review a medical topic in-depth, and also allows them to practice their verbal and non-verbal communication skills. Grand rounds present medical knowledge from past and present that helps doctors, residents, and medical students to stay up with the latest information. These activities keep them competent and, moreover, they promote a collegial atmosphere [80].

The grand rounds presentation feedback has sections similar to simulation evaluation. It gathers residents' information through the presentation title and the name of the presenter. The users submitting the feedback also need to fill the date of presentation and their rank. It helps the application to group and aggregate data before presenting it to the residents. The **rank** field help to prioritize the feedback based on

the audience educational status like the faculty feedback first, then anesthetist and so on. The **presenter name** and **evaluator rank** are clickable dropdowns with a pre-defined list. There are four options coded to the rank dropdown: student, resident, anesthetist, and faculty. The **presenter name** pre-defined list comes from the database.

The residents' presentation skills are evaluated based on organization, relevance of topic, the inclusion of appropriate visuals, effectiveness delivery, and knowledge shared throughout the presentation. As discussed before in the roles section, every user in the application is provided with access to this page. The audience can rate these performance fields similar to simulation evaluation from strongly disagree to strongly agree with five mutually exclusive radio buttons. The presentation evaluation form is shown in figure 4-13.

4.2.3 Case Log

The intent of the ACGME Resident Case Log system is to record the clinical experience of the residents [81]. The recorded experience enables residents to check the progress of their clinical experience. The ACGME logs the number of cases the resident performs in cardiopulmonary bypass; cardiopulmonary without bypass; intrathoracic non cardiac; intracerebral endovascular; intracerebral nonvascular open; intracerebral vascular open; vascular; vaginal delivery; ceserias section; complex life-threatening pathology; spinal; epidural; nerve blocks; children under three months, under three years, and under 12 years; and initial pain consultation.

To maintain a record of these cases, we developed a page to upload an excel spreadsheet file in the application. The access to the case log upload page is given only to the admin users. To partition the data from excel, the system uses the inbuilt java and spring libraries. We created a configuration class for the batch job, and it contains spring bean to describe the flow of it. We also configured an itemreader bean

Grand Rounds Presentation Feedback

Presentation Title

Presenter

Ryan Birdsall

Presentation Date

06/11/2018

Your Rank

Open this rank menu

Was Organised *

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

comments

Increased My Medical Knowledge *

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

comments

Covered Relevant Topic *

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

comments

Included Appropriate Visuals *

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

comments

Was Delivered Effectively *

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

comments

Overall Rating *

Poor

Fair

Satisfactory

Good

Excellent

Overall Rating Comments

comments

Figure 4-13: Grand rounds form



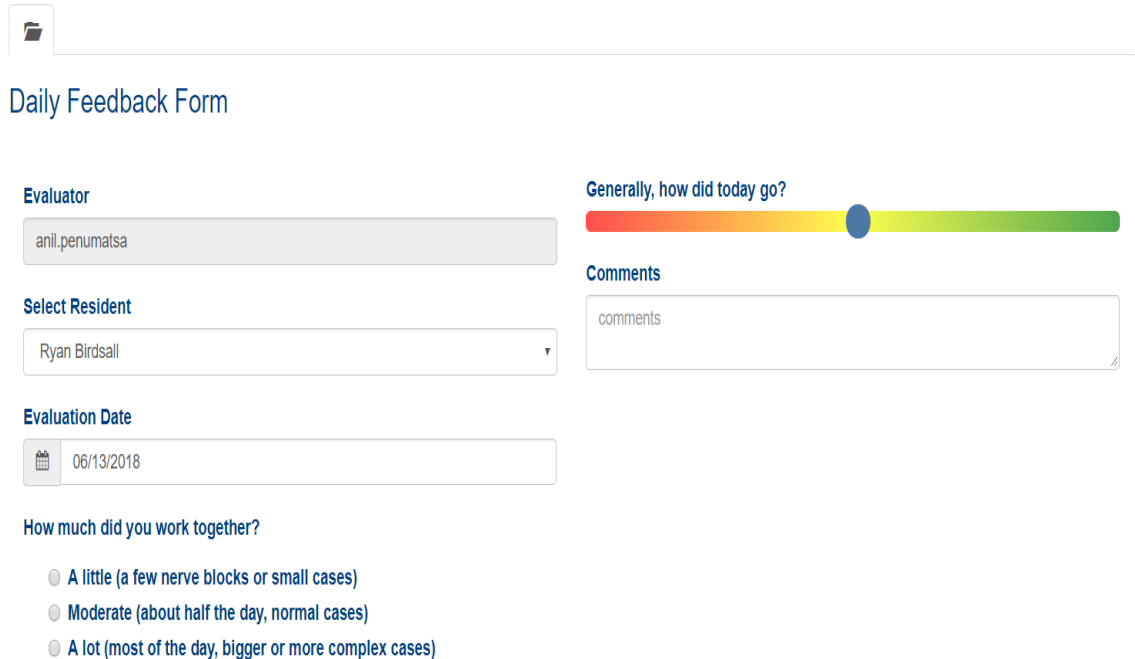
Figure 4-14: Case log file upload

to ensure the method returns residents cases object. The application will ensure that the created job will ignore the header section of the file. The path is configured in the system which is not discussed here to secure it. Since the file has a header, we mapped the rows into X object by using a BeanWrapperRowMapper X class. The system will use a loop structure to gather all data from the file. When it finds the empty cells, it stops processing the file. The created objects are further transferred and stored in the database through DAO and DAOIMPL sections. The file will also have additional residents' information to map data with users. The department assigns cases to the residents in rotation and updates the case logs. Case logs are considered part of the basis for graduation. Figure 4-14 shows an image of a case log file upload.

The case log file is uploaded to the system every month. The results are displayed to residents' dashboard and the customized case log results page to the admins.

4.2.4 Daily Feedback

The daily feedback form gathers the residents' performance from the medical instructors. The residents' information is collected through the residents' drop-down that comes with a pre-defined list from CARATS database. The evaluator name field will take the logged in username by default. This field is not accessible to the admin users to alter or modify. The date of evaluation is gathered to display data effectively with specific information. When compared with other pages in the application, the residents' performance information is taken on this page is developed in a different format. One of the fields (Generally, how did today go) is a sliding bar of one to one-



The image shows a web-based 'Daily Feedback Form'. At the top left is a small icon of a folder with a document. The title 'Daily Feedback Form' is in a blue font. The form is divided into several sections. The 'Evaluator' section has a text input field containing 'anil.penumatsa'. The 'Select Resident' section has a dropdown menu with 'Ryan Birdsall' selected. The 'Evaluation Date' section has a date picker showing '06/13/2018'. To the right of these is a horizontal sliding bar with a red-to-green gradient, labeled 'Generally, how did today go?'. Below the bar is a 'Comments' text area with the placeholder text 'comments'. At the bottom, there is a section titled 'How much did you work together?' with three radio button options: 'A little (a few nerve blocks or small cases)', 'Moderate (about half the day, normal cases)', and 'A lot (most of the day, bigger or more complex cases)'.

Figure 4-15: Daily feedback form

hundred. The sliding bar is developed with a gradient increase in colors red, yellow, and green. The bar starts with red and turns into yellow and from then yellow to green. The admins select position among the bar which signifies how they felt the resident performed, and the selected score is not displayed anywhere in the page. To further strengthen the score metrics, we implemented mutually exclusive radio buttons with options of how much time was spent with the resident that day. They are given with three options: 1) **a little** means a few nerve blocks or small cases, 2) **moderate** for normal cases or half of the day, and 3) **a lot** for most of the day or more complex cases. The comments section is provided for appreciation or remarks and it is an optional field. We combine the work together and sliding bar fields after submission of form. The medical instructors submit the form after examining the resident's work on cases. The main intention of this form is to evaluate the residents' performance on cases. The daily feedback template is shown in figure 4-15.

Figure 4-16: Sample results page navigation link

4.3 Outputs

There are two kinds of data display pages in the application. They are categorized on the basis of the access to those pages and the information on it. Four pages display residents' information to the admins' account (admin pages) and one page to residents' account (residents' dashboard).

1. Admin pages
2. Residents' dashboard

4.3.1 Admin pages

To distinguish the data in the application, four individual result pages are created. The re-routing for these pages is not provided in the navbar. Instead, they are presented in the same evaluation pages. For example, as shown in figure 4-16 the re-routing link for the presentation evaluation highlighted on the top right-hand side as the summary. The detailed description is provided in the following section.

4.3.1.1 Summary page details

Once the admin user request for the evaluations results, the HTTP request is forwarded to the controller. The controller then pulls the data and transfers to HTML pages. The users to all the summary pages in the application are admin account holders. They are given access to observe the data and restricted to alter or modify data in these pages. The data is displayed in table format for all pages in

the application. The table is equipped with some features to allow further analysis. The table's data is stored in static program storage as part of an initialization phase or even stored in hardware in application-specific platforms. The first feature is the search or filter the application uses the JavaScript libraries component to do the operation on the table data. During the search or filter operation, the system loops through each row to match the input field value against a list of valid items in the table. The table is formatted in a way that it hides the content that does not match the search. These search methods are implemented in the application using smart searching and they are case-insensitive. The smart searching examples are described below.

1. Match words out of order: If the user searches for the first and last name it will match a row containing the first and last name, they appear in the table regardless of the order or position.
2. Partial word matching: These tables provide immediate feedback to the user means on-the-fly filtering. A small portion of words can be matched in the result set like presen will match presentation.
3. Preserved text: The tables are added with the ability to search for an exact phrase. This can be done by enclosing the search text in double quotes. For example "presentation evaluation" matches text contains the phrase presentation evaluation.

The second feature is the column visibility, this option controls the visibility of one or more columns in a table. When the user clicks on the column visibility button, a popup with column names will come out on the screen. The visibility options are different for every table as they are equipped with different evaluation results. The expanded popup will show the buttons of all the table's columns or subsets of those

Rocket ID	Technical Skills	Technical Skills Comments	Goals
Form type	Team Work	Team Work Comments	Evaluator Name
Rank	Leadership	Leadership Comments	Date
Medical Knowledge	Professionalism	Professionalism Comments	Restore visibility
	Medical Knowledge Comments	Additional Comments	

Figure 4-17: Sample column visibility popup

columns. If the user selects the column buttons, they will be hidden on the table. The color of the column button will change once the user selects it. The transition of the color is to show the remaining columns. The restore visibility is another button created on the popup to return to the initial state of the table. The user can click anywhere outside of the popup to close it. The sample column visibility popup is shown in figure 4-17, where the white columns are the hidden ones. The restore column visibility is located at the end of the column list in default white color.

The last feature is the export to excel. It allows the user to export the data from the table. Once the user clicks on the excel button, it downloads the excel file with rows of the table data. By default, the exported spreadsheet does not show any row data if that table row holds null or empty fields. It is noteworthy to mention that the unusual characters are not allowed and will be removed. Every option provided in these pages has its own functionality and consists of a range of options. The export button also has some abilities like including the page title, table captions or custom message information in addition to the table data. The name of the excel file is given as the name of the page. For example, the simulation evaluation export spreadsheet file will have the simulation_evaluation name. The first and second rows of the file will be occupied with the name of the page and the small message of where it has been downloaded. The header section of the file is copied from the column headers of the table and highlighted with bold characters. There are some ways which allows

user to export the data is explained below.

1. The user can export a small portion of the data by search or filter option. If the user clicks on the export button after filtering the data as discussed above, the file will only be included with filtered data.
2. The user can export the selected columns of table data. This works if the user applies the column visibility functionality before exporting the data. The selected columns will not be posted on the downloaded file.
3. The user can export a filtered data with few columns. This export works when the user applies the filter and column visibility functionalities before exporting the table data. The downloaded file will have the filtered data with only a few selected columns.

The simulation and presentation evaluation tables are shown in figure 4-18 and the other daily feedback and case log are shown in figure 4-19. To secure the data, tables are posted with no data on it.

4.3.2 Residents' dashboard

Dashboard is designed in CARATS application as a collection of widgets that gives an overview of the evaluation results and performance metrics are of interest to the residents. The dashboard lets the residents monitor many metrics at once, so they can quickly check the evaluations results or see correlations between different reports. Residents' dashboard is customized for **simulation evaluation**, **presentation evaluation**, **case log**, and **daily feedback** results. Additionally, these results are used to produce badges and residency scores. All these metrics are developed for residents to recognize their growth at any point in time where they can enhance their performance during the semester. The metrics are displayed to the residents

Presentation Evaluation

Presentation Evaluation FormSummary

ExcelColumn visibility

Search: noth

Title

Rocket ID

Presentation Date

Your Rank

Resident Year

Was Organised

Was O

Showing 0 to 0 of 0 entries (filtered from 84 total entries)

PreviousNext

Simulation Evaluation Summary

Evaluation FormSummary

ExcelColumn visibility

Search: noth

Rocket ID

Form type

Rank

Medical Knowledge

Technical Skills

Team Work

Leadership

Professio

Showing 0 to 0 of 0 entries (filtered from 51 total entries)

PreviousNext

Figure 4-18: Presentaion and simulation summary tables

Daily Feedback Form

Daily Feedback FormSummary

Excel

Column visibility

Search:

Rocket Id

Evaluator Name

Evaluation Date

Work Together

Did Today

Showing 0 to 0 of 0 entries (filtered from 89 total entries)

Previous

Next

Choose File

No file chosen

Go!

Excel

Column visibility

Search:

NAME

CADIAC WITH CPB

CADIAC WITHOUT CPB

INTRATHORACIC NON CARDIAC

INTRACEREBRAL ENDOVASC

INTRACEF

Showing 0 to 0 of 0 entries (filtered from 21 total entries)

Previous

Next

Figure 4-19: Daily feedback and case log tables

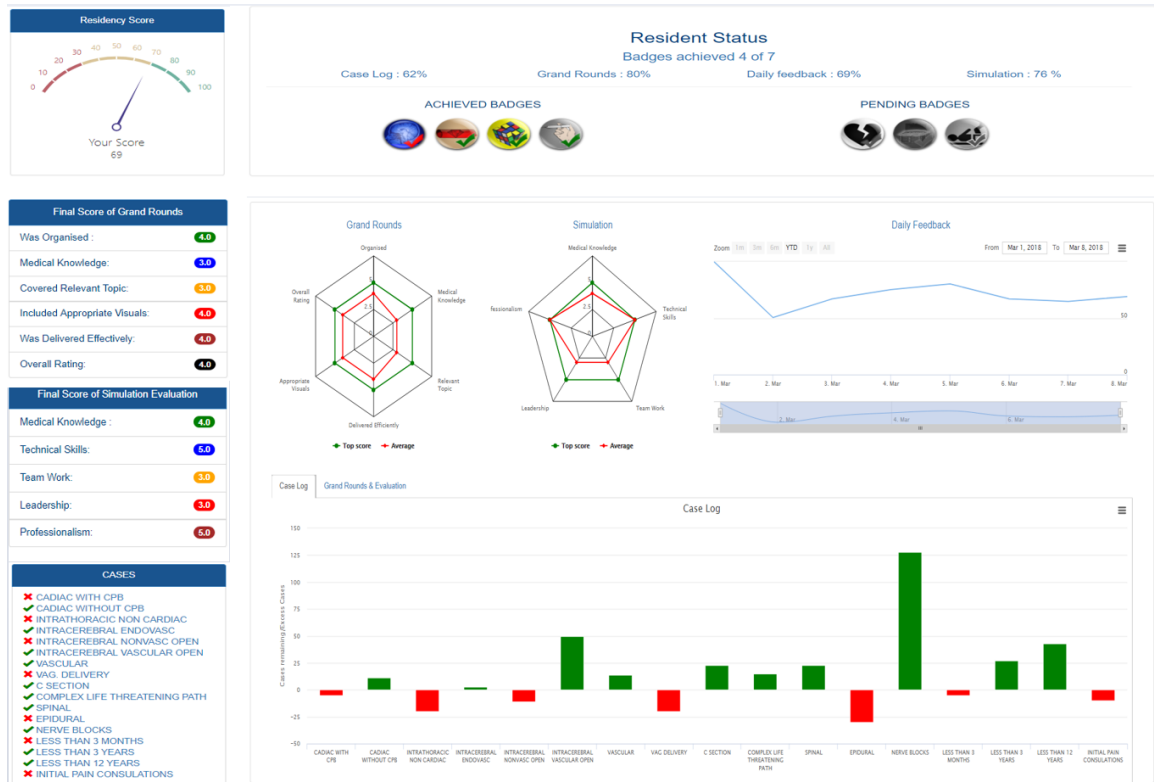


Figure 4-20: Resident's dashboard

after several computations. The complete dashboard is shown in figure 4-20. The dashboard is categorized into the following three sections depending on their purpose or usage:

1. Graphically interfaced individual feedback
2. Digital badges
3. Residency score

4.3.2.1 Graphically interfaced individual feedback

The graphically interfaced individual feedback category displays the simulation evaluation, presentation evaluations, case log, and daily feedback in analytics like

charts, bar graphs, data tables etc. The **simulation evaluation** and **presentation evaluations** are presented to the residents in spider graphs and listed tables. However, the daily feedback and case log are presented with bars having both negative and positive values, stock charts and check listed tables. The detailed description is provided in the following subsections and they are categorized based on the number of input sections.

1. Simulation evaluation
2. Grand rounds or Presentation
3. Case log
4. Daily feedback

Simulation Evaluation The usual way of displaying the data is projecting the numbers on the screen. But instead of the traditional display of data, the simulation evaluation is projected with charts, graphs, and data tables. The simulation evaluation data is displayed in the following formats:

- Spider graph
 - Listed representaion
 - Bar graph
 - Data table
1. Spider Graph: The spider chart represents the model for the graphic representation of all simulation evaluation fields. This visual medium provides the ability to see the resident's performance on evaluations. It displays the multi-variant data of all the simulation evaluations with five wings: residents' medical knowledge, technical skills, professionalism, leadership, and teamwork. The

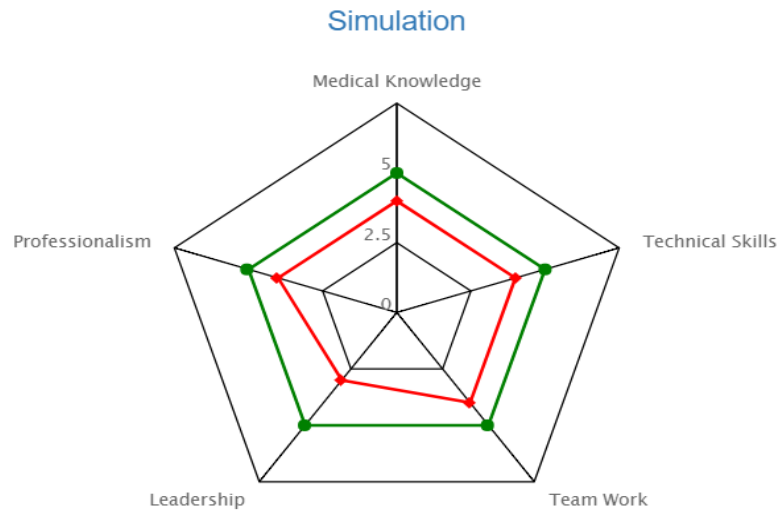


Figure 4-21: Spider graph representation of simulation evaluations

simulation evaluations spider graph is shown in figure 4-21. The outer ring in green color represents the maximum limit that the resident can reach in this simulation evaluation. The inner ring in red color illustrates the actual score of the resident.

2. Scorecard: Scorecard of the simulation evaluations is defined in the application as the listed representation of the average overall scores of the evaluations. The parameters are the same as the spider graph. The listed simulation evaluations scorecard is shown in figure 4-22.
3. Bar Graph: The simulation evaluations bar graph represents the overall score of a resident in all simulation evaluations submitted by each medical instructor. Each bar is the average simulation evaluation score by one medical instructor. The number of bars in the bar graph represents the number of medical instructors that have submitted simulation evaluations so far. The meaning of the value on the bar can be seen in table 4-1 and the simulation bar graph is shown in figure 4-23.

Final Score of Simulation Evaluation	
Medical Knowledge :	4.0
Technical Skills:	4.0
Team Work:	4.0
Leadership:	3.0
Professionalism:	4.0

Figure 4-22: Listed representation of simulation evaluations

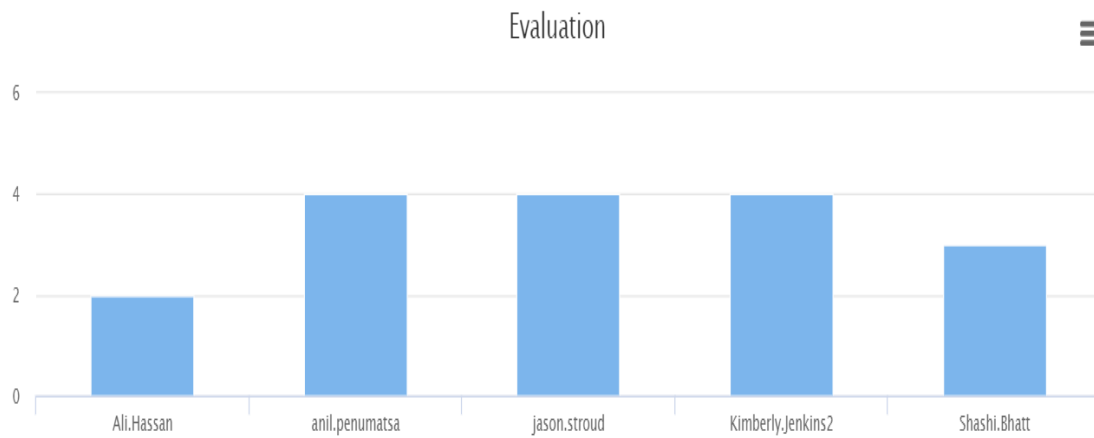


Figure 4-23: Bar graph representation of simulation evaluations

4. Data table: When the user clicks on the simulation link on top of the spider graph, a pop up will emerge with the simulation evaluation data table. The simulation evaluation data table is equipped with the resident's scores in **medical knowledge, technical skills, teamwork, leadership, and professionalism**, and their respective comments. In addition to that, it is also equipped with evaluator name, date of evaluations, goals and the type of simulation details. All the features equipped for the instructors' table are also provided to this data table.

Table 4.1: Bar graph value representation of simulation evaluation

Value	Meaning
1	Poor
2	Fair
3	Satisfactory
4	Good
5	Excellent

Grand rounds The grand rounds data is also presented in a similar way like simulation evaluations. They are presented in the following formats:

- Spider graph
- Listed representaion
- Bar graph
- Data table

1. Spider graph: The spider graph displays the multi-variant presentation evaluation data with six wings: organization of slides, increasing the medical knowledge, covering the relevant topic, delivering the topic efficiently, placing appropriate visuals, and overall rating. Figure 4-24 display the representation of grand rounds in spider graph. The scoring representation is also similar to the simulation evaluation. The spider ring in color green is the maximum limit which is 5 and ring in color red is the actual score.
2. Scorecard: The scorecard is the listed representation of spider graph details of grand rounds. It works similar to the simulation evaluations scorecard and it is shown in figure 4-25.
3. Bar graph: The bar graph represents the overall resident's score in a grand round presentation. Each bar represents the average of all evaluations on that presentation. The number of bars represents the number of grand rounds presented by the resident. The bar range is the mean of feedback submitted by the audience and it is limited from 1 to 5. The bar values meaning is shown in table 4-2, whereas the representation of grand rounds in the bar graph is shown in figure 4-26.

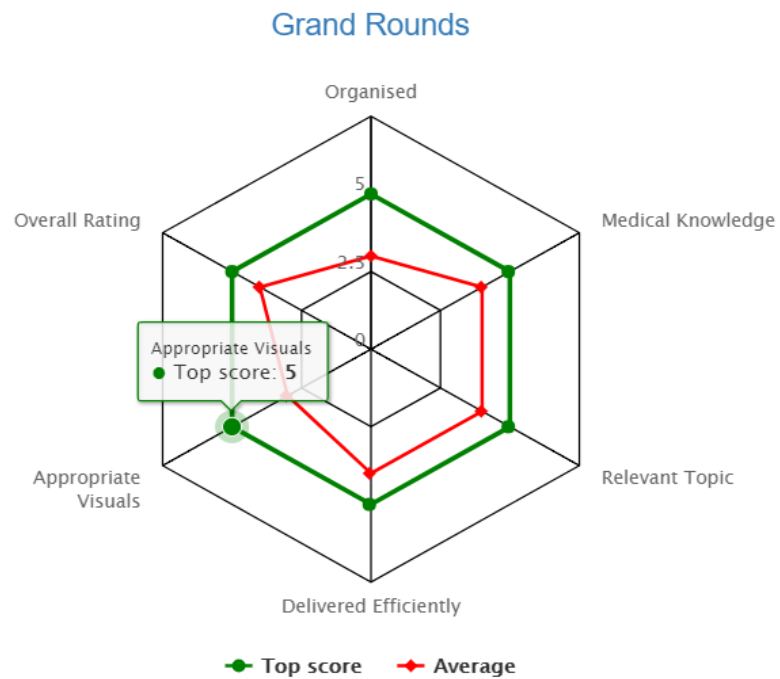


Figure 4-24: Spider graph representation of grand rounds

Final Score of Grand Rounds	
Was Organised :	3.0
Medical Knowledge:	4.0
Covered Relevant Topic:	4.0
Included Appropriate Visuals:	3.0
Was Delivered Effectively:	4.0
Overall Rating:	4.0

Figure 4-25: Listed representation of presentation evaluations

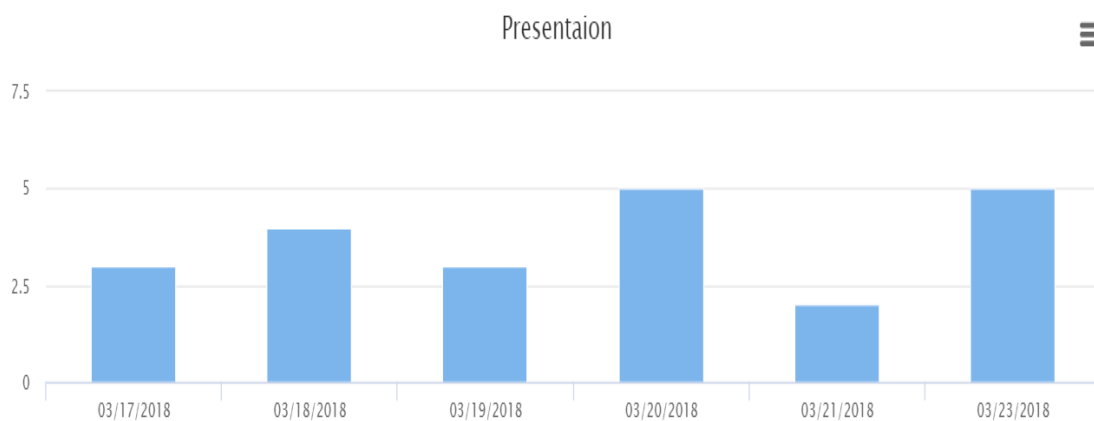


Figure 4-26: Bar graph representation of grand rounds

4. Data table: The grand rounds data table is equipped with resident's scores in the **organization, medical knowledge, the relevance of the topic, effective delivery, appropriateness of slides; and overall rating**. This table is also provided with all the features that instructors' table held.

Table 4.2: Bar graph values representation of grand rounds

Value	Meaning
1	Strongly disagreed
2	Disagreed
3	Neutral
4	Agreed
5	Strongly agreed

Case log The case log data imported into the system is presented in two different formats to the residents on the dashboard. One is a checklist which gives an overlook of the cases and the other is a bar graph plots the number of cases in bars.

- Checklist
- Bar graph

1. Checklist: A checklist is chosen in the dashboard to ensure residents' consistency and completeness in case logs. It works as a to-do list. The cases completed will be appended with the check sign in green color, whereas the cases pending are appended with the cross mark sign in red color. The primary task of the checklist is to notify the type of cases that residents' focus should be on. It audits against the established numbers in the system. The comparison numbers can be seen in table 4-3 and the checklist is shown in figure 4-27. The checklist is provided with a hover option. It helps to check the exact number of cases the resident required to perform or the access number of cases the resident has already completed. If the resident has performed more than the required number of cases, the hover will display the number of excess cases with no added symbol; otherwise, the negative symbol will be affixed to indicate underperformance of the number of cases.
2. Bar graph: The bar graph consists of parallel bars with varying lengths from negative to a positive range. The bars at any point in time represents the resident's position according to the ACGME standards. When the resident completes the target number of cases with reference to the ACGME standards, the rectangular bar crosses zero limit and appears in green; otherwise, it stays below zero and appears in red. The below zero position directs the resident to work more on that type of cases. The representation of case log in the bar

CASES	
✗	CADIAC WITH CPB
✓	CADIAC WITHOUT CPB
✗	INTRATHORACIC NON CARDIAC
✓	INTRACEREBRAL ENDOVASC
✗	INTRACEREBRAL NONVASC OPEN
✓	INTRACEREBRAL VASCULAR OPEN
✓	VASCULAR
✗	VAG. DELIVERY
✓	C SECTION
✓	COMPLEX LIFE THREATENING PATH
✓	SPINAL
✗	EPIDURAL
✓	NERVE BLOCKS
✗	LESS THAN 3 MONTHS
✓	LESS THAN 3 YEARS
✓	LESS THAN 12 YEARS
✗	INITIAL PAIN CONSULTATIONS

Figure 4-27: Checklist representation of case logs

graph is shown in figure 4-28. The menu symbol on the top right side of the graph allows the residents to print or download the graph. The residents can download the graph in four different formats, namely PNG, JPEG, PDF and SVG vector image. Once the residents click on the type of download option, the application pulls out the data from the case logs and downloads it to the resident with a name chart. The bars also provided with a hover option to know the current number of cases performed. The hover option will be either in green or red color depending upon the side of the graph. The hover displays the type of case, the residents' name and the number of cases. The bars graph varies its limit on y-axis according to the maximum number of cases completed in a type of case. Let's suppose if the resident completed 128 cases in nerve blocks which is the highest among all other case types, the system will assign the highest limit on the y-axis to 150.



Figure 4-28: Bar graph representation of case log

Table 4.3: ACGME standards of Case logs

Case	Limit
Cardiopulmonary bypass	10
Total Cardiac Cases	20
Intrathoracic non-cardiac	20
Intracerebral nonvascular open	11
Total intracerebral	20
Vascular	20
Vaginal delivery	40
Cegareas section	20
Complex life-threatening pathology	20
Spinal	40
Epidural	40
Nerve blocks	40
Children under three months	5
Children under three years	20
Children under twelve years	100
Initial pain consultations	20

Daily feedback Daily feedback submitted by the medical examiners is displayed in two different formats on the dashboard.

- Stock chart
- Data table

1. Stock chart: The stock chart is to help the residents to analyze their performance ups and downs. By observing the rise and fall of the chart, the residents will have an edge at predicting the upcoming performance. The first number in the stock chart displays the residents' score of their first-day performance. From

that point onwards, the score will be the cumulative average of all the resident's scores. The stock chart is provided with options to display data from the last one month, three months, six months, one year, or a custom date range. This options will be in the disabled state if the data is not within the custom range. The menu option is included at the top-right corner of the graph. This menu is customized with print and download options. The graph can be downloaded in PNG, JPEG, PDF, and SVG vector formats. This helps the residents to perform analysis more intensively. Additionally, if the user clicks on any point on the graph, a straight line will be highlighted. On top of the line, a hover display with the cumulative average up to that day and at bottom of the line date of that performance evaluation will be displayed. The scrollbar is added at the bottom to move either side of the graph. The representation of the daily feedback ratings in the stock chart is shown in figure 4-29.

2. Data table: When the user clicks on the daily feedback link on top of the stock chart, a pop up will emerge with daily feedback data table. The daily feedback data table is equipped with the resident's daily performance scores on different cases. The fields in the table are evaluator name, evaluations date, score, and comments. All the features equipped for the instructors' table are also provided to this data table.

4.3.2.2 Badges

Badges are presented for accomplishments as a token of appreciation for the good work [87]. Badges can be either physical or digital. In CARATS the badges are digital badges. The application is defined with a few benchmarks to assign digital badges to the resident. For example, if the resident completes the required number of cases in a chosen field, the badges will be awarded. The unaccomplished badges

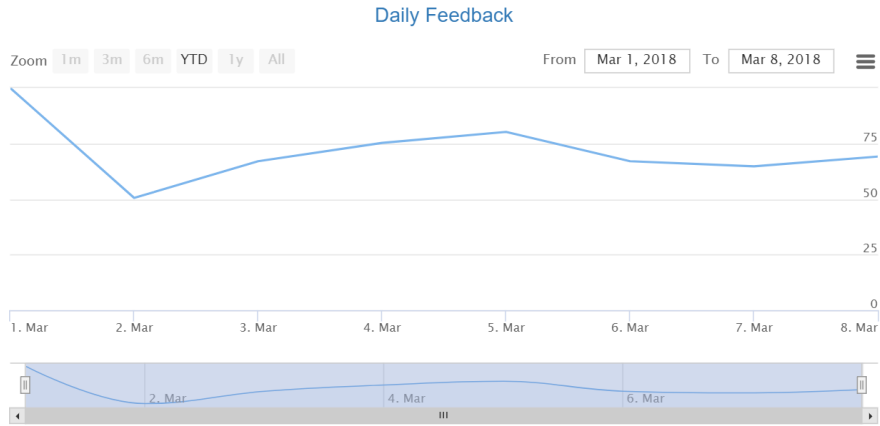


Figure 4-29: Stock graph representation of daily feedback

are displayed as gray images on the right-hand side. Once the resident completes the targeted number of cases, the badges become to color images and move to the left-hand side. Currently, the total number of badges in the system is seven. Depending on the completion of a number of cases in a field, the residents' will be assigned digital badges. For example, if residents complete twenty complex life-threatening pathology cases then they will be awarded the “Life Saver” badge. Once the badge is awarded, it moves to the left-hand side as shown in figure 4-30. Similarly, if residents perform five intrathoracic noncardiac cases, “Double the Lumens Double the Fun” badge will be awarded. The badges are awarded according to equation 4-1.

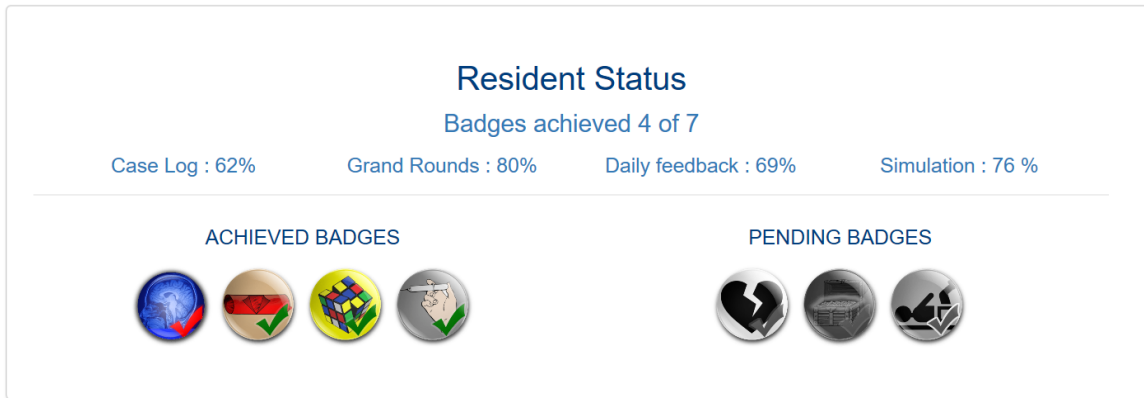


Figure 4-30: Resident badges

$$\begin{aligned}
 \text{Badges} = \left\{ \begin{array}{ll} \text{What's become of the broken hearted?} & \text{Cardiac} \geq 20 \\ \text{Head case} & \text{Intracerebral} \geq 20 \\ \text{Go with the flow} & \text{Vascular} \geq 20 \\ \text{Double the lumens, double the fun} & \text{Intrathoracic non cardiac} \geq 20 \\ \text{Life saver} & \text{Complex life threatening path} \geq 20 \\ \text{Daiper dandy} & \text{Less than three months old babies} \geq 20 \\ \text{I finally see your needle} & \text{Nerve blocks} \geq 50 \end{array} \right. \quad (4.1)
 \end{aligned}$$

On top of the **achieved** and **pending** badges, the individual evaluations are displayed with percentages. The percentages are calculated by taking the arithmetic mean and then expressed as a fraction of 100. The four sections **simulation evaluation**, **presentation evaluation**, **case log**, and **daily feedback** are presented in that block.

4.3.2.3 Residency Score

The residency score is displayed in meter gauge format. It is the score of the resident's overall performance. The gauge needle adjusts its position according to the residency score. The score range is limited between zero and one hundred. The residency score comprised of the resident's performance in simulation evaluations, grand rounds, case logs, and daily feedback.

We used equation 4-3 to calculate the residency score. The maximum score that resident can get is 100. Out of 100, the daily feedback gets the major share of 40 percent, badges get 32 percent, simulation evaluation and grand rounds 10 percent each, and case log gets 8 percent. The scores are scaled according to their percentages. The sum of all the parameters will give the residency score. The residency score is shown in figure 4-31 and equation 4-2 is used to equivalent the case log to eight percent.

Table 4.4: Weight-age for resident score.

form	weightage
Simulation Evaluation (SSE)	10
Grand Rounds (SGR)	10
Daily Feedback (SDF)	40
Number of Badges (NB)	32
Case log (f(NCL))	8

$$ResidencyScore = (SDF*0.4)+(SGR*2)+(SSE*2)+(NB*(32/7)+f(NCL) \quad (4.2)$$

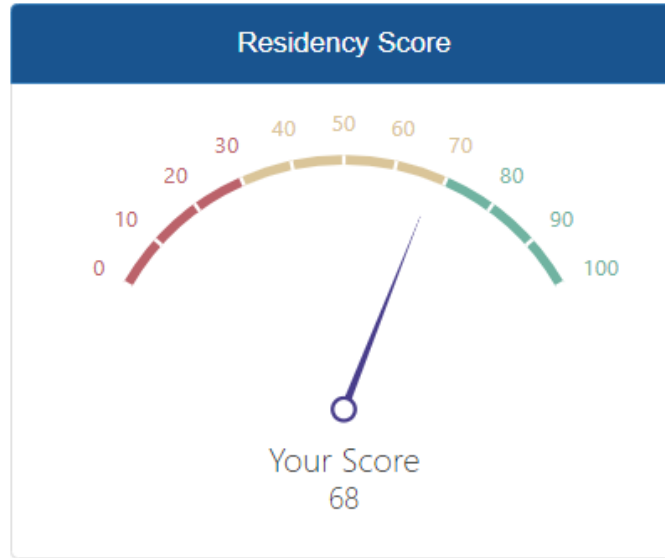


Figure 4-31: Residency score

$$f(NCL) = \begin{cases} 8 & NCL \geq 425 \\ \frac{NCL*8}{425} & \text{otherwise} \end{cases} \quad (4.3)$$

where: NCL = Number of cases completed so far

Chapter 5

Results & Discussions

The CARATS is one of the first applications introduced residency score and achievement badges for delivering residents' evaluations through web tools. Following the deployment of the CARATS, we conducted survey to get the residents' feedback. Residents were provided with forms to complete an anonymous survey regarding the CARATS web tool and its performance. The questions were given with 5 options ranging from one as bad to five as exceptional. The thematic analysis was conducted on the data. The survey results were aimed to explore the accessibility and impact of the web tool on the residents. In particular, we studied the interaction between the residents and the tool on five sections: daily feedback, dashboard, badges, residency score, and overall performance of the web tool.

5.1 Daily Feedback

The residents' reviews of daily feedback are presented in figure 4-1. Seventy two percent of the residents considered the daily feedback made an impact in day-to-day clinical education. Most of them believed that it encouraged them to correct their mistakes and perform well in subsequent clinical operations. They also believed that the daily feedback representation in stock charts allowed them to identify trends, pinpoint problem areas, and direct in an efficient manner.

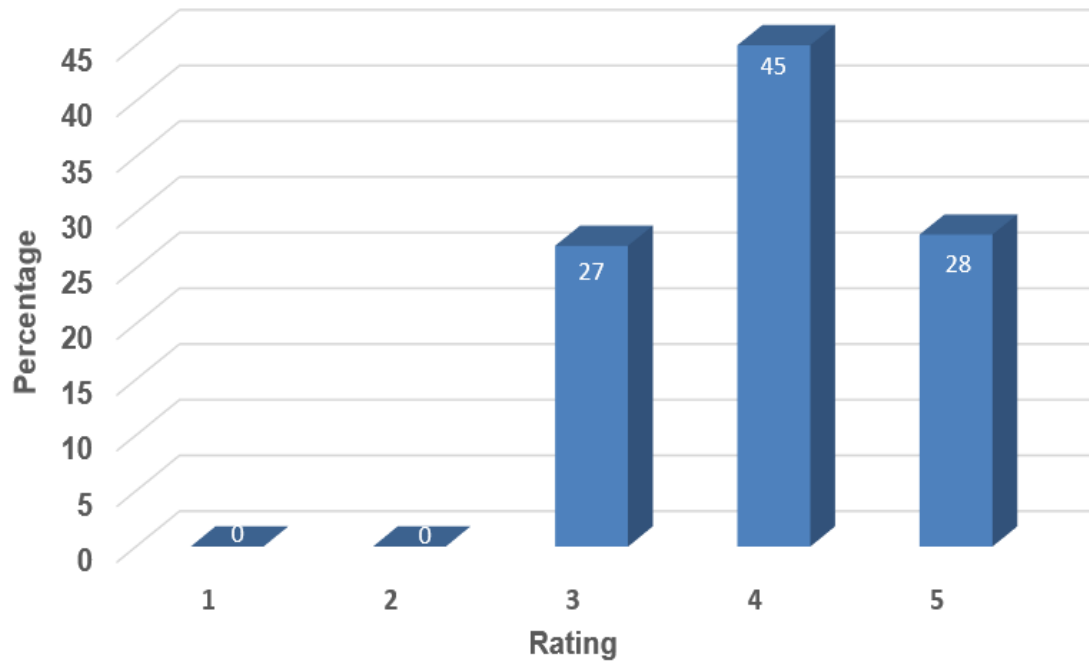


Figure 5-1: Bar graph representation of residents' rating on daily feedback

5.2 Dashboard

As figure 4-2 indicates, 90% of the residents considered the dashboard is a useful resource to receive evaluations. The majority of the residents reported that they were able to interpret evaluations at a glance. Through the analysis of the residents' reviews, it was found that 95% of the residents' felt dashboard is more informative and effective than the regular paper evaluations. An element of the dashboard that had a major influence on the residents was the inclusion of the evaluations in percentage. They reported that they were able to judge their standings in evaluations by percentage representation. All residents affirmed the dashboard information was effective, informative and it has made a difference in understanding the evaluations. Residents welcomed the use of analytics for portraying the data. Eighty-four percent of the residents agreed the use of analytics for evaluations made a huge difference in decision making. Residents commented that the graphs illustrating their relative

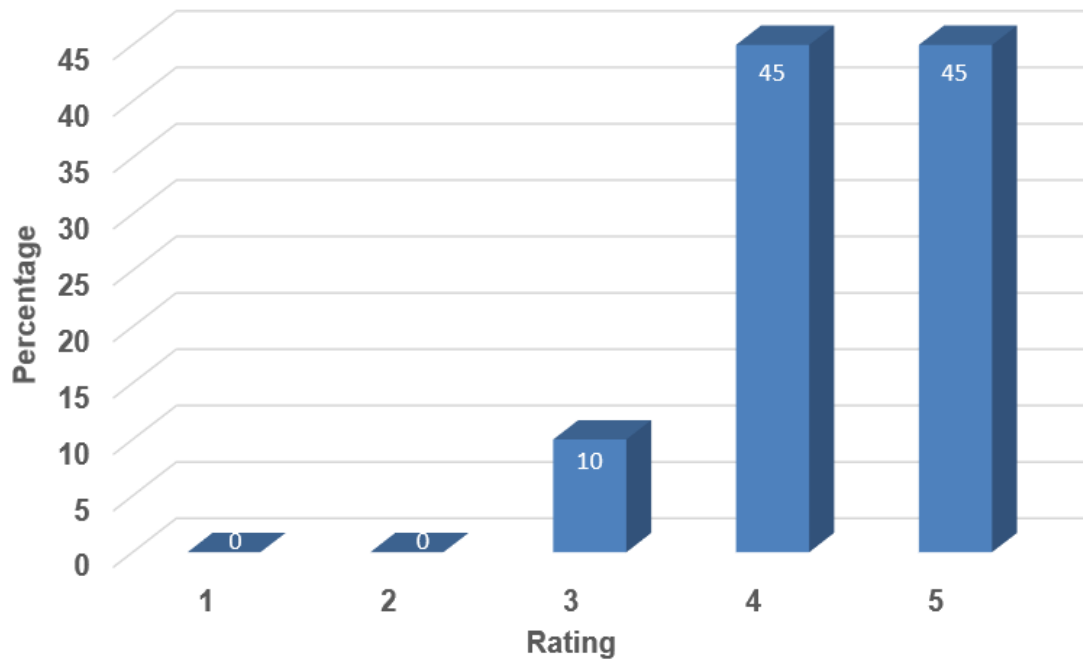


Figure 5-2: Bar graph representation of residents' rating on dashboard

position in evaluations were useful. Moreover, they added that the graphs would boost their motivation. Residents also appreciated the change of dashboard template for multiple screens like mobiles, tablets, and laptops. Dashboard improved residents perceptions of the evaluation's data and increased the degree of insight. The survey data also suggested the dashboards can act as a form of a guide for residents planning their graduation. One of the residents response when asked about the best thing in the dashboard is "it keeps track of your numbers need to graduate".

5.3 Residency Score

The residency score is divided into two categories; namely, usefulness and accuracy as shown in figure 4-3. Residents showed positive intentions towards residency score. Sixty-four percent of the residents believed the residency score is useful to

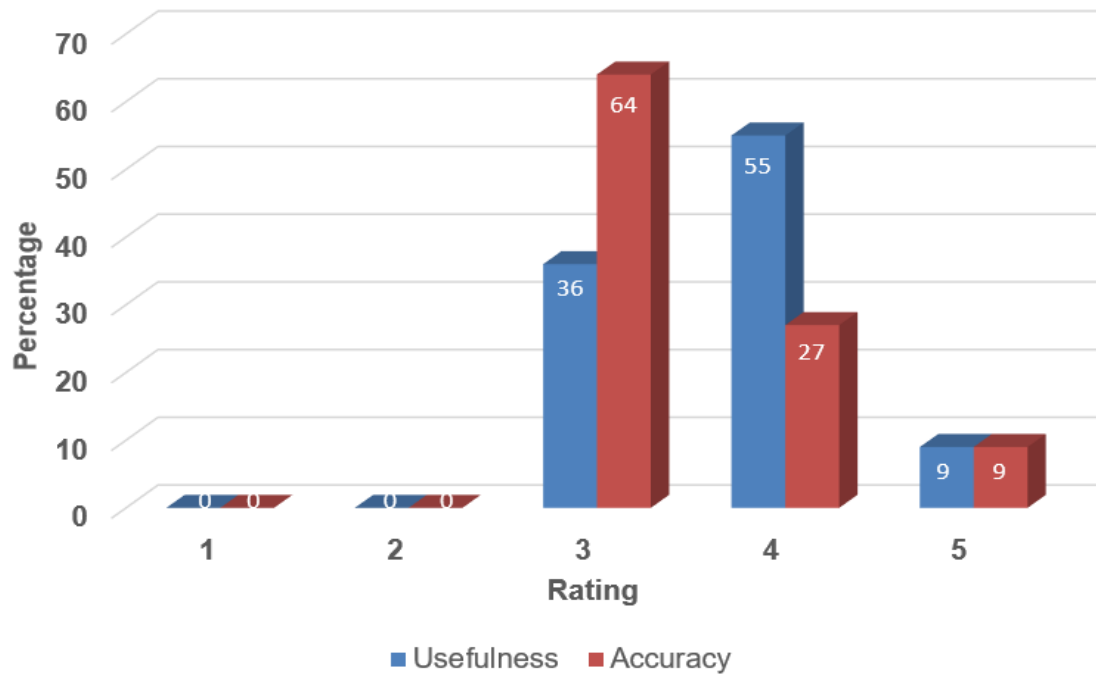


Figure 5-3: Bar graph representation of residents' rating on residency score

analyze their overall performance. They believed the residency score could help them to understand the amount of work required to showcase excellence in academics. Although the residents agreed with the usefulness of the residency score, only 36% of them admitted the current algorithm generated residency score is accurate. Majority of the residents suggested there is a need to improve current algorithm to increase the accuracy in predicting residency score.

5.4 Badges

Figure 4-4 presents an overview of two characteristics of badges: usefulness and motivation. We analyzed these characteristics along the research questions presented to the residents. Seventy-three percent of the residents acknowledged the inclusion of badges is highly useful in clinical education. Residents acknowledged with badges they

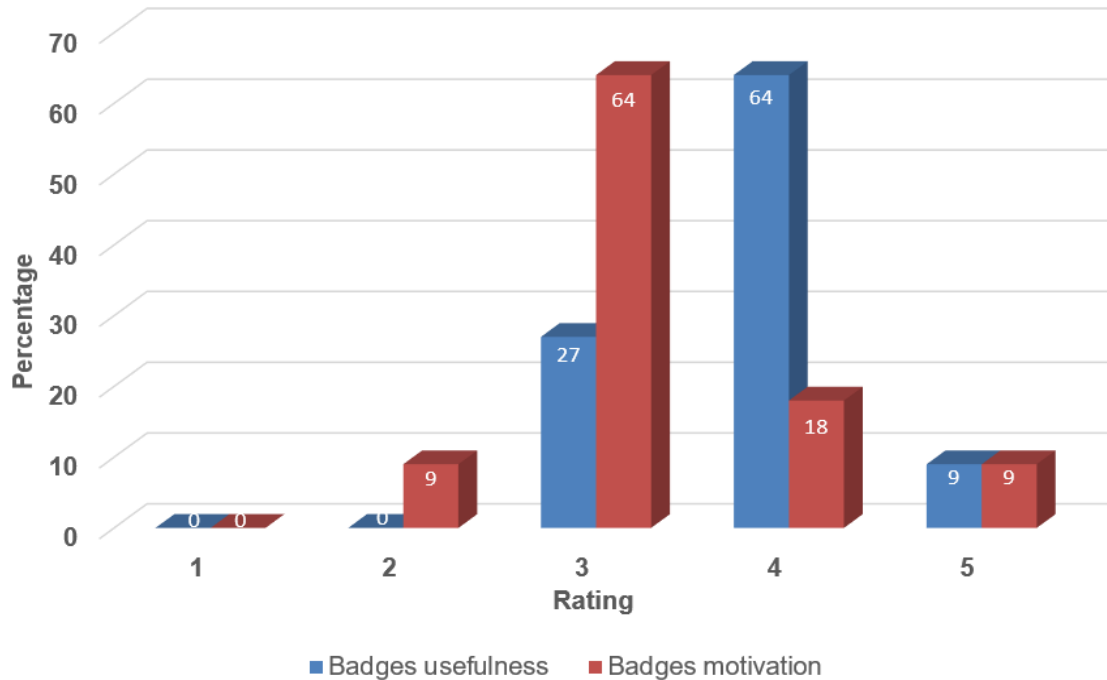


Figure 5-4: Bar graph representation of residents' rating on badges

can compete with others, or with a self-imposed goal to improve their performance in academics. Our analysis of badges usefulness identified that they can support resident empowerment by putting them in control of their clinical education. We also observed concerns that residents expressed about motivational factor through badges. Sixty-four percent of the residents responded that they are uncertain about the impact of badges in motivation. Despite the fact that the majority of the residents agreed with the usefulness of the badges, they considered the increase in the number of badges might allow them to judge the impact of badges in motivation. Residents preferred to have more badges in the system to obtain a better conclusion on badges motivational factor.

5.5 Overall performance of CARATS

The performance of CARATS is examined by several factors, namely, constructiveness, effectiveness, timeliness, overall performance, and the impact on medical evaluations. Reviews showed that the residents were able to receive real-time assessment and effective evaluations. The residents appreciated the real time assessments. Additionally, they confirmed that there used to be uncertainty in receiving paper evaluations. They were also not able to confirm the time factor in receiving paper evaluations. Ninety-eight percent of the residents praised CARATS for delivering quality evaluations. They confirmed the CARATS made evaluations easier and increased the transparency. They felt it made easier to evaluate themselves in clinical education. Eighty-two percent of the residents confirmed that the usage of CARATS made an immense impact in clinical evaluations as shown in figure 4-5 . Moreover, all residents rated the CARATS overall performance a better solution for effective feedback in clinical education compared to the paper evaluations. They observed the CARATS has the potential to become an e-portfolio.

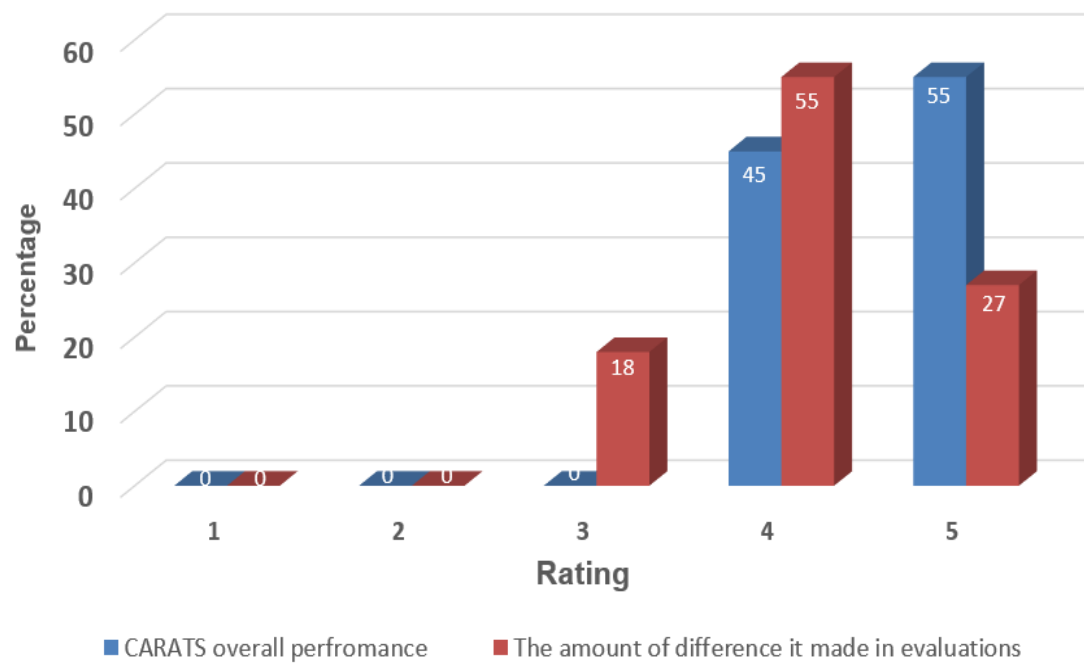


Figure 5-5: Bar graph representation of residents' rating on overall performance of CARATS

Chapter 6

Conclusion Remarks & Future work

The goal of this research was to produce a new effective feedback mechanism that is transparent, constructive, motivational, and informative. The web tool called CARATS is proposed in chapter 3. The primary idea behind the proposed web tool is to use analytics, badges, and algorithms to produce residency score. The reviews carried out here strongly confirmed that the residents accepted the representation of evaluations in a dashboard. Residents believed the dashboard transformed the evaluations into consumable information. The use of learner analytics through the CARATS application to different evaluations has shown great improvement in delivering the feedback. The longitudinal studies with dashboards are required to assess to what extent dashboards can contribute to improving residents performance in clinical education. Most of the concerns raised in the literature were about effective feedback mechanisms to residents, and the findings from this study have demonstrated that the CARATS has potential as an effective way to provide feedback as shown by the fact that the majority of the residents were able to identify gaps in their performance. Despite it's efforts and achievements, the current study could not confirm the accuracy of residency scores and motivation through badges.

The CARATS achieved the following: 1) A single interface to provide resident feedback in multiple areas of training; 2) a daily feedback form that is simple, mobile device-friendly and informative; 3) a dashboard view allowed residents to conveniently view evaluation data; 4) evaluation data is made available immediately (timely); 5) achievement badges introduced gamification to motivate resident activity; and 6) dynamic structure allowed for development of new evaluation inputs in the future.

6.1 Future work

The current product represents the first step toward the development of an all-inclusive resident evaluation suite. Further, we can develop a Clinical Competency Committee Dashboard view for instructors in which all of the residents' evaluation data will be available in one view. There are still many challenges to improve the effective feedback delivery via CARATS. However, the usefulness of dashboard, badges, and motivational feedback to students found in this study indicates that further exploration of other possibilities is worthwhile. While this analysis is not without its drawbacks or areas for continued improvement like the algorithm of residency score, the outcomes are such that continued use of CARATS would help to develop a new algorithm to produce the residency score and to ultimately assist students in their academics.

References

- [1] **Robert B. Barr & John Tagg (2012)**, *From Teaching to Learning A New Paradigm For Undergraduate Education*, *Change: The Magazine of Higher Learning*, 27:6, 12-26, DOI: 10.1080/00091383.1995.10544672
- [2] **Carol Evans**, *Making Sense of Assessment Feedback in Higher Education* , Volume: 83 issue: 1, page(s): 70-120.
- [3] **Johannes Brug & Karen Glanz** , *The Impact of Computer-Tailored Feedback and Iterative Feedback on Fat, Fruit, and Vegetable Intake* , Volume: 25 issue: 4, page(s): 517-531.
- [4] **David A. Davis, MD and Mary Ann Thomson, BHSc**, *Changing Physician Performance A Systematic Review of the Effect of Continuing Medical Education Strategies*, JAMA. 1995;274(9):700-705.
- [5] **Julian C Archer** , *State of the science in health professional education: effective feedback*
- [6] **Hattie, J. (1999)**, *Influences on student learning* , Unpublished inaugural lecture presented at the University of Auckland, New Zealand.
- [7] **Paul Black & Dylan Wiliam (2006)**, *Assessment and Classroom Learning*, *Assessment in Education: Principles, Policy & Practice* , 5:1, 7-74, DOI: 10.1080/0969595980050102
- [8] **Oxford English Dictionary Online**, [Accessed 06 06 2018.] 1995;70:898931.

- [9] **J M Monica van de Ridder**, *What is feedback in clinical education?*, Medical Education 2008; 42: 189197 doi:10.1111/j.1365-2923.2007.02973.x
- [10] **Dreifuerst, Kristina Thomas**, *THE ESSENTIALS of DEBRIEFING in Simulation Learning: A Concept Analysis*, March-April 2009 - Volume 30 - Issue 2 - p 109114.
- [11] **Thanos Hatziapostolou, Iraklis Paraskakis**, *Enhancing the Impact of Formative Feedback on Student Learning Through an Online Feedback System*, Electronic Journal of e-Learning Volume 8 Issue 2 2010, (pp111 - 122), available online at www.ejel.org.
- [12] **Astin, A.W. (1991)**, *Student Perceptions and Preferences for Feedback*, vol. 4, No. 3
- [13] **Frey, K. , Edwards, F. , Altman, K. , Spahr, N. and Gorman, R. S. (2003)**, *The Collaborative Care curriculum: an educational model addressing key ACGME core competencies in primary care residency training.*, Medical Education, 37: 786-789. doi:10.1046/j.1365-2923.2003.01598.x.
- [14] **Robert MD, MEd, MBA**, *Feedback for Learners in Medical Education: What Is Known? A Scoping Review*, Academic Medicine. doi:10.1097/ACM.0000000000001578
- [15] **Clare Delany & Elizabeth Molly**, *Clinical Education in the Health Professions*, ISBN 978 0 7295 3900(pbk).
- [16] **J M Monica van de Ridder**, *What is feedback in clinical education?*, Medical Education 2008; 42: 189197 doi:10.1111/j.1365-2923.2007.02973.x
- [17] **Butler D, Winne P**, *Feedback and selfregulated learning: a theoretical synthesis.*, Rev Educ Res1995;65 (3):24581.

- [18] **Gary J. Greguras** , *SelfConstruals, Motivation, and FeedbackSeeking Behaviors*, International Journal of Selection and Assessment, 16: 282-291. doi:10.1111/j.1468-2389.2008.00434.x.
- [19] **Jennifer Moye, Daniel C. Marson** , *Assessment of Decision-Making Capacity in Older Adults: An Emerging Area of Practice and Research* , The Journals of Gerontology: Series B, Volume 62, Issue 1, 1 January 2007, Pages P3P11.
- [20] **StevenLonn**, *Investigating student motivation in the context of a learning analytics intervention during a summer bridge program*, 10 August 2014.
- [21] **Leah P.Macfadyen**, *Mining LMS data to develop an early warning system for educators: A proof of concept*, 29 September 2009.
- [22] **Campbell, & Oblinger, D. (2007)**, *Academic analytics. Washington, DC: EDUCAUSE Center for Applied Research.*
- [23] **Maged N Kamel Boulos, Inocencio Maramba and Steve Wheeler**, *Wikis, blogs and podcasts: a new generation of Web-based tools for virtual collaborative clinical practice and education*, Boulos et al; licensee BioMed Central Ltd. 2006.
- [24] **Graham Walton, Susan Childs and Elizabeth Blenkinsopp**, *Using mobile technologies to give health students access to learning resources in the UK community setting* ,09 November 2005.
- [25] **Thanos Hatziapostolou, Iraklis Paraskakis**, *Enhancing the Impact of Formative Feedback on Student Learning Through an Online Feedback System* , Electronic Journal of e-Learning Volume 8 Issue 2 2010, (pp111 - 122), available online at www.ejel.org.

- [26] **Ms. Sana Rahman & Mr. Amit P. Raut**, *ONLINE STUDENT FEEDBACK SYSTEM*
- [27] **Tanya Elias**, *Learning Analytics: Definitions, Processes and Potential* ,January, 2011
- [28] **Buckingham Shum, S., Gaevi, D., & Ferguson, R. (Eds.). (2012).**, *Proceedings of 2ndInternational Conference on Learning Analytics and Knowledge* New York, NY: ACM.
- [29] **Verbert, Katrien & Duval, Erik**, *Learning Analytics Dashboard Applications*
- [30] **Thomas H. Davenport, Jeanne G. Harris**, *Competing on Analytics: The New Science of Winning*
- [31] **James G Dolan, Peter J Veazie and Ann J Russ**, *Development and initial evaluation of a treatment decision dashboard*, Dolan et al.; licensee BioMed Central Ltd. 2013
- [32] **Linda Corrin, Paula de Barba** , *Exploring students' interpretation of feedback delivered through learning analytics dashboards* , November 2014
- [33] **Stephen Porter PhD, Sean McCabe Michael Woodworth Kristine A. Peace** , *Genius is 1% inspiration and 99% perspiration or is it? An investigation of the impact of motivation and feedback on deception detection*,24 December 2010.
- [34] **Samuel Abramovich , Christian Schunn, Ross Mitsuo Higashi**, *Are badges useful in education? it depends upon the type of badge and expertise of learner*

- [35] **Linda Corrin, Paula de Barba** , *How do students interpret feedback delivered via dashboards?*, Proceedings of the Fifth International Conference on Learning Analytics And Knowledge, 978-1-4503-3417-4, Poughkeepsie, New York, 430-431
- [36] **Spring mvc** [online] , Available : <https://www.quora.com/What-are-the-advantages-and-disadvantages-of-architecture-1-tier-2-tier-3-tier-and-n-tier> .
- [37] **10 Reasons Why You Should Use AngularJS** [online] , Available : <https://www.sitepoint.com/10-reasons-use-angularjs/> .
- [38] **Web MVC framework** [online] , Available : <https://docs.spring.io/spring/docs/5.0.0.M4/spring-framework-reference/html/mvc.html> .
- [39] **Advantages and Disadvantages of Microsoft SQL** [online] , Available : <https://www.techwalla.com/articles/advantages-disadvantages-of-microsoft-sql> .
- [40] **What is the advantage of Microsoft SQL Server over Access?** [online] , Available : <http://expresstechnology.com/knowledgebase/what-is-the-advantage-of-microsoft-sql-server-over-access/> .
- [41] **Hibernate Advantages** [online] , Available : <https://javapapers.com/hibernate/hibernate-advantages/> .
- [42] **Hibernate Tutorial** [online] , Available : <https://howtodoinjava.com/hibernate-tutorials/> .
- [43] **Bootstrap 3: The Leading Responsive, Mobile-First Framework** [online] , Available : <https://www.upwork.com/hiring/development/bootstrap-3-front-end-framework-responsive-mobile-first-sites/> .
- [44] **6 Reasons to Choose the Bootstrap CSS Framework** [online] , Available : <https://www.ostraining.com/blog/coding/bootstrap/> .

- [45] **Bootstrap (front end framework [online]** , Available : [https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)) .
- [46] **Introducing JSON [online]** , Available : <https://www.json.org/>.
- [47] **Javascript [online]** , Available : <http://profsamscott.com/javascript/> .
- [48] **Overview of Spring MVC Architecture [online]** , Available : <https://terasolunaorg.github.io/guideline/1.0.1.RELEASE/en/Overview/SpringMVCOverview> .
- [49] **Spring MVC Architecture [online]** , Available : <https://www.java4coding.com/contents/spring/08springMVCArchitecture.html>.
- [50] **The DispatcherServlet [online]** , Available : <https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/ch15s02.html> .
- [51] **Interface HandlerMapping [online]** , Available : <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/servlet/HandlerMapping.html> .
- [52] **Interface HandlerAdapter [online]** , Available : <http://profsamscott.com/javascript/> .
- [53] **Interface Controller [online]** , Available : <https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/mvc/Controller.html> .
- [54] **Spring Boot @Controller [online]** , Available : <http://zetcode.com/springboot/controller/> .

- [55] **Spring MVC - Auto translation of view name** [online] , Available : <https://www.logicbig.com/tutorials/spring-framework/spring-web-mvc/view-name-translator.html> .
- [56] **Views and resolving them** [online] , Available : <https://docs.spring.io/spring/docs/3.0.0.M3/reference/html/ch16s05.html> .
- [57] **Model, ModelMap, and ModelAndView in Spring MVC** [online] , Available : <https://www.baeldung.com/spring-mvc-model-model-map-model-view> .
- [58] **Interface Model** [online] , Available : <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/ui/Model.html> .
- [59] **Building REST services with Spring** [online] , Available : <https://spring.io/guides/tutorials/bookmarks/> .
- [60] **Understanding REST** [online] , Available : <https://spring.io/understanding/REST> .
- [61] **Using HTTP Methods for RESTful Services** [online] , Available : <https://www.restapitutorial.com/lessons/httpmethods.html> .
- [62] **Hibernate Tutorial** [online] , Available : <https://howtodoinjava.com/hibernate/merging-and-refreshing-hibernate-entities/> .
- [63] **Hibernate Merging and Refreshing Entities** [online] , Available : <http://profsamscott.com/javascript/> .
- [64] **Hibernate Query Cache Example** [online] , Available : <http://www.tecbar.net/hibernate-query-cache-example/> .

- [65] **Flow of Hibernate Application** [online] , Available : <http://www.java2success.com/hibernate/hibernate-flow.jsp> .
- [66] **Hibernate - Sessions** [online] , Available : https://www.tutorialspoint.com/hibernate/hibernate_sessions.htm .
- [67] **Hibernate Transaction Management Example** [online] , Available : <https://www.javatpoint.com/hibernate-transaction-management-example> .
- [68] **Hibernate - Persistent Class** [online] , Available : https://www.tutorialspoint.com/hibernate/hibernate_persistent_classes.htm .
- [69] **Anomalies** [online] , Available : https://www.sqa.org.uk/e-learning/MDBS01CD/page_22.htm .
- [70] **Amazon Route 53** [online] , Available : <https://aws.amazon.com/route53/> .
- [71] **What Is a Classic Load Balancer?** [online] , Available : <https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/introduction.html> .
- [72] **Elastic IP Addresses.** [online] , Available : https://docs.aws.amazon.com/index.html#lang/en_us .
- [73] **Amazon Relational Database Service (RDS)** [online] , Available : <https://aws.amazon.com/rds/> .
- [74] **Gerald Kayingo, PhD, PA-C, Virginia McCoy** , *The Health Professions Educator: A Practical Guide for New and Established* .
- [75] **Okuda, Y. , Bryson, E. O., DeMaria, S. , Jacobson, L. , Quinones, J. , Shen, B. and Levine, A. I. (2009),** *Utility of Simulation in Med-*

- ical Education: What Is the Evidence?*. , Mt Sinai J Med, 76: 330-343. doi:10.1002/msj.20127.
- [76] **Cooper, J B and Taqueti, V R**, *A brief history of the development of mannequin simulators for clinical education and training* , BMJ Quality & Safety 2004;13:i11-i18.
- [77] **Gaba DM**, *The future vision of simulation in health care* , BMJ Quality & Safety 2004;13:i2-i10.
- [78] **Van Hoof, Thomas J. MD, EdD; Monson, Robert J. PhD; Majdalany, Gibran T. PhD; Giannotti, Tierney E. MPA; Meehan, Thomas P. MD, MPH**, *A Case Study of Medical Grand Rounds: Are We Using Effective Methods?* , Anesthesiology 8 2006, Vol.105, 279-285. doi.
- [79] **Hebert, Randy S. MD, MPH; Wright, Scott M. MD** , *Re-examining the Value of Medical Grand Rounds* , Anesthesiology 8 2006, Vol.105, 279-285. doi.
- [80] **Salazar, Dane MD; Schiff, Adam MD; Mitchell, Erika MD; Hopkinson, William MD** , *Variability in Accreditation Council for Graduate Medical Education Resident Case Log System Practices Among Orthopaedic Surgery Residents* , Anesthesiology 8 2006, Vol.105, 279-285. doi.
- [81] **Corbeil, Maria Elena; Corbeil, Joseph Rene; Rodriguez, Ignacio E** , *DIGITAL BADGES IN HIGHER EDUCATION: A THREE-PHASE STUDY ON THE IMPLEMENTATION OF DIGITAL BADGES IN AN ONLINE UNDERGRADUTE PROGRAM* , Issues in Information Systems . 2015, Vol. 16 Issue 4, p1-9.
- [82] **Frey, K. , Edwards, F. , Altman, K. , Spahr, N. and Gorman, R. S. (2003)** , *The Collaborative Care curriculum: an educational model address-*

- ing key ACGME core competencies in primary care residency training.*, Medical Education, 37: 786-789. doi:10.1046/j.1365-2923.2003.01598.x.
- [83] **Hyland, P. (2000)**, *Learning from feedback in assessment*, In P. Hyland & A. Booth (Eds.), *The practice of university history teaching* (pp. 233-247). Manchester: Manchester University Press.
- [84] **Astin, A.W. (1991)**, *Assessment for excellence: The philosophy and practice of assessment and evaluation in higher education*, New York: Macmillan Publishing Company.
- [85] **Black, P & William, D. (1998)**, *Assessment and classroom learning. Assessment in Education: Principles, Policy & Practice*, 5 (1), 7-74.
- [86] **Hattie, J.A., & Timperley, H. (2007)**, *The power of feedback. Review of Educational Research*, 77, 81-112.
- [87] **Corrin, Linda & de Barba, Paula. (2014)**, *Exploring students' interpretation of feedback delivered through learning analytics dashboards*.
- [88] **Veloski J, Boex JR, Grasberger MJ, Evans A, Wolfson DB**, *Systematic review of the literature on assessment, feedback and physicians clinical performance* : BEME Guide No. 7. Med Teach 2006;28 (2):11728.
- [89] **Rolfe IE, Sanson-Fisher RW**, *Translating learning principles into practice a new strategy for learning clinical skills* Med Educ 2002;36:34552.
- [90] **Georges L. Savoldelli, M.D., M.Ed.**, *Value of Debriefing during Simulated Crisis Management: Oral versus Video-assisted Oral Feedback*, Anesthesiology 8 2006, Vol.105, 279-285. doi.

- [91] **Dreifuerst, Kristina Thomas**, *THE ESSENTIALS of DEBRIEFING in Simulation Learning: A Concept Analysis*, March-April 2009 - Volume 30 - Issue 2 - p 109114.