DESIGN AND IMPLEMENTATION OF SPECIAL EDUCATION APPS INFORMATION MANAGEMENT SYSTEM

A thesis submitted
to Kent State University in partial
fulfillment of the requirements for the
degree of Master of Sciences

by
Xiqian Han
April 3, 2017
© Copyright
All rights reserved
Except for previously published materials
Thesis written by

Xiqian Han

M.S., Kent State University, USA, 2017
M.S., Shandong University, China, 2012
B.S., Shandong University, China, 2008

Approved by

Dr. Cheng-Chang Lu, Chair, Master Thesis Committee
Dr. Richard E. Ferdig, Members, Master Thesis Committee
Dr. Austin Melton, Members, Master Thesis Committee
Dr. Xiang Lian, Members, Master Thesis Committee

Accepted by

Dr. Javed Khan, Chair, Department of Computer Science
Dr. Janis Crowther, Dean, College of Arts and Science
TABLE OF CONTENTS

LIST OF FIGURES..................................................................................................................vi

ACKNOWLEDGEMENTS..........................................................................................................viii

CHAPTER 1 INTRODUCTION .....................................................................................................1

1.1 Introduction of Mobile Application and Special Education.................................1

1.2 Existing Problem of Mobile Application and Special Education......................1

1.3 Related Work..................................................................................................................2

1.4 The Overview of SpedApps......................................................................................4

CHAPTER 2 REQUIREMENTS ANALYSIS ............................................................................6

2.1 User Requirements......................................................................................................6

2.1.1 System Environment Requirements...................................................................6

2.1.2 User Interface.......................................................................................................7

2.1.3 User Characteristics and Assumptions.................................................................7

2.2 Functional Requirements...........................................................................................7

2.2.1 User Case Diagram..............................................................................................7

2.2.2 Unregistered Users’ User Case Descriptions.........................................................9

2.2.2.1 Search Apps.....................................................................................................9

2.2.2.2 Registration....................................................................................................9

2.2.3 Normal Users’ User Case Descriptions...............................................................10

2.2.3.1 Search Apps...................................................................................................10

2.2.3.2 Users Log In....................................................................................................10

2.2.3.3 Users Log Out.................................................................................................11

2.2.3.4 Users Forget Password and Reset Password...............................................11

2.2.3.5 Users Suggest Apps.......................................................................................11
2.2.3.6 Users Suggest App’s Tags ........................................................................... 12
2.2.3.7 Post Review ................................................................................................... 12
2.2.3.8 Profile Management ...................................................................................... 13
2.2.4 Admin Users’ User Case Descriptions ............................................................. 13
   2.2.4.1 Search Apps ............................................................................................... 13
   2.2.4.2 Users Log In ............................................................................................... 14
   2.2.4.3 Users Log Out ........................................................................................... 14
   2.2.4.4 Admins Add Apps ...................................................................................... 14
   2.2.4.5 Apps Management ..................................................................................... 15
   2.2.4.6 User Management ..................................................................................... 16
   2.2.4.7 User Comments Management ................................................................. 17
   2.2.4.8 News Management .................................................................................... 17
   2.2.4.9 Report ........................................................................................................ 18
2.2.5 Non-Functional Requirements ......................................................................... 18
   2.2.5.1 Security ..................................................................................................... 18
   2.2.5.2 Speed ......................................................................................................... 18
   2.2.5.3 Usability ..................................................................................................... 18
   2.2.5.4 Maintainability .......................................................................................... 18
   2.2.5.5 Scalability .................................................................................................. 19
2.3 Summary ............................................................................................................ 19

CHAPTER 3 DESIGN ................................................................................................. 20
3.1 System Architecture ............................................................................................ 20
3.2 Web Interface Design .......................................................................................... 21
3.3 Database Design ................................................................................................. 23
   3.3.1 Entity-Relationship Diagram ....................................................................... 23
CHAPTER 4 IMPLEMENTATION

4.1 User Account Module

4.1.1 Salted password hashing Algorithms

4.1.2 Authentication and Authorization

4.2 App Search Module

4.3 Admin Module

4.3.1 App Management

4.3.2 App Evaluation

4.4 User Module

4.5 Summary

CHAPTER 5 TESTING

5.1 The Importance of Software Testing

5.2 Testing Plan

5.2.1 Website Display Test

5.2.2 Website Accessibility Test

5.2.3 Function testing

5.3 Summary

CHAPTER 6 CONCLUSION

REFERENCES
LIST OF FIGURES

Figure 1 Unregistered User Case Diagram ......................................................... 8
Figure 2 Normal User Case Diagram ................................................................. 8
Figure 3 Admin User Case Diagram ................................................................. 8
Figure 4 System Architecture ........................................................................... 21
Figure 5 SpedApps Home Page ....................................................................... 22
Figure 6 SpedApps Admin User Dashboard ................................................. 23
Figure 7 SpedApps Admin User App Management Dashboard ................ 23
Figure 8 SpedApps ER Diagram ..................................................................... 25
Figure 9 Results of Mapping Normal Entities: APP and USER ...................... 26
Figure 10 Results of Mapping Weak Entities: COMMENT and MODIFICATION... 26
Figure 11 Results of Mapping Relationship Types ......................................... 27
Figure 12 Results of Mapping Multivalued Attributes ................................... 27
Figure 13 Pseudo Code of Salted Password Hashing Algorithms ................. 30
Figure 14 Screenshot of User Registration ...................................................... 31
Figure 15 Screenshot of User Login ................................................................. 31
Figure 16 Session Code Sample ..................................................................... 32
Figure 17 Password Recovery ...................................................................... 33
Figure 18 Screenshot Of Search App Main Page ........................................... 34
Figure 19 Screenshot Of App Search Result with "Free" as Filter .................. 34
Figure 20 Sample Code Of App Details by AJAX .......................................... 35
Figure 21 The Workflow Chart Of Admin Module ......................................... 35
Figure 22 Screenshot Of Approved Not Edited Apps List .............................. 37
Figure 23 Sample Code of App Category by AJAX ....................................... 39
Figure 24 Screenshot Of Checking the existence of Apps ............................. 40
Figure 25 Sample Code of App Crawler Function ........................................ 40
Figure 26 Screenshot of App Evaluation Form ........................................... 41
Figure 27 Workflow of Normal User Module ............................................. 41
Figure 28 Screenshot of App Main Page ................................................... 42
Figure 29 Screenshot of Writing a Review ................................................ 42
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Cheng-Chang Lu, for his guidance and support during my master study. He was gracious to provide me with the opportunity to join his research group. Dr. Lu was very supportive whenever I ran into a trouble spot or had a question about my study. I appreciate his support for me to become who I am and in what I am doing now.

I also would like to thank Dr. Richard E. Ferdig for giving me this opportunity to join this fantastic project. Without his guidance and organization, we cannot finish so many amazing projects. He is a great mentor, and I have learned a lot from him.

I also would like to thank and Dr. Austin Melton and Dr. Xiang Lian, for the comments and feedbacks that enriched this thesis.

I also would like to thank AT &T. his project was funded by a corporate gift from AT & T. I would like to thank members in Dr. Ferdig's research group helped me with the design of this application. They put significant efforts into this project. Without them, we cannot have such valuable app evaluation database system.

I would like to thank my parents who support me with all their hearts. Finally, I thank my husband for always being there for me.
CHAPTER 1

INTRODUCTION

1.1 Introduction of Mobile Application and Special Education

Mobile devices are becoming more popular worldwide, changing the way people learn and access information (Alvarez, Alarcon, & Nussbaum, 2011). In the last decade, thousands of novel applications in mobile learning are emerging. Research has provided evidence that mobile technology is expanding the learning experience both inside and outside the classroom. It has many advantages over traditional education technology such as being interactive, immersive and engaging, increases in student’s motivation and the ease of personalized and collaborative learning (Al-Hmouz, Shen, Yan, & Al-Hmouz, 2010). However, there are some considerations when implementing mobile educational applications for students, especially for students with special educational needs.

Special education programs are designed for teaching students with special educational disabilities such as: learning disabilities, mental retardation, speech impairment, physical disabilities and developmental disabilities (Glass, 1983).

1.2 Existing Problem of Mobile Application and Special Education

Due to these special educational needs, some of the mobile applications apparently cannot meet the need of all the students. It is important to identify which applications are
designed with special educational considerations.

Also, thousands of special educational applications are launched on the iTunes and Google Player daily, but the question for parents, educators, and therapeutic professionals is how they can identify which applications are worth using. The applications industry is out to make a profit, and many times application developers are very astute at making an application that "looks" educational, but in reality is just full of ads and sneaky ways of getting young kids to buy something or play games. While there are many advantages of mobile applications, the simple fact is that it is tough for parents, educators, and therapeutic professionals to select appropriate applications for their kids, especially for the kids with special needs.

Parents and educators cannot only rely on mobile applications to effectively engage learners. Because the apps company may not fully realize the special need and implement the apps meeting the special requirements. To learn how to use mobile applications in the classroom effectively, educators and therapeutic professionals need some professional instructional guidance. However, many teachers and therapeutic professionals have rarely received any formal training in the knowledge, skills, and practices that are necessary to integrate mobile technology in ways that will affect positive changes in students (Balderaz & Rosenblatt). New mobile applications are emerging every day, and one-time training is not possible.

1.3 Related Work

With the massive quantity of mobile applications available, some groups and labs have already built some evaluation tools to address this mobile application selection issue. There are two potential solutions to address this problem: free form style reviews and reviews based on some rubrics.

Free form style reviews are totally free form reviews, such as App Store user review and
Amazon customer reviews, where no one is using a template or rubrics. Parents, educators, and therapeutic professionals can post and share these reviews through the Internet. Reading some user reviews before downloading the app, we can get broad feedback on a product. It can be very useful when we're looking for general issues. Some professional researchers recommend applications and post them on their blogs or social networking sites, like Twitter or Facebook. Some of them also conduct app evaluations and post their reviews including user guidance through some evaluation blogs such as Teacherswithapps (http://www.teacherswithapps.com/).

However, free form style reviews have many limitations. First, when editors in Teacherswithapps generate each review, it seems that there is no template to follow and they need to create a detailed report based on every application. It is a time-consuming work. Second, different people might have different evaluation criteria. It can cause a data consistency issue even within the same evaluation site with many editors. Third, for the users such as educators and parents, the information in these detailed reviews is not quite straightforward. Sometimes, they might lose patience to read individual reviews.

The second solution is generating reviews based on predefined rubrics and templates. There are many online searchable tools available now, such as Common Sense Media (https://www.commonsensemedia.org/), MathApps (http://mathapps.kent.edu/), etc. Common Sense Media is a nonprofit education website. It has seven categories: Apps, Books, Games, Movies, Music, TV, and Website and publishes ratings and reviews for nearly everything kids want to watch, read and learn. First, generating reviews based on predefined rubrics and templates can ease the workload of editors when they evaluate each application. The professional editors will evaluate each application based on predefined templates or rubrics, such as evaluating app from app objective, content area, audience, protocols, and progress report, etc.,
which will establish a clear expectation for all users. Once the template has been formed, the editors just need to follow the evaluation criteria when evaluating each app. Second, it provides concise information, and it is easy for users to search and get their resources with standard filters, such as filtering by grade. When you’re looking to find the best interactive multiplication application that can be used in your classroom tomorrow, you can just type “multiplication, grade, and objective” and the tool will query the database and return the matching applications. You don’t need to navigate hundreds of lines of words to figure out whether the application is a must.

However, there are no existing mobile application evaluation sites for special educational applications. Research has already provided evidence that traditional and special educational are different. There is an urgent demand for creating a tool to help the educators in special educational area (Walser, Ayres, & Foote, 2012).

1.4 The Overview of SpedApps

The main goal of our project is implementing a special educational applications information management system called SpedApps. SpedApps is not only a scientific, informative and timely applications management system, but also is a searchable online database that catalogs high-quality applications for special education, especially those focusing on science, technology, engineering, arts, and math (STEAM).

The database categorizes each app by the app name, publisher, price, objective, content area, domain, audience, and disability tags. Most of the app evaluation reports contain an objective review, evaluating whether the app provides the information such as: practice, feedback, progress, usability, and customization. Users can query the database through different filters such as: app keyword search, app name search, cost, topic search, etc.
There are three groups of target users: admin users, normal users, and unregistered users. Admin users are professional special educators, and they will evaluate applications, post their reviews based on rubrics, manage and maintain the database. Admin users must continually evaluate the apps in the database and modify app evaluation reports based on user suggestions and update version of the apps.

Normal users and unregistered users are usually parents, educators, and therapeutic professionals. They can access the evaluation database through the app search module interface. The site currently contains reviews of more than 580 applications. The reviews can be filtered by subject matter, name, or whether they're free or for sale. SpedApps is not only a database, and it is more like a social network community. Normal users can create their login accounts, post their reviews and also can suggest modifications of currently existing formal evaluations. Many research findings indicated that educators and parents not only like to search apps through social networks but also can provide useful suggestions for other users (Richard E. Ferdig, 2016).
CHAPTER 2

REQUIREMENTS ANALYSIS

Requirement analysis is to identify and analyze the requirements of the system. We will discuss the user requirements and system requirements, functional requirements and nonfunctional requirements in this chapter.

2.1 User Requirements

User requirements are the statements of facts and assumptions that define the expectation of the system regarding the environment, user interface, and user characters.

2.1.1 System Environment Requirements

SpedApps is based on the Browser/Server architecture. The software and hardware requirements for servers and clients are as following:

- Server: The web server is set up with the latest Apache, PHP, and MySQL package. MySQL will be used to connect to the database in the backend, and script language is PHP.
- Client: The software must be able to run on a multitude of operating systems, such as: OSX, Windows, and Unix. The client’s system should have a web browser installed.
2.1.2 User Interface

The system interacts with its users through a web-based GUI. Normal users or admin users can log in to the system either by PC, Mac, desktop, phone or tablet.

2.1.3 User Characteristics and Assumptions

Users occupy a broad spectrum of technical expertise and experiences. Users, however, are assumed to understand the basics of computer operation, such as: how to access a website, navigation on the website, etc. Beyond these basic assumptions, an instructional material will be provided to ensure successful operation.

2.2 Functional Requirements

Functional requirements define what a system is required to accomplish. User cases describe complete interactions between the software and users. There are three kinds of users in this system: admin users, normal registered users, and unregistered users. We analyzed functional requirements using Unified Modeling Language (UML), user case diagram (Rumbaugh, Jacobson, & Booch, 2004).

2.2.1 User Case Diagram

A user case diagram is a description of the interactions among users and the application modules. As mentioned above, three kinds of users are using the system. The primary roles of each user groups are different. Admin users are usually professional educational specialists. They support app management, user management and processes of the application evaluation. Unregistered users can only search apps. Normal registered users are normal users who can not only search apps but also participate in the uploading and reviewing apps.

The user case diagrams are shown in Figure 1, Figure 2 and Figure 3.
Figure 1 Unregistered User Case Diagram

Figure 2 Normal User Case Diagram

Figure 3 Admin User Case Diagram
2.2.2 Unregistered Users’ User Case Descriptions

2.2.2.1 Search Apps

- Requirements: Allow users to search apps by the name, app name, app publisher, app price, app objective, app content area, app domain, app audience, and app disability tags and sort the searching result by name or date.
- Preconditions: None.
- Postconditions: None.
- Scenarios:
  - Apply all the input search filter conditions.
  - Retrieve the app list from the database according to the search filters.

2.2.2.2 Registration

- Requirements: Allow a new normal user to register to the system.
- Preconditions: User does not exist in the database.
- Postconditions: User exists in the database.
- Scenarios:
  - User submits his/her registration request by providing email (unique), first name, last name, and password.
  - Save newly created instance to the database with the status “inactive”.
  - The system sends the user an activation link via email.
  - The User activates his/her account by clicking the link.
  - Update the status of that user to “active”.


2.2.3 Normal Users’ User Case Descriptions

2.2.3.1 Search Apps

- Requirements: Allow users to search apps by the name, app name, app publisher, app price, app objective, app content area, app domain, app audience, and app disability tags and sort the searching result by name or date.
- Preconditions: None.
- Postconditions: None.
- Scenarios:
  - Apply all the input search filter conditions.
  - Retrieve the app list from the database according to the search filters.

2.2.3.2 Users Log in

- Requirements: Allow a normal user to log in.
- Preconditions: User does not log in.
- Postconditions: User logs in to the system and jumps to the search apps page or the page before the user logs in, such as suggesting app page, editing app tag page and reviewing app page.
- Scenarios:
  - User submits his/her access request by email and password.
  - The System authenticates user request.
  - After authentication, if succeeded, the user enters the system. Otherwise, display an error message and return to the login page.
2.2.3.3 Users Log Out

- Requirements: Allow a normal user to log out.
- Preconditions: User has logged in.
- Postconditions: User logs out the system.
- Scenarios:
  - User submits log out request or user is not active for 15 min.
  - The System terminates the current login session and user logs out the system.

2.2.3.4 Users Forget Password and Reset Password

- Requirements: Allow a normal user to reset his/her password if he/she forgot his/her password.
- Preconditions: None.
- Postconditions: User succeeds to reset his/her password.
- Scenarios:
  - User submits reset password request by providing the email.
  - The System checks whether this email exists in the database. If not, display ‘email not found’ information and ask the user to input the correct email.
  - System emails user reset password link.
  - User resets his/her password by clicking the link.
  - System updates user’s new password in the database.

2.2.3.5 Users Suggest Apps

- Requirements: Allow a normal user to suggest new apps to the system by providing the app download link from iTunes or Google Play Store.
- Preconditions: User has logged in.
• Postconditions: A new app instance is added to the Not Approved Apps category under admin’s app management panel.

• Scenarios:
  o User submits suggesting new app request by providing app download link from iTunes or Google Play Store.
  o The System checks the existence of the app. If the app is already in the database, it will jump to the evaluation report of this app. If not, it will jump to app evaluation page.
  o The User can choose to submit either app link alone or together with his/her reviews about app content and app disability tag information.
  o The System adds this new app instance into the Not Approved Apps category under admin's app management panel.

2.2.3.6 Users Suggest App’s Tags

• Requirements: Allow a normal user to suggest the modification of app disability tags.

• Preconditions: User has logged in.

• Postconditions: A new app instance is added to the Updated Apps category under admin’s app management panel.

• Scenarios:
  o User submits modifying app disability tag request.
  o The System adds that new app instance into the Updated Apps category under admin's app management panel.

2.2.3.7 Post Review

• Requirements: Allow a normal user to post a review.
● Preconditions: User has logged in.

● Postconditions: A new review is added to the database.

● Scenarios:
  ○ User submits the request by providing a review and rating score (1-5).
  ○ The System adds this new review instance into the database.

2.2.3.8 Profile Management

● Requirements: Allow a normal user to edit his/her profile information, first name, last name, country, zip code, occupation, state, school district, school name, grade, password, and email.

● Preconditions: User has logged in.

● Postconditions: The user’s information reflects the new changes.

● Scenarios:
  ○ User submits updating request.
  ○ The System saves the updated information into the database.

2.2.4 Admin Users’ User Case Descriptions

2.2.4.1 Search Apps

● Requirements: Allow users to search apps by the name, app name, app publisher, app price, app objective, app content area, app domain, app audience, and app disability tags and sort the searching result by name or date.

● Preconditions: None.

● Postconditions: None.

● Scenarios:
Apply all the input search filter conditions.

Retrieve the app list from the database by the search filters.

### 2.2.4.2 Users Log in

- **Requirements:** Allow an admin user to log in.
- **Preconditions:** User does not log in.
- **Postconditions:** User logs in to the database and jumps to the admin management dashboard page.
- **Scenarios:**
  - User submits his/her access request by email and password.
  - The System authenticates user request.
  - After authentication, if succeeded, the user enters the system. Otherwise, display the error message and return to the login page.

### 2.2.4.3 Users Log Out

- **Requirements:** Allow an admin user to log out.
- **Preconditions:** User has logged in.
- **Postconditions:** User logs out the system.
- **Scenarios:**
  - Admin user submits logout request, or user is not active for 5 min.
  - The System terminates the current login session and user logs out the system.

### 2.2.4.4 Admins Add Apps

- **Requirements:** Allow an admin user to add new apps to the system by providing the app download link from iTunes or Google Play Store.
- Preconditions: User has logged in.
- Postconditions: A new app instance is added to the Approved Apps or Approved Not Edited Apps category under admin user’s app management panel.
- Scenarios:
  - User submits adding new app request by providing app download link from iTunes or Google Play Store.
  - The System checks the existence of the app. If the app is already in the database, the database will retrieve the existing evaluation information and display app's evaluation form with information filled in. If the app is not in the database, it will show an empty app evaluation form.
  - Admin user can choose to evaluate the app now or only submit basic app information and evaluate it later. Basic information includes app price, app platform, app publisher, app content, app subdomain, app audience, and disability category tag. Evaluation information includes app objective, editor comments, and editor evaluation.
  - If admin user chooses to evaluate app later, the system will add this new app instance into the Approved Not Edited Apps category under admin's app management panel. If the user chooses to evaluate app now, the system will store the evaluation information, and add this new app instance into the Approved Apps category under admin’s app management panel.

2.2.4.5 Apps Management

- Requirements: Allow an admin user to manage the apps in the database.
- Preconditions: User has logged in.
• Postconditions: App’s information reflects the updates.

• Scenarios:
  o If admin user clicks Approved Apps Management tab, the system will retrieve the list of the existing published apps which admin users have finished evaluating. Admin user can modify, delete the app and its evaluation information.
  o If admin user clicks Approved Not Edited Apps Management tab, the system will retrieve the list of the apps, which have not been evaluated. Admin user can delete the app from the list or proceed to assess this app.
  o If admin user clicks Updated Apps Management tab, the system will retrieve the list of the apps that normal users submit modifications. Admin user can delete an app from the list or proceed to check the modification suggestion. And then they can accept or decline that modification.
  o If admin user clicks Not Approved Apps Management tag, the system will retrieve the list of the new apps that normal user suggested. Admin user can delete an app from the list or proceed to check and evaluate this app.

2.2.4.6 User Management

• Requirements: Allow an admin user to manage registered normal users in the system.

• Preconditions: User has logged in.

• Postconditions: None.

• Scenarios:
  o The System retrieves the list of the registered normal users in the system.
  o Admin user can check, modify normal user's information such as first name, last name, country, zip code, occupation, state, school district, school name, and
grade. Admin user can also delete that user from the database.

- The System saves that update.

### 2.2.4.7 User Comments Management

- **Requirements:** Allow an admin user to manage the reviews posted by normal users.
- **Pre-conditions:** User has logged in.
- **Post-conditions:** The reviews have been updated or deleted.
- **Scenarios:**
  - The System retrieves the list of the reviews posted by normal users.
  - Admin user can check, modify and delete normal users’ reviews.
  - The System saves the updated information into the database.

### 2.2.4.8 News Management

- **Requirements:** Allow an admin user to post and edit news in the database.
- **Preconditions:** User has logged in.
- **Postconditions:** The news has been created, updated or deleted.
- **Scenarios:**
  - The System retrieves the list of the published news and unpublished news in the database.
  - Admin user can create new news and mark it as published or unpublished.
  - If admin user chooses to post this news now, the system will save this news under published news category and publish this news on the website.
  - If admin user chooses to post this news later, the system will save this news under unpublished news category.
2.2.4.9 Report

- Requirements: Allow an admin user to track the traffic of the website.
- Preconditions: User has logged in.
- Postconditions: None.
- Scenarios:
  - Admin user submits generating report request.
  - It will take admin user to Google Analytics website.

2.2.5 Non-Functional Requirements

We defined the non-functional requirements for the system in terms of security, speed, usability, maintainability and scalability.

2.2.5.1 Security

The security of the application is very critical as it consists of users’ sensitive information. The application will run inside a firewall. The System will support different roles for users. They will have to enter credentials to log in to the system so that normal users should not be allowed to get access to the admin app management module.

2.2.5.2 Speed

The system should respond within 3 seconds to user’s requests.

2.2.5.3 Usability

All errors will be trapped, and appropriate error messages will be shown.

2.2.5.4 Maintainability

It should be the sustainable, modular code that can be reused and integrated into other
projects if/when necessary.

2.2.5.5 Scalability

The system should be highly scalable for growing number of users trying to register themselves concurrently.

2.3 Summary

In this chapter, we identified the features of system and system functions. We allocate services to system elements. After the first iteration of requirements elicitation, we interviewed some potential users and refined our prototype. Finally, we settled down our requirements analysis. We will cover the design, implementation, and testing in the following chapters.
CHAPTER 3

DESIGN

Software design is the process of analyzing user requirements to design the system architecture and the components of the system. SpedApps system is an interactive system, which is characterized by significant amounts of interactions between humans and computers. The Design is critical in the interactive system and needs to be evaluated at each stage of development. The interactive system design shows that every component depends on each other to serve the purpose of designing and creating of the product. It is a continuous process, and once one component has been partially or completely implemented, we will evaluate it and update our design (Baecker, 2014). In this chapter, we will first discuss a high-level design of system architecture. Then we will describe web interface design and database design.

3.1 System Architecture

As shown in Figure 4, a client's browser passes an HTTP Request, which includes metadata about the browser, user preferences, and any stored cookies for that domain over the Internet to our web server. The server will generate the corresponding query and send it to the database server. After retrieving the data, the web server will process the data and generate an HTTP Response, which is sent back to the client. Once the browser receives the Response, it will begin to parse it for display. Our system runs on Linux with Apache and PHP for web serving. MySQL will be used to connect to the database in the backend.

There are two production servers, and one development server in the system. The
production server is used to host SpedApps applications and database. The development server is designed to help us to implement and test applications. The web server hosts the live application. The database server hosts the database. We deploy and test our application on the development server before pushing the application to the production server.

![System Architecture Diagram]

**Figure 4 System Architecture**

### 3.2 Web Interface Design

User interface design is an important stage for the interactive system. To design and make a great user experience, we worked with UI experts and different levels of potential users who assist us to make design choices for the system.

In general, we would like to create a system with a smart layout of HTML modules. It can help users understand functionalities and purposes of our system easier and more quickly (Galitz, 2007). The homepage is the most critical page for most of the websites. It is the starting point that most users visit. As shown in Figure 5, we start the homepage with a tagline that summarizes what the site does. We put headings such as “Recently Added Apps”, “News” and “Our Apps” in the prominent positions, offering easy access to these contents. For new users, they can understand the purposes of the system and start to use the system quickly. For normal
users, they can also learn the updates of the system just by visiting our homepage, instead of checking the database. The navigation bar is very concise, and it is easier for users to navigate in our system.

![SpedApps Home Page](Image)

**Figure 5 SpedApps Home Page**

Additionally, the design of our interface follows the progressive disclosure principle. Our system only shows what is necessary on each screen. Every screen should have a primary functionality. For example, Figure 6 is the admin dashboard. It shows enough information to allow admin users to make a choice of either user management or app management. Then admin users can work on each subtask in a subsequent screen. As shown in Figure 7, when admin users open the app management page, they can then choose to modify or delete the apps from four categories: Approved Apps, Not Approved Apps, Approved Not Edited Apps and Updated Apps. The system avoids the tendency to over-explain or show everything all at once. It can keep users’ interactions more clear.
3.3 Database Design

3.3.1 Entity-Relationship Diagram

The first step of the database design process is the requirements collection and analysis that we have done in Chapter 2. We interviewed prospective user groups to study and document
Once the requirements have been analyzed, we first need to create a database conceptual schema. The conceptual schema is a brief description of the data requirements of the users (Elmasri, 2016). We are using Entity-Relationship (ER) model to represent this information. The conceptual concepts focus on the requirements instead of implementation, which are easy to be understood. We can use them to communicate with normal clients (Song & Chen, 2009). The data are described as entities, relationships, and attributes in ER model (Elmasri, 2016). An entity is either a physical object or a conceptual object. The features of each entity are called attributes. For instance, an APP entity has three attributes: name, publisher, and price. Additionally, an entity might be weak entity if it cannot be uniquely identified by its attributes alone. For example, when users post COMMENTS, COMMENTS will be a weak entity, which is always related to a specific USER. So we can use user_id and comment_id to identify that comment. There are three kinds of attributes: single-valued attribute, multivalued attribute, and composite value attribute. A composite attribute is composed of multiple attributes. For example, Full Name is a composite attribute, and it can be represented by First Name and Last Name. A multivalued attribute has more than one value. For example, App Tag is a multivalued attribute since an app can have more than one tag. A relationship type is a set of relations among entity types, such as an association MANAGES for two entity types USER and NEWS.

A rectangular box is used to represent an entity type. Attributes are displayed by ovals and are linked to the owner entity types by straight lines. Multivalued attributes are represented as double ovals. Relationship types are displayed in diamonds. Weak entity types are represented as double line rectangular, and the corresponding relationship types are represented as double line diamonds (Elmasri, 2016). The ER diagram of our system is shown in Figure 8.
After we discussed the ER diagram with potential users and made sure it met all of our original requirements, we started to create a relational database schema.

IBM first introduced the relational data model in 1970s (Codd, 1970). A relation schema is composed of a relation name and a set of features. The relation name represents entities or relationships in ER diagram.

Each relational database management system (DBMS) uses a data definition language (DDL). DDL is used to describe the database structure. We choose MySQL, which is one of the most popular open source databases, as our back-end database.

The procedure of mapping an ER to a database schema included four steps, modified from the seven steps mentioned in Elmasri’s book (Elmasri, 2016)). Multiple relation tables first were created based on the regular entity types in ER diagram. Because of the atomic property of attributes in the relational database, if the attribute was a composite attribute, we only list every
single-value attributes. For example, we included *First Name* and *Last Name* instead of *Full Name* in the USER schema. We created the relations APP and USER corresponding to the entity types APP and USER (Figure 8 and Figure 9). We would not consider the foreign keys at this step.

<table>
<thead>
<tr>
<th>APP</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_id</td>
<td>user_id</td>
</tr>
<tr>
<td>app_name</td>
<td>password</td>
</tr>
<tr>
<td>img_src</td>
<td>email</td>
</tr>
<tr>
<td>google_url</td>
<td>firstname</td>
</tr>
<tr>
<td>itunes_url</td>
<td>lastname</td>
</tr>
<tr>
<td>publisher</td>
<td>description</td>
</tr>
<tr>
<td>app_price</td>
<td>school</td>
</tr>
<tr>
<td>app_object</td>
<td>district</td>
</tr>
<tr>
<td>is_approved</td>
<td>grade</td>
</tr>
<tr>
<td>is_evalued</td>
<td>zipcode</td>
</tr>
<tr>
<td>timestamp</td>
<td>country</td>
</tr>
<tr>
<td></td>
<td>occupation</td>
</tr>
<tr>
<td></td>
<td>status</td>
</tr>
<tr>
<td></td>
<td>type</td>
</tr>
<tr>
<td></td>
<td>token</td>
</tr>
</tbody>
</table>

**Figure 9 Results of Mapping Normal Entities: APP and USER**

The second step was mapping weak entity types. The weak entity type always was associated with an owner entity type. We defined a relation table, which included all single-value attributes of the weak entity. The relation table also added the primary key of the associated owner entity as the foreign key. As shown in Figure 10, we created the relation COMMENT, MODIFICATION. *User_id* is the primary key of the owner entity type USER for weak entities COMMENT and MODIFICATION.

<table>
<thead>
<tr>
<th>COMMENT</th>
<th>MODIFICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment_id</td>
<td>modification_id</td>
</tr>
<tr>
<td>app_id</td>
<td>app_id</td>
</tr>
<tr>
<td>user_id</td>
<td>user_id</td>
</tr>
<tr>
<td>time</td>
<td>status</td>
</tr>
<tr>
<td>review</td>
<td></td>
</tr>
<tr>
<td>rating</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 10 Results of Mapping Weak Entities: COMMENT and MODIFICATION**

The third step was mapping Relationship types. There were two types of relationships in this project, 1: N and M: N. Because the 1: N relationship type was the identified relationship for weak entity types COMMENT and MODIFICATION, we will not consider recreating relationships for these two entities again. For the M: N relationship types, we used a relationship relation approach. By using this approach, we designed a separate relation table with the primary keys of two entity types as foreign keys. For instance, we mapped the M: N relationship types.
EVALUATE and MANAGE by creating the relations: EVALUATE, MANAGE_USER,
MANAGE_COMMENT, MANAGE_NEWS, and MANAGE_MODIFICATION (Figure 8 and
Figure 11). EVALUATE relation included the app_id (Primary key of APP) and of user_id
(Primary key of USER) as the foreign keys.

<table>
<thead>
<tr>
<th>EVALUATE</th>
<th>id</th>
<th>user_id</th>
<th>app_id</th>
<th>practice</th>
<th>progress</th>
<th>usability</th>
<th>customization</th>
<th>feedback</th>
<th>summary</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MANAGE_USER</th>
<th>id</th>
<th>user_id</th>
<th>normaluser_id</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MANAGE_COMMENT</th>
<th>id</th>
<th>user_id</th>
<th>comment_id</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MANAGE_MODIFICATION</th>
<th>id</th>
<th>user_id</th>
<th>modification_id</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MANAGE_NEWS</th>
<th>id</th>
<th>user_id</th>
<th>news_id</th>
</tr>
</thead>
</table>

**Figure 11 Results of Mapping Relationship Types**

The last step was mapping multivalued attributes. We created a new relation based on
each multivalued attribute. For example, the APP_CONTENT was a multivalued attribute. We
created a relation APP_CONTENT and the attribute content was the potential values fields
(Figure 8 and Figure 12). Figure 12 shows us the mapping of multivalued attributes
APP_CONTENT, APP_TAG, APP_CONTENT_USER and APP_TAG_USER.

<table>
<thead>
<tr>
<th>APP_CONTENT</th>
<th>id</th>
<th>app_id</th>
<th>content</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>APP_TAG</th>
<th>id</th>
<th>app_id</th>
<th>tag</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>APP_CONTENT_USER</th>
<th>id</th>
<th>app_id</th>
<th>content</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>APP_TAG_USER</th>
<th>id</th>
<th>app_id</th>
<th>tag</th>
</tr>
</thead>
</table>

**Figure 12 Results of Mapping Multivalued Attributes**
3.4 Summary

In chapter three, we discussed the design phase of our project development. We discussed from the system architecture design, which was a somehow high-level view of the system, to low-level views, that included interface design, component design and data structure and database design. After we designed the system, we would discuss how to implement it in the following chapter.
CHAPTER 4

IMPLEMENTATION

In the previous chapters, we discussed the requirements analysis and established a solid architectural foundation for the project. We aimed to design and implement an application called SpedApps, which is used to manage all the evaluation information related to the special educational application. Based on the requirements analysis and design, we implemented the entire system through four main modules: User Account Module, Admin Module, Normal User Module, and App Search Module. Educators, parents and therapeutic professionals can use the system by interacting with those four modules. Some of these modules are very similar to each other and share some common functionality. For example, Admin Module and Normal User Module need functionality to log in and search apps, which are provided by User Account Module and App Search Module. Some similar functionality is used in all applications, but they differ from each other slightly. For example, normal users can suggest and evaluate apps using the rubrics, which are slightly different from admin users. In this chapter, we move to the implementation phase for each module.

4.1 User Account Module

User Account Module handled user account authentication and authorization. User account databases are frequently hacked. SpedApps supports large numbers of users. These users vary widely in the way they interact with the system, such as only registered normal users can suggest new applications, and normal users can not get access to the admin management
dashboard. We have to monitor all of the user requests for resources. Every request must be authenticated and authorized to use the resources. In this section, we will introduce the Salted Password Hashing Algorithms, then discuss how the system handles authentication and authorization for registration tasks, login tasks and account recovery tasks.

4.1.1 Salted Password Hashing Algorithms

Hash algorithms can turn any amount of data into random generating strings based on hash function. However, if two users input the same passwords, they'll have the same hash values. In order to prevent these situations, we can append a random value to the password. Then we can implement hash function on the new password. The random value is called a salt, and the whole process is called Salted Password Hashing. This is ideal for protecting passwords. We can protect user passwords even if the password file itself is compromised. Pseudo code is shown in Figure 13.

```
// $password, user original password
$randomvalue = random();
$random_salt = hash($randomvalue);
$psw = hash($password.$random_salt);
store $psw, $random_salt in the database
```

Figure 13 Pseudo Code of Salted Password Hashing Algorithms

4.1.2 Authentication and Authorization

SpedApps uses email authentication. Users first need to register using their email addresses and create their passwords (Figure 14). The email field should be unique for each user. SpedApps will hash each password with the hashing method mentioned in Section 4.1.1 and store users’ instances in the database. Next time, users should be able to submit login requests by providing their emails and passwords.
Figure 14 Screenshot of User Registration

Figure 15 Screenshot of User Login

*SpedApps* is a multi-page application. To avoid asking users to provide credentials when they are navigating between different pages, sessions are used to store information across page
requests. The session starts when a user first logs in to a system and continues for as long as he/she stays on the site. The System will assign each user a session id and store user type and user status. When the user closes his/her browser or logs off, the session ends. Pseudo code is shown in Figure 16.

```php
//When user first logs in
session_start();
$_SESSION['user_name'] = $rec['user_name'];
$_SESSION['user_type'] = $rec['user_type'];
$_SESSION['user_status'] = $rec['user_status'];

//When admin user navigates through multiple pages
session_start();
if(isset($_SESSION['user_name']) && $_SESSION['user_status']=='Active')
{
    if($_SESSION['user_type'] != 'admin'){
        header("Location: login.php");
    }
} else {
    header("Location: login.php");
    exit();
}
```

Figure 16 Session Code Sample

It often happens that users forget their password and need to recover their accounts. SpedApps can handle password resetting in the following way. When a user makes a password-resetting request via providing his/her email (Figure 17), the system will generate a random token. The token will be stored in the database. Then an email is sent to the user with a special URL containing this token information. When the user clicks this link, they will check the token against the database, and the user can access our system and modify the password.
Parents, educators, and therapeutic professionals, all can access to the App Search Module without logging in to the system. The system categorizes each application through app name, app publisher, app price, app objective, app content area, app domain, app audience, and app disability tags. Users can query the system through cost search, item name search, content search, or an advanced search of any of the filters mentioned (Figure 18). As shown in Figure 19, when users query database with the filter “free”, total 257 applications are retrieved.

To improve users' browsing experiences, we use AJAX (Asynchronous JavaScript And XML) in App Search Module. AJAX allows web search results to be updated asynchronously according to the search filters by exchanging data with the web server instead of reloading the whole page. AJAX allows users to interact with the system in a more convenient way, compared to using the traditional back and forward buttons in the browsers. Sample code is shown in Figure 20.
Select a Search Mode

Q. Type in Your Search

Search by App Name or Publisher Name...

Q. Filter Search

Content: Select options

Price:
- All
- Free
- Paid

Domain: filter apps

Advanced Search Option

Number of Apps Found: 586 | Total Apps in Database: 586

Sort by Name: A-Z  Sort by Name: Z-A  Sort by Newest  Sort by Oldest

Figure 18 Screenshot Of Search App Main Page

SpedApps

Number of Apps Found: 257 | Total Apps in Database: 586

Sort by Name: A-Z  Sort by Name: Z-A  Sort by Newest  Sort by Oldest

Character Story Planner

Average User Rating: ★★★★★

Publisher: Irmantis
Price: Free
Platform: Android
Content: Writing
SubDomain: Planning and Organizing Writing
Audience: Upper Elementary, Middle School, High School
Disability: Specific Learning Disability
Category: Speech or Language Impairment

Objective:
This creative tool addresses character creation and profiling in order to build effective narratives and plots. It provides the pertinent details that should be added and described. Users use a planner chart for sketching their ideas, associating different characters and sharing their creations for feedback.

Editor Comments:
Not yet reviewed

Figure 19 Screenshot Of App Search Result with "Free" as Filter
4.3 Admin Module

The Admin Module provides an interface for admin users to handle all the management procedures related to the database. It includes six functionalities: app search, app evaluation, news management, comment management, apps management and user management (Figure 21).
The app search function will be implemented the same as App Search Module in section 4.2. Admin users can modify, create and delete news about SpedApps and special education through the News management panel. Admin users can manage normal users and check users’ comments to control the spam information through comment management and user management panel. News management, comment management, apps management and user management share many similarities. I will discuss the app evaluation and app management in the following parts.

### 4.3.1 App management

Apps are divided into four categories: Approved Apps, Not Approved Apps, Approved Not Edited Apps and Updated Apps, according to their status in the database. Normal users can suggest new apps to our system, and these apps will be added under Not Approved Apps category. Once admin users review these apps and accept them, these apps will move to Approved Not Edited Apps or Approved Apps depending on whether admin users have performed the evaluation for these apps. The apps added by admin users will also be in either Approved Not Edited Apps or Approved Apps Category. Users can also suggest modifications for current existing apps, and this information will be added under Updated Apps. We implement these four categories by using tabs. The contents for each tab panel are a list of apps. Users can edit and delete apps in the list. (Figure 7 and Figure 22). There are two ways to implement the list, in-page or AJAX. As mentioned in section 4.2, AJAX allows web search results to be updated asynchronously instead of reloading the whole page. We define the list via AJAX. The core code is shown in Figure 23.
### Approved Not Edit

<table>
<thead>
<tr>
<th>App Name</th>
<th>Publisher Name</th>
<th>Price</th>
<th>Edit</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Story Planner - Outline your Novel &amp; Screenplay</td>
<td>SCVisuals</td>
<td>$3.99</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Character Story Planner</td>
<td>Ifantasia</td>
<td>Free</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>LitCharts</td>
<td>LitCharts LLC</td>
<td>Free</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>iDeas for Writing</td>
<td>SCVisuals</td>
<td>$2.99</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Pocket: Save Articles and Videos to View Later</td>
<td>Read It Later, Inc</td>
<td>Free</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Poetry Creator</td>
<td>Verses - Poetry, Poems &amp; Poets</td>
<td>Tiny Mobile Inc.</td>
<td>Free</td>
<td>✔</td>
</tr>
<tr>
<td>Same Sound Spell Bound (Homophones)</td>
<td>NRCC Games</td>
<td>$0.99</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Same Meaning Magic (Synonyms)</td>
<td>NRCC Games</td>
<td>$0.99</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Figure 22** Screenshot Of Approved Not Edited Apps List
<ul class="nav navbar-nav">
    <li>
        <a href="#Approved" role="tab" data-toggle="tab">
            Approved Apps <span class="label label-success badge">?</span></a>
    </li>
    </ul>
    </div>
    </div>
</div>
</div>
</div>
</div>
</div>
</ul>

Not Approved Apps <span class="label label-danger badge">?</span></span>
    </a>
    </li>
</div>
</div>
</ul>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
4.3.2 App Evaluation

As shown in Figure 24, the first step of adding an app to the database is to check the existence of this app. The system handles this task by checking an iTunes URL or a Google Play URL in the APP table in the database. If it is a new app, the system will display a new app evaluation form. There are two sections in an app evaluation form: app identity and app evaluation details. To facilitate admin work, we implement a crawler function, which can scan the web and retrieve the contents, such as app name, publisher, price, and app image automatically. Parts of the pseudo code and an app evaluation form are shown in Figure 25, Figure 26.
Figure 24 Screenshot Of Checking the existence of Apps

```html
<script>
  function iTunes($link) {
    $content = file_get_contents($link);
    $dom = new DOMDocument();
    $dom->loadHTML($content);
    $title = $dom->getElementsByTagName('title')[0].nodeValue;
    $publisher = $dom->getElementsByTagName('publisher')[0].nodeValue;
    $description = $dom->getElementsByTagName('description')[0].nodeValue;
    $img_url = $dom->getElementsByTagName('img')[0].getAttribute('src');
    $price = $dom->getElementsByTagName('price')[0].nodeValue;
    return $meta;
  }
</script>
```

Figure 25 Sample Code of App Crawler Function
Figure 26 Screenshot of App Evaluation Form

4.4 User Module

The User Module provides an interface for normal users to suggest and modify the existing apps in the database. It includes three functionalities: app search, app suggestion and app modification (Figure 27). The app search function will be implemented the same as the App Search Module in section 4.2. The app suggestion function is similar to the admin app adding function. The app evaluation form for normal users only includes an app identity part. Normal
users can choose to fill either part or all of it. Users can suggest modifications of app disability tags or review apps at the app main page as shown in Figure 28 and Figure 29.

Figure 28 Screenshot of App Main Page

Figure 29 Screenshot of Writing a Review
4.5 Summary

In this chapter, we discussed how to achieve the functionalities of the system through four main modules: the User Account Module, the Admin Module, the Normal User Module and the App Search Module. We discussed the security strategy such as Salt Hash Algorithms and password recovery methods. We also discussed ways to optimize operating and browsing speed of the system. After we built the system, we moved to testing phase in the following chapter.
CHAPTER 5

TESTING

Verification and validation are the phases of software development life cycle when we check whether the system achieves the specifications and the target purposes. There are several ways to verify a software system. We will use testing to fulfill this task in our project. In this chapter, we will first discuss the importance of testing, and then we will discuss our testing plan and testing result.

5.1 The Importance of Software Testing

Testing is the most popular approach for system verification. The aim of testing is not only at finding whether systems achieve the specifications, but also at finding errors. The main advantage about testing is that it does not generate false positive. Software testing helps us identify and fix bugs before the software launched, and the risk of failure can be reduced.

The main limitation of testing is that it is highly incomplete. It is not possible to test every case when users use our system. We can only perform testing under very small-optimized subsets of all the possible combinations (Myers, Sandler, & Badgett, 2011). Goodenough and Gerhart in their paper said a test is successful if the program fails (Goodenough & Gerhart, 1975).
5.2 Testing Plan

5.2.1 Website Display Test

To test the website display effects, we test our system in four kinds of the web browser: IE, Google Chrome, Firefox, and Safari. We also preview web pages in a variety of predefined screen resolutions including desktops, tablets, and mobile phones. After typing the URL (http://spedapps.kent.edu/) in the address bar, the homepage should be able to load correctly. We need to consider the following cases:

- To check whether the web browser displays the web pages with some errors, such as page display errors and messy codes, etc.
- To check the function of the navigation bar. When we click navigation button, it should jump to the corresponding page correctly.
- To check the image display. Due to the image size and resolution, the image might not be able to display properly. We need to modify the CSS codes to solve this problem.
- To check the font format, font size. The errors are often caused when HTML or CSS compiled. We need to make sure the font size is consistent with our original UI design.

After testing, the system displayed correctly.

5.2.2 Website Accessibility Test

An accessible website is a website that can be used by individuals with disabilities such as blindness, deafness, etc. Accessibility is important for a website, especially for our website which is targeting at apps in the special educational area. We test our website using the wave tool (http://wave.webaim.org/). The system passed the test, and it is accessible.
5.2.3 Function testing

To test the function of our system, we need to consider the following cases:

- Registration function.
- Authentication function and Rest Password function when user forgot his/ her original password.
- App search function.
- Normal users suggest apps and modify Apps evaluation function.
- Admin management module function including news management, app management, user management, user comments management. The admin should be able to edit, delete and change the status of all of the above information.

The system passed all of the test cases, and it works correctly.

5.3 Summary

In chapter five, we discussed the importance of testing and the test plans. SpedApps satisfied all of the user requirements and passed all of the test cases. It can be released to the final users. After the software was released, the clients might request new requirements, the system environment might change, and users might report the potential bugs. We will perform maintenance activities for SpedApps and release a new version of the application after some maintenance activities. The cycle will continue throughout the lifetime of the software.
CHAPTER 6

CONCLUSION

In this thesis, we presented a searchable, online educational applications database management system called SpedApps. SpedApps was developed according to the following principles: Information Science design paradigm, Software Engineering development process, Computer-Human Interaction Guidelines and Education guidelines. It is a well-designed object-oriented application. The database currently contains more than 580 apps related to special educational area. It categorizes each app by the app name, publisher, price, objective, content area, domain, audience, and disability tags. Most of the app evaluation reports contain an objective review, evaluating whether the app provides the information such as: practice, feedback, progress, usability, and customization. Users can query the database through different filters such as: app keyword search, app name search, cost, topic search, etc.

SpedApps has made a significant impact on the daily life of parents, educators, and therapeutic professionals. It improves the efficiency of apps selections and the outcomes of using mobile applications that are designed with the special needs.

Although SpedApps is a customized information management system for the special educational area, it is implemented in a generic way. The frame can be used not only for special education but also for any other subjects and can be scaled to other domains as well in the future.
REFERENCES


Richard E. Ferdig, K. E. P., Karl W. Kosko,Enrico Gandolfi,Rachel Mathews. (2016). *Use and perceptions of mobile applications and technologies by those interested in special*
education.

