**Software Requirements Specification**

**for**

Student Portal

**Version 1.0 approved**

**Prepared by Yang Ding, Peiyan Duan, Xiaoying Ji,**

**Duan Li, Tianlin Lu, Chris Tsuei**

**EECS393**

**Sep 26, 2016**

**Table of Contents**

**Table of Contents**

**Revision History**

**1. Introduction**

1.1 Purpose

1.2 Intended Audience and Reading Suggestions

1.3 Product Scope

1.4 References

**2. Overall Description**

2.1 Product Perspective

2.2 Product Functions

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 Assumptions and Dependencies

**3. External Interface Requirements**

3.1 User Interfaces

3.2 Software and Communication Interfaces

**4. System Features**

4.1 Homepage Requirements

4.2 Exchange Requirements

4.3 Events Requirements

**5. Other Nonfunctional Requirements**

5.1 Performance Requirements

5.2 Security Requirements

5.3 Software Quality Attributes

**Appendix A: Glossary**

**Appendix B: Analysis Models**

**Appendix C: Figure and Table Lists**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| SRS for Student Portal | 9-26-16 | Initial Release | v1.0 |
|  |  |  |  |

# 1. **Introduction**

## 1.1 **Purpose**

The purpose of this software requirement specification document is to describe the software behavior and functionalities of the Student Portal system. The Student Portal is designed to be an application on the Android operating system that provides a platform for posting and joining on-campus events, as well as exchanging second-hand items. This document is intended to serve as a guideline and to state the intended goals for the implementation of the various functions of the program. Additionally, the purpose of this document is to help both the client and the developers to understand the project in details, and minimize any mutual misunderstanding that may exist between them.

## 1.2 **Intended Audience and Reading Suggestions**

This document is intended for team members including developers, testers, documentation writers, and all interested stakeholders. The demonstration application will be tailored for the students of Case Western Reserve University. However, this application can be applied for students of other colleges.

Readers interested in a brief overview of the product should focus on the rest of Section 1: Introduction, as well as Section 2: Overall Description, which provide a brief overview of each aspect of the project as a whole.

Readers who wish to explore the features of Student Portal in more detail should read on to Section 4: System Features, which includes functional requirements and use-case reports of the product. Section 3: External Interface Requirements offers further technical details, including information on the user interface as well as the software and communication interfaces on which the application will run.

Readers interested in the nonfunctional requirements of the project should read Section 5, which covers performance, security, and software quality attributes that will be important to users. Readers who have trouble understanding the jargon or acronym mentioned in this document should check Appendix A: Glossary. Readers who are curious about the dynamic aspect of the product should check Appendix B: Analysis Model, which includes a flowchart to represent the operation of the product. In addition, Appendix C: Figure and Table Lists summarizes all the visual aids used in this document.

## 1.3 **Product Scope**

## The student portal will be used primarily by college students in order to better interact with the community as a whole. This mobile application will better allow users to participate in activities on campus and to be informed of upcoming events. It will also simplify the process of exchanging second-hand items on campus in hope that users will be more willing to share their idle resources. The events calendar will use the Google Calendar© API. Event location navigation service will be based on Google Maps©. If needed, users can request a ride using Uber, which will be integrated into the application using the Uber© API.

The complexity of the product will reside in the featured event/item recommendation algorithm and QR code implementation. Recommendation algorithms should take user’s interests and search history as input, and generate a list of recommended events or items. Collaborative filtering or cluster models might be used to solve recommendation problem. In addition, the product has the ability to generate QR code for each user as unique identification as well as scanning barcodes to enable event organizers grant access to their events. .

**1.4 References**

[1] Amazon Web Services. (2016). *Amazon EC2 - Virtual Server Hosting* [online]. Available: <https://aws.amazon.com/ec2/?nc1=h_ls>

[2] Christensson, P. (2016, May 16). *Android Definition* [online]. Available: <http://techterms.com/definition/android>

[3] Christensson, P. (2016, June 20). *API Definition* [online]. Available: <http://techterms.com/definition/api>

[4] Wikipedia contributors. (2016, September 15). *Cluster Analysis* [online]. Available: <https://en.wikipedia.org/wiki/Cluster_analysis>

[5] Wikipedia contributors. (2016, September 22). *Collaborative Filtering* [online]. Available: <https://en.wikipedia.org/wiki/Collaborative_filtering>

[6] Christensson, P. (2015, January 30). *FTP Definition* [online]. Available: <http://techterms.com/definition/ftp>

[7] Wikipedia contributors. (2016, September 21). *Hashtag* [online]. Available: <https://en.wikipedia.org/wiki/Hashtag>

[8] Christensson, P. (2015, May 28). *HTTP Definition* [online]. Available: <http://techterms.com/definition/http>

[9] Margaret, R. (2012, June). *OAuth* [online]. Available: <http://searchsoa.techtarget.com/definition/OAuth>

[10] Christensson, P. (2015, March 5). *QR Code Definition* [online]. Available: <http://techterms.com/definition/qr_code>

[11] Christensson, P. (2006). *XML Definition* [online]. Available: <http://techterms.com/definition/xml>

## 2. O**verall Description**

## 2.1 **Product Perspective**

The Student Portal aims to create a community for college students by including features such as an event information calendar and a secondhand exchange. Our team came up with this idea because we discovered that a secondhand exchange and an events calendar are two fundamental needs for college students, as they need a platform to get first-hand event information as well as ways to exchange goods, such as books and school supplies, from time to time. However, there is not such a unified, community-based application available on Google Play app store for a school community that satisfies both needs in one application. Hence, this Student Portal is a new, self-contained product that fulfills functionalities mentioned above.

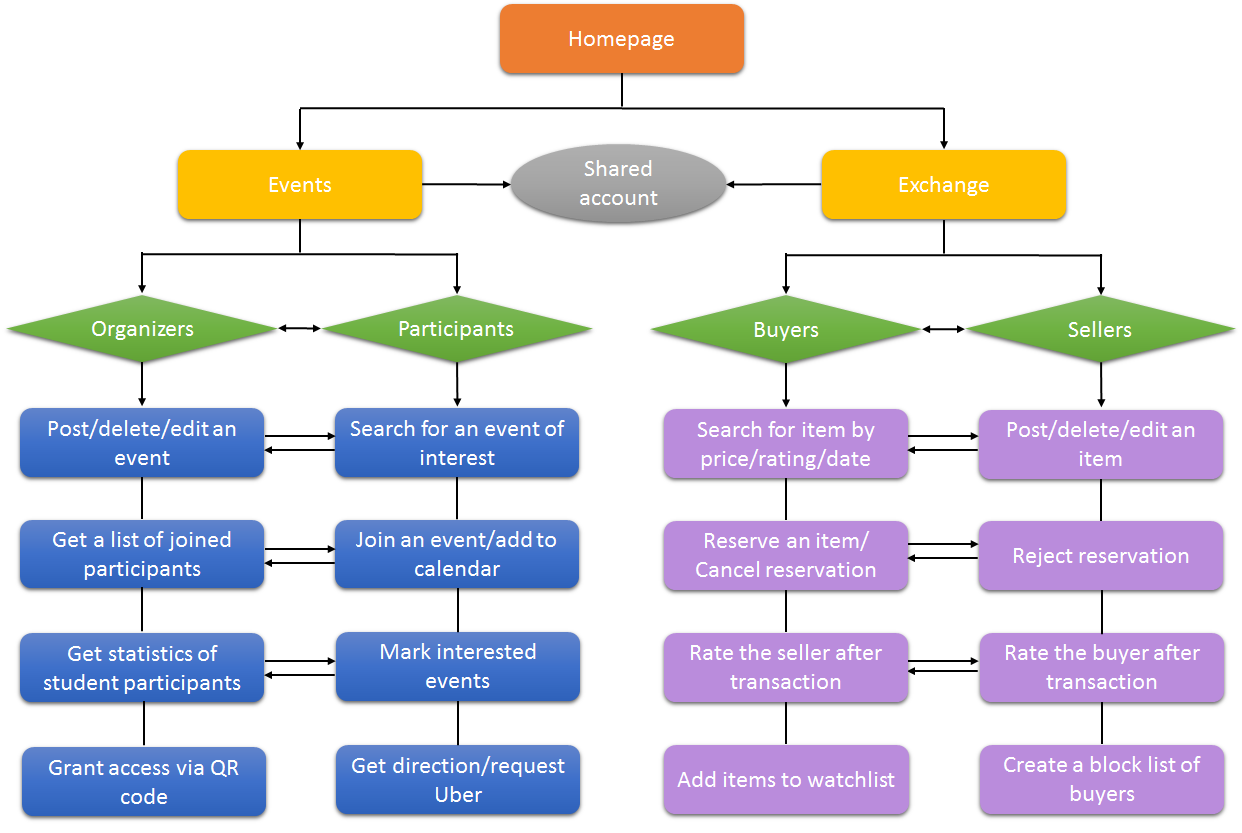
The major components of this system will be a campus event section and a secondhand exchange section. The functionalities of these two sections will be described in details in the section of System Features.

## 2.2 Product Functions

Upon the first login, users will be able to create an account, obtain a QR identification code, and manage their personal information on their profile.

For the event information platform part, the student participants will be able to receive filtered on-campus event information, manage their event calendar and mark interested events. The organizer can scan the participant’s QR code in order to grant access to a certain event. The organizers will also be able to advertise their event information, update event information and get a feedback of students who are going or interested in their event. Data about popular events and organizations will be collected.

With regard to the exchange section, the student sellers will be able to post an item for sale with information such as condition, price, time used and available date, and they will also receive a message if their item has been sold or reserved. The student buyers will be able to search for the items they want based on their indicated interests. They will also be able to acquire the contact information of the seller to make an offer or get first-hand update on the item of interest, for instance, if a book has been retrieved or sold. There will also be a watchlist for buyers to get updates of the items that are temporarily reserved, but might be potentially available in the future. On the seller’s side, they can post, delete or edit items they wanted to sell. They can also reject a reservation if they doesn’t want to sell this product to a certain buyer. We will also extend functions such as a block list or rating system after an item is sold. Our external entities of the Student Portal are illustrated through the context diagram in Figure 1.



*Figure 1: External Entity of the Student Portal*

## 2.3 **User Classes and Characteristics**

|  |  |
| --- | --- |
| Participants(Events) | An event participant is a college student who browses featured events from the Student Portal on a daily basis and occasionally joins events posted. All participants using this application are assumed to have an Android device. Most participants will wish to look up events by their interests, so search functionality needs to be present. Also, participants can be expected to cancel joined events from their calendar. |
| Organizers(Events) | An event organizer is an executive member of an on-campus organization or an individual who wishes to publish a small scaled event. Organizers can be expected to edit an posted event and acquire statistics of participants of their events. Organizers may also wish to grant access using a QR code at time of the event. |
| Sellers(Exchange) | A seller is a college student who wishes to trade his/her items with classmates using the Student Portal. Other than posting, deleting and editing an item, the seller can also be expected to accept or decline a reservation autonomously. After an item is sold, the seller may want to resolve the post and rate the buyer so that other participants in the system will be informed. |
| Buyers(Exchange) | A buyer is a college student who wishes to find the items s/he wants by browsing the items listed by his/her classmates using the Student Portal. If the buyer finds the item that s/he wishes to buy, s/he can reserve the item. The buyer can also add the item to the watchlist if s/he is interested in the item or the item is temporarily unavailable (e.g. reserved) for now. The buyer in the watchlist of a certain item will be able to get update notifications of that item. |

*Table 1: User Classes and Characteristics*

## 2.4 **Operating Environment**

OE-1: The Student Portal System should operate on Android 6.0 Marshmallow.

OE-2: The Student Portal System should operate on a server running the current approved versions of Amazon EC2. We might change the server system in our future implementation.

## 2.5 **Design and Implementation Constraints**

CO-1: The application should accommodate approximately 200 users during the peak usage time when the semester begins.

CO-2: All our scripts are written in Android Studio.

CO-3: The application will use an Android device to test our application.

CO-4: The application will follow the Google Java Style which can be found on <https://google.github.io/styleguide/javaguide.html>

CO-5: The communication protocol will be HTTP(s).

CO-6: The user information should be kept private from other users and administrators.

## 2.6 **Assumptions and Dependencies**

AS-1: We are assuming that the users of our prototype application are all college students, using their campus network ID to register an account.

AS-2: Exchange buyers and sellers will make monetary transactions within a reasonable time determined by both parties once the item is reserved. The application will not provide an online payment service.

AS-3: Users who post an event will only post events that will actually occur and those posts will not contain malicious link.

AS-4: Students who joined an event and wish not to attend will deselect the “join” button for an event on time.

AS-5: Exchange sellers will change the status to sold once the transaction is complete.

AS-6: Database has a large enough capacity for all the information we need to store.

DE-1: We will request permission to access or modify a user’s online calendar.

DE-2: We will request permission to access the user’s current location.

# 3. **External Interface Requirements**

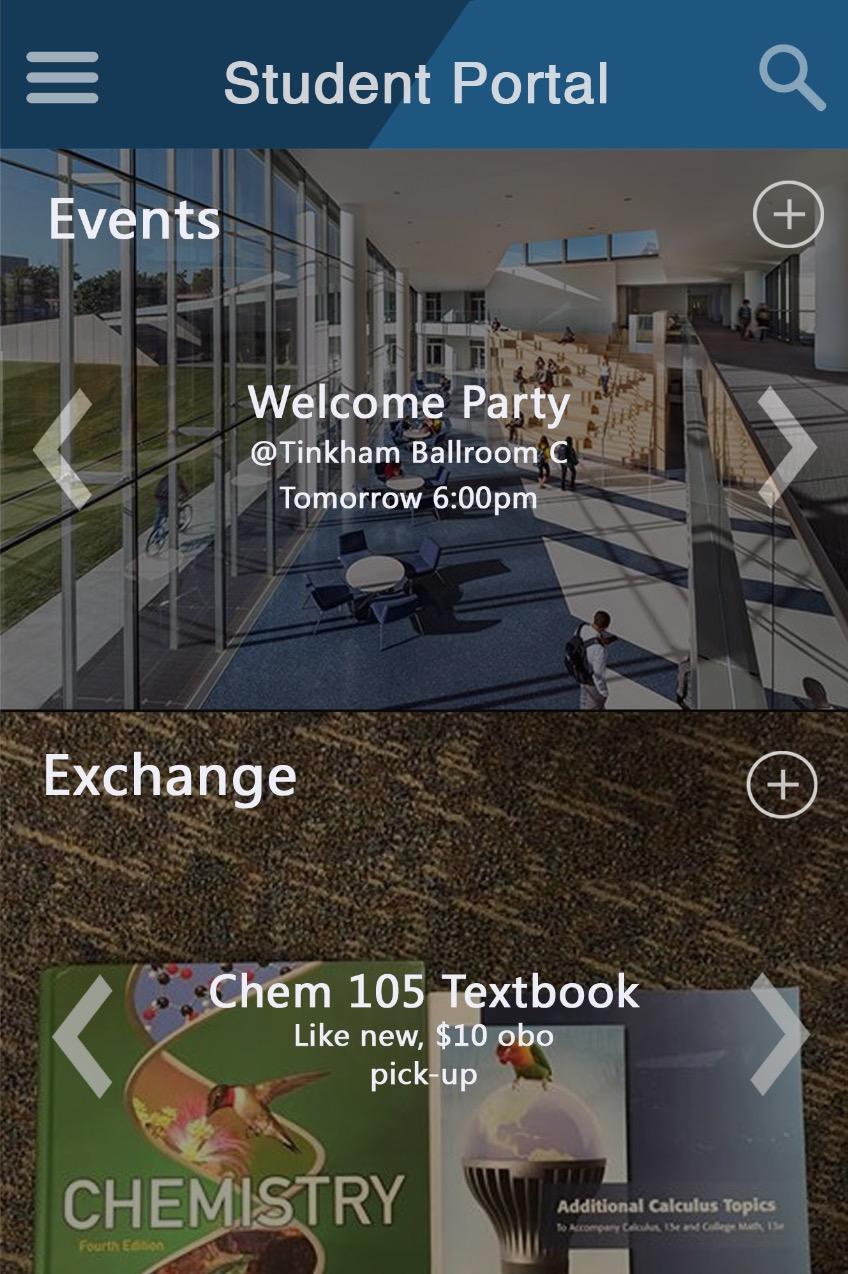
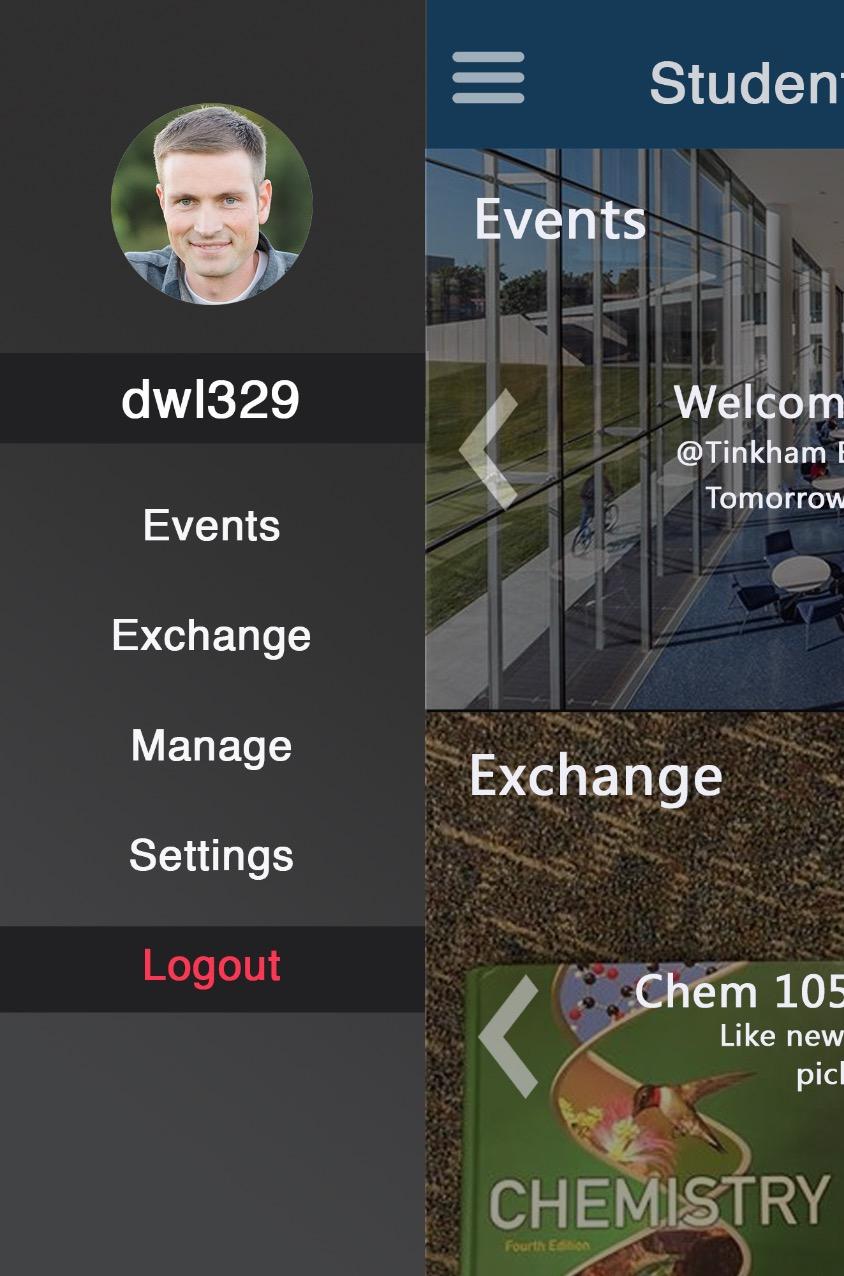
## 3.1 **User Interfaces**

**Homepage**

As shown in Figure 2, after opening the Student Portal application, the student will have to log in with their campus email. Once a user has logged in or created an account with a valid campus email account and password, that user can access their personalized campus events calendar which includes the option to import their own calendar and add those events to their portal calendar. These users can create events as well as post and interact with listings on the secondhand exchange. Users that have logged in also can access a profile page where they can edit their profile picture, the portal background image as well as manage event subscriptions.

**Sliding menu**

As shown in Figure 2, using a sliding menu on the upper left hand corner of the application, users can access their user profile, view the public events and exchange pages, as well as the manage menu, and log out of their account.

*Figure 2: Prototype sample view of the homepage & Sliding Menu*

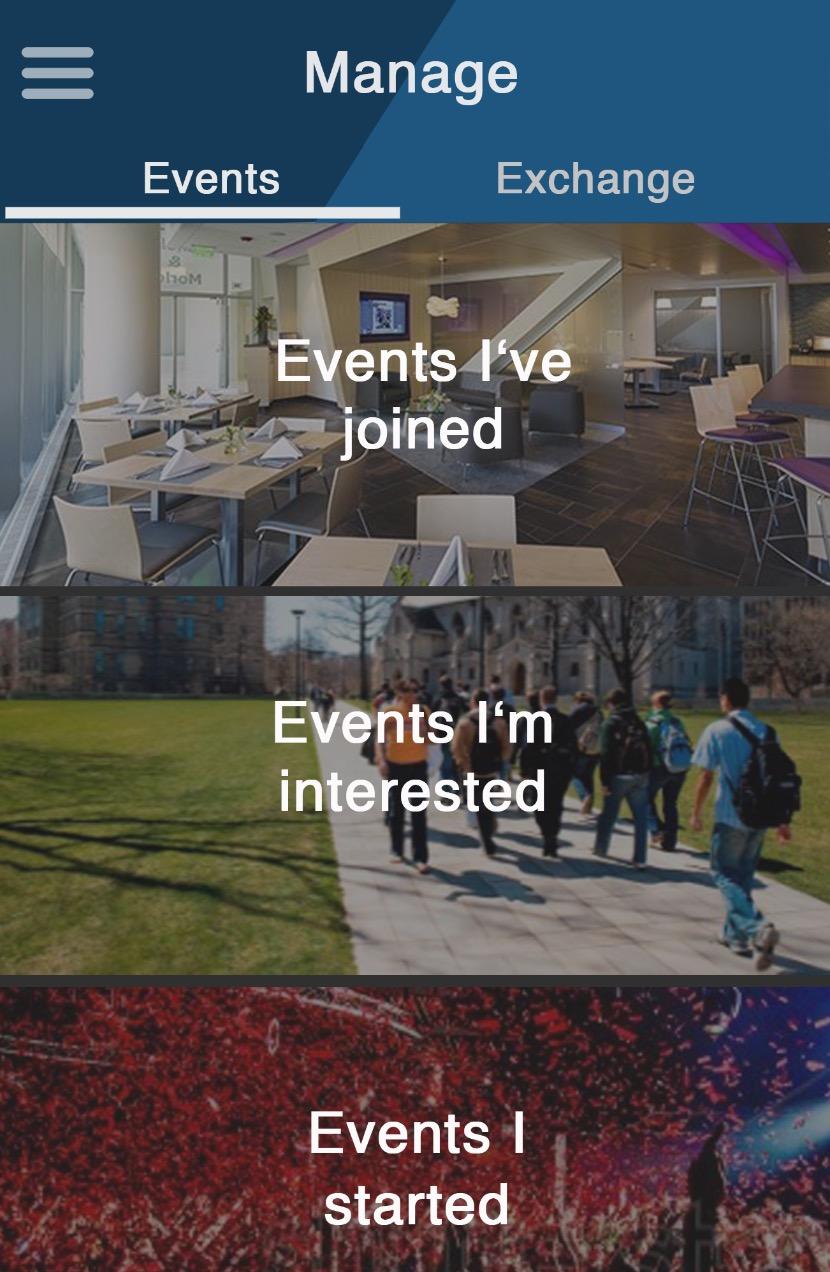
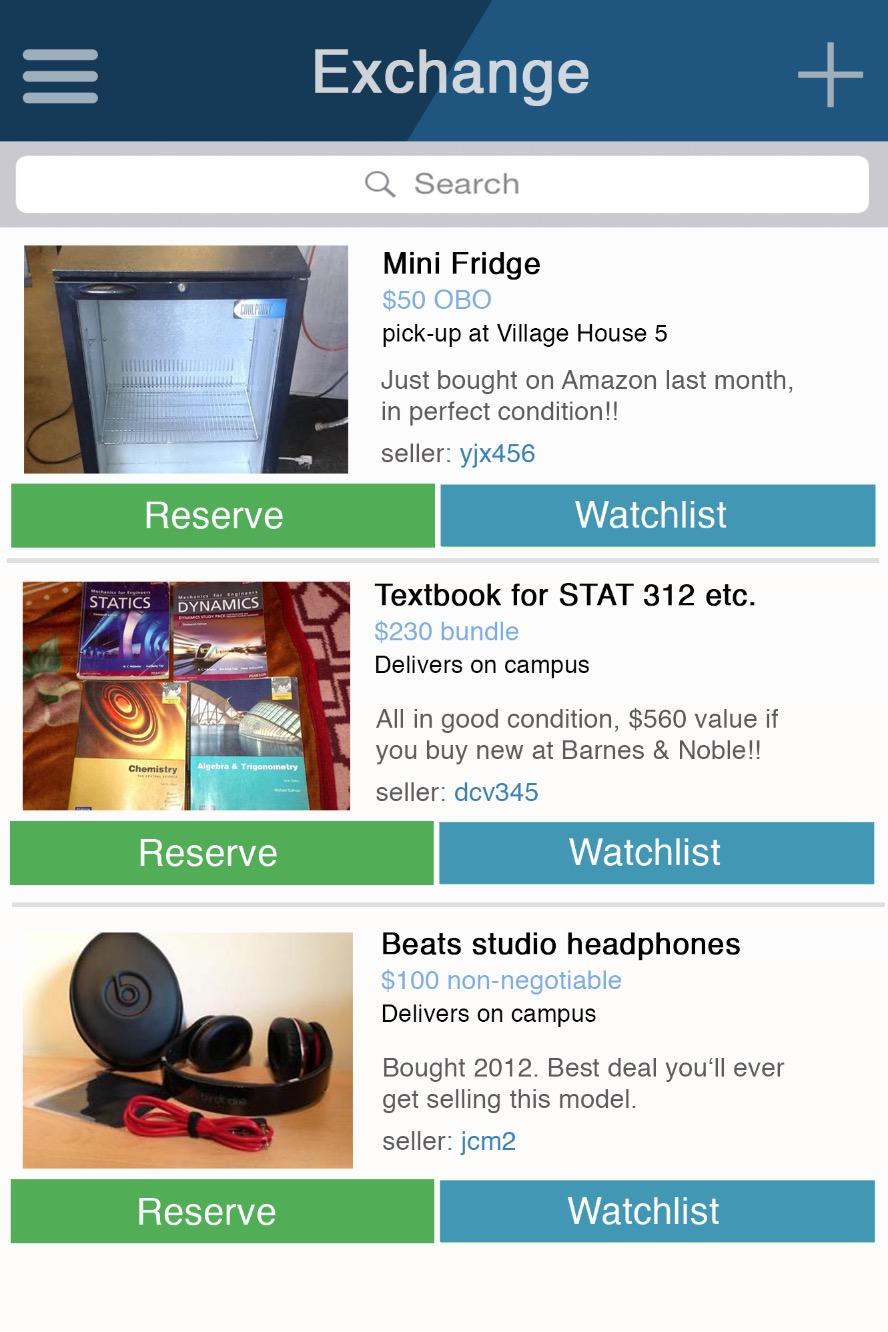
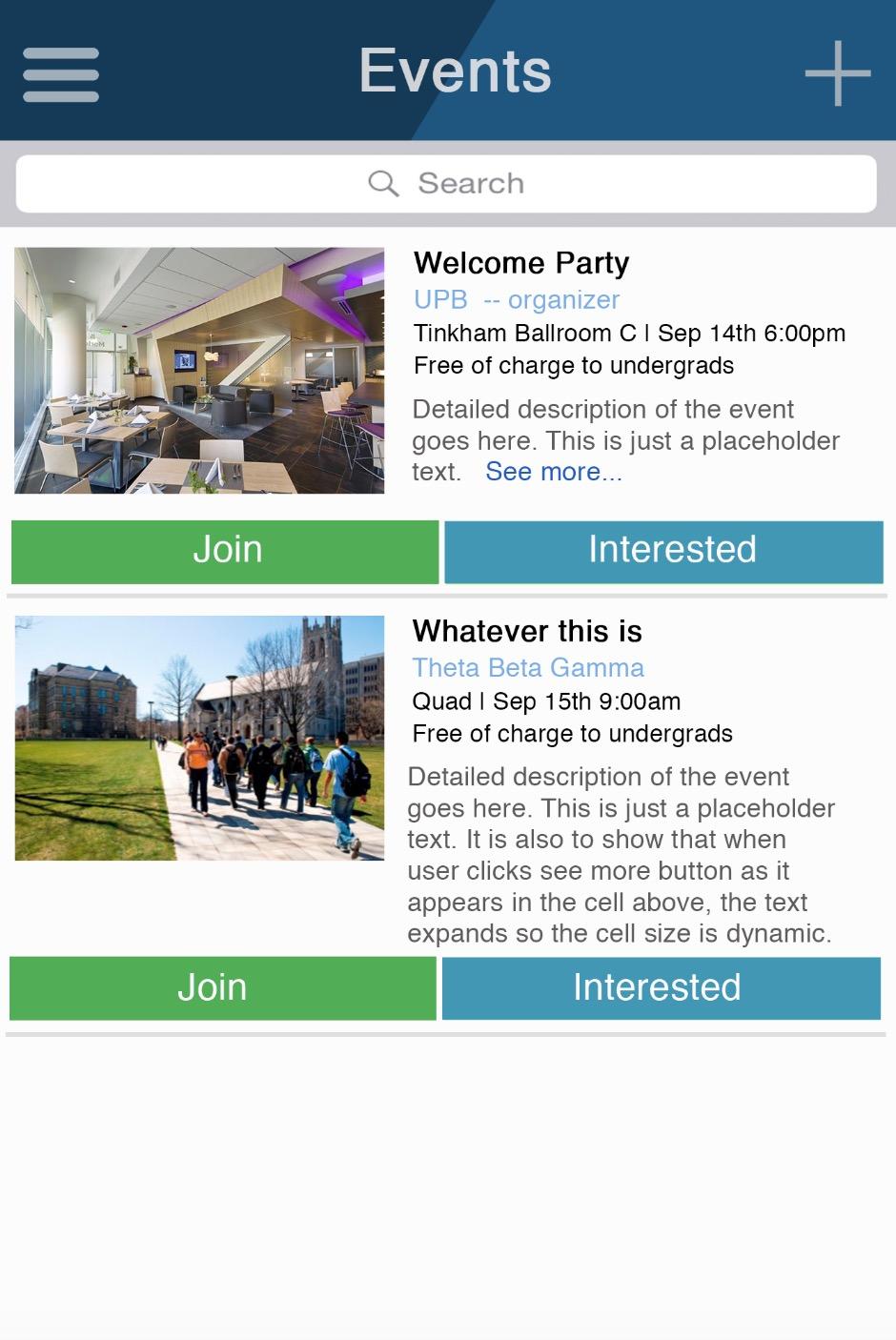
**Events and Exchange**

As shown in Figure 3, the event platform works as a social media for students and clubs to post and exchange campus-wide event information and serves to provide better communication between event organizers and potential participants. As shown in Figure 3, the exchange section is a platform where students can post their second-hand or unneeded items for sale. For instance, a student would be able to search for a book, buy it or exchange it for another book that the other student is selling. Books will categorized for different subjects and will have information of condition, edition, picture of cover page and location. All data will be stored in an online database.

Both the events and exchange pages have search bars on the top of their respective pages as well as buttons on each listing to indicate interest. Users can also create new exchange entries or events in their respective pages by touching the + sign in the upper right corner, which will allow the user to add new items for other users to view.

**Manage Menu**

As shown in Figure 3, the manage menu will allow a user to manage the events and exchange items they are interested as well as view the events and items they are putting on the exchange for other users. It is split into an events and exchange submenus, with the default screen being the events menu. Each submenu will separate the user’s events and listings into interested, participate (for events) and offer (for exchange), as well as the user’s created entries on the portal.



*Figure 3: Prototype sample view of the Events Page, Exchange page and Manage page.*

## 3.2 **Software a**nd Communication **Interfaces**

This portal will interface with an online database in order to access the user profile, events and exchange databases. Importing events will use the Google Calendar API. Forms to store the user entries will be using some sort of forms API (Google forms) XML for entries and FileStream for files.

Most network communication will be done using the HTTP(s) protocol. Images as well as event schedules will be done using FTP protocol. Log in will be managed using a secure SSL encryption.

# 4. **System Features**

## 4.1 Homepage Requirements

4.1.1 Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Code** | **Description** | **Priority** |
| FR-1 | The user should be able to login with case.edu account. His/her account default username is automatically set to corresponding Case network ID. | High |
| FR-2 | Account usernames must be unique. | High |
| FR-3 | The user should be able to change the username provided it is unique. | High |
| FR-4 | After the login, the homepage should display the sliding menu button on the upper left corner, which can navigate to the Events, Exchange, Manage and Settings page. | High |
| FR-5 | Featured events will be displayed on the in the upper Events section. A plus button will allow user to quickly create a new event. | Medium |
| FR-6 | The Events section will consist of current event’s name, location and time, overlaid on event’s cover image. | High |
| FR-7 | The Events section will also consist of a left and right arrow to allow users to browse other featured events. | Medium |
| FR-8 | Items for exchange will be displayed in the Exchange section in the lower half of the home screen. A plus button will allow user to quickly add a new item for exchange. | Medium |
| FR-9 | The Exchange section will consist of current item’s name, price and a brief description, overlaid on the item’s picture. | High |
| FR-10 | The Exchange section will also consist of a left and right arrow to allow users to browse other items on sale. | Medium |

*Table 2: Homepage Functional Requirements*

4.1.2 Use-Case Reports

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 01 | |
| **Use Case Name:** | Register | |
| **Actors:** | | General User (Any User) |
| **Description:** | | This use case allows a user to register for a new account. |
| **Frequency of Use:** | | Once Per User |
| **Preconditions:** | | The user does not have an account. |
| **Postconditions:** | | The user has an account. |
| **Normal Flow:** | | 1. The user visits the register page. 2. The user provides a valid .edu E­mail address. 3. The user creates a valid password (at least 8 digits). 4. The user provides contact information (Optional phone number). 5. The user clicks on the register button. |
| **Exceptions:** | | 1. E­mail address is invalid. 2. Password is invalid. 3. Account already exists. |
| **Includes:** | | Valid Email Address Verification Valid Password Match Verification Contact Information |
| **Special Requirements:** | | The administrator should not be able to collect user campus ID and password during this process. |
| **Assumptions:** | | The user is a college student with standardized campus ID and campus email address. |

*Table 3: Homepage User-Case: Register*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 02 | |
| **Use Case Name:** | Sign In | |
| **Actors:** | | General User (Any User) |
| **Description:** | | This use case allows a user to sign in with his/her registered accounts. |
| **Frequency of Use:** | | Every time a user switches accounts or signs in to a new device. |
| **Preconditions:** | | The user was logged out |
| **Postconditions:** | | The user is signed in. |
| **Normal Flow:** | | 1. The user visits the sign-in page. 2. The user enters the valid .edu E­mail address of the account or the username. 3. The user provides a valid password. 4. The user clicks on sign in button. |
| **Exceptions:** | | 1. E­mail address is invalid. 2. Password is invalid. |
| **Includes:** | | Account Match Algorithm |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | None |

*Table 4: Homepage User-Case: Sign In*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 03 | |
| **Use Case Name:** | Log out | |
| **Actors:** | | Logged in User |
| **Description:** | | This use case allows a user to logout his/her account |
| **Frequency of Use:** | | Every time user wants to log out |
| **Preconditions:** | | The user was signed in |
| **Postconditions:** | | The user is logged out |
| **Normal Flow:** | | The user clicks on logout button |
| **Exceptions:** | | The user is creating new event or new items for sale. |
| **Includes:** | | None |
| **Special Requirements:** | | None |
| **Assumptions:** | | User has internet access. |
| **Notes and Issues:** | | None |

*Table 5: Homepage User-Case: Log out*

## 4.2 Exchange Requirements

4.2.1 Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Code** | **Description** | **Priority** |
| FR-10 | The user should be able to add items for sale. The system should ask the user for item names, description, photos, price, and personal contact information. The status of the item is available automatically. | High |
| FR-11 | The user (seller) should be able to edit or update item information. The system will send notifications to other users who have reserved the item. | High |
| FR-12 | The user (seller) should be able to delete items for sale. The system will send the notifications to other users who have reserved the item. | High |
| FR-13 | The user (seller) should be able to receive notifications from the system automatically if the item posted is reserved. | High |
| FR-14 | The users (seller) should be able to reject a reservation from other users (buyers). The status of the item will change from reserved to available. | High |
| FR-15 | The user (seller) should be able to add certain users to a block list. The user (buyer) on the seller’s block list cannot reserve any items sold by the seller. | Low |
| FR-16 | The user (seller) should be able to change the item status to sold after the transaction is completed. The system does not provide online payment support and will not run queries on sold items when user is searching. | High |
| FR-17 | The user (seller) should be able to rate the user (buyer) when the item is sold. The system will ask for a score from 0 to 5 stars in a pop-up window when the status of item becomes sold. | Medium |
| FR-18 | The user should be able to see a list of items in the exchange homepage without searching. The list is sorted by the system according to the previous searching and buying records. If the user is a first time to use the system, user, the list will display the latest items posted by other users to sell. | High |
| FR-19 | The user (buyer) should be able to search an item by keywords and check all the available and reserved items related to the keyword. | High |
| FR-20 | The user (buyer) should be able to filter the search results by price or time. | High |
| FR-21 | The user (buyer) should be able to filter the search results by seller rate. | Medium |
| FR-22 | The user (buyer) should be able to reserve an available item. The status of item will change from available to reserved. | High |
| FR-23 | The user (buyer) should be able to cancel the reservation on items. The status of the item will be changed from reserved to available. | High |
| FR-24 | The user (buyer) should be able to add a reserved or an available item to their watch list. The status of the item will not change. | Medium |
| FR-25 | The user (buyer) should be able to receive notifications from the system automatically if detailed information or status of the item reserved has been changed. | High |
| FR-26 | The user (buyer) should be able to receive notifications from the system automatically if detailed information or status of the item in watchlist has been changed. | Medium |
| FR-27 | The user (buyer) should be able to rate the user (seller) when the item is bought. The system will ask for a score from 0 to 5 stars in a pop-up window when the status of the item is changed by user (seller) to sold. | Medium |

*Table 6: Exchange Functional Requirements*

4.2.2 Use-Case Reports

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 04 | |
| **Use Case Name:** | Search Items | |
| **Actors:** | | Users as buyers |
| **Description:** | | This use case allows a user (buyer) to search for items he/she wants to buy with keywords. |
| **Frequency of Use:** | | Every time a user (buyer) wants to buy an item. |
| **Preconditions:** | | The user (buyer) is on Exchange page |
| **Postconditions:** | | The user (buyer) has a list of results displayed in the page. |
| **Normal Flow:** | | 1. The user (buyer) clicks on the search bar. 2. The user (buyer) provides a keyword. 3. The user (buyer) selects sort with price/time/seller rate option (not required) 4. The user (buyer) clicks on search response. 5. The search results are displayed on the page. |
| **Exceptions:** | | No items match with the keyword. |
| **Includes:** | | Matching Algorithm; Sorting Algorithm |
| **Special Requirements:** | | 1. The searching record will be archived as user preference. 2. The searching record is not accessible by administrators or other users. |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | In further development, keyword suggestion could be implemented. |

*Table 7: Exchange User-Case: Search Items*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 05 | |
| **Use Case Name:** | Reserve Items | |
| **Actors:** | | Users as buyers |
| **Description:** | | This use case allows the user (buyer) to reserve an item he/she wants to buy. |
| **Frequency of Use:** | | Every time the user (buyer) finds an item he/she wants to buy. |
| **Preconditions:** | | The user (buyer) searches for an item and has a list of results, but does not have reservation |
| **Postconditions:** | | The user (buyer) has reservation on the item. |
| **Normal Flow:** | | 1. The user (buyer) clicks on the reserve button. 2. The reservation status of the item is checked. 3. If the item is available, the reservation is successful and the item is now listed in the reserved item list. 4. If the item becomes unavailable during the search time, the reservation is not successful and the status of the item is updated. |
| **Exceptions:** | | The item is deleted by the user (seller) by the time. |
| **Includes:** | | Item Status Verification |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | The user (buyer) should receive notifications from the system if the information or status of the item he/she reserved has changed. |

*Table 8: Exchange User-Case: Reserve Items*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 06 | |
| **Use Case Name:** | Watch Items | |
| **Actors:** | | Users as buyers |
| **Description:** | | This use case allows the user (buyer) to watch an item he/she has interest but not sure if he/she wants to buy. |
| **Frequency of Use:** | | Every time the user (buyer) finds an item he/she has interest or the item which was reserved by another user. |
| **Preconditions:** | | The user (buyer) searched for an item and has a list of results, but has not already specified watch on the item. |
| **Postconditions:** | | The item is added to the user’s watchlist. |
| **Normal Flow:** | | The user (buyer) clicks on the watch button. |
| **Exceptions:** | | The item is deleted by the user (seller) by the time. |
| **Includes:** | | Locating Item Information |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | The user (buyer) should receive notifications from the system if the information or status of the item he/she watched has changed. |

*Table 9: Exchange User-Case: Watch Items*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 07 | |
| **Use Case Name:** | Cancel Reservation | |
| **Actors:** | | Users as buyers |
| **Description:** | | This use case allows the user (buyer) to cancel a reservation he/she has made before. |
| **Frequency of Use:** | | Every time Buyer decides not to buy an item |
| **Preconditions:** | | The user (buyer) has a reservation on an item. |
| **Postconditions:** | | The user (buyer) does not have a reservation on the item. |
| **Normal Flow:** | | 1. Buyer visits the reserved item page. 2. Buyer clicks on the cancel reservation button for certain item. 3. A notification is sent to the seller about the cancellation of the reservation. |
| **Exceptions:** | | None |
| **Includes:** | | Status Update |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access and the user has not finished the deal with the seller. |
| **Notes and Issues:** | | None |

*Table 10: Exchange User-Case: Cancel Reservation*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 08 | |
| **Use Case Name:** | Cancel Watch | |
| **Actors:** | | Users as buyers |
| **Description:** | | This use case allows the user (buyer) to remove items from watch list. |
| **Frequency of Use:** | | Every time the user (buyer) is no longer interested in an item. |
| **Preconditions:** | | The item is in the user’s watchlist. |
| **Postconditions:** | | The item is not in the user’s watchlist. |
| **Normal Flow:** | | 1. The user (buyer) visits the watched item page. 2. The user (buyer) clicks on the cancel watch button. |
| **Exceptions:** | | None |
| **Includes:** | | Status Update |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | The seller will not receive any notifications from the system. |

*Table 11: Exchange User-Case: Cancel Watch*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 09 | | | |
| **Use Case Name:** | Post Items for Sale | | | |
| **Actors:** | | Users as sellers | | |
| **Description:** | | This use case allows the user (seller) to post an item he/she wants to sell. | | |
| **Frequency of Use:** | | Every time the user (seller) has an item to sell. | | |
| **Preconditions:** | | The user (seller) has not posted the item for sale | | |
| **Postconditions:** | | The user (seller) has posted the item for sale | | |
| **Normal Flow:** | | 1. The user (seller) clicks on the plus button in either the exchange posts page or the exchange section on homepage. 2. The user (seller) provides name, description, photo, price, and contact information. 3. The user (seller) clicks on post button. | | |
| **Exceptions:** | | The required information is empty. | | |
| **Includes:** | | Input Verification  Attachments Upload/FTP Transfer | | |
| **Special Requirements:** | | The attachments should have size limitations. | | |
| **Assumptions:** | | The user has internet access and the user (seller) is honest. | | |
| **Notes and Issues:** | | The user (seller) is assumed to be honest, so that no further verification of the item is included in the application. | | |

*Table 12: Exchange User-Case: Post Items for Sale*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 10 | | | |
| **Use Case Name:** | Reject Reservation | | | |
| **Actors:** | | Users as sellers | | |
| **Description:** | | This use case allows the user (seller) to reject a reservation from other user (buyer). | | |
| **Frequency of Use:** | | Every time the user (seller) decide not to sell the item to the user (buyer) who made the reservation. | | |
| **Preconditions:** | | The user (buyer) has a reservation on the item. | | |
| **Postconditions:** | | The user (buyer) no longer has a reservation on the item. | | |
| **Normal Flow:** | | 1. The user (seller) visits the posted item page. 2. The user (seller) clicks on reject reservation button. | | |
| **Exceptions:** | | The reservation is already cancelled by the user during the time. | | |
| **Includes:** | | Status Verification | | |
| **Special Requirements:** | | None | | |
| **Assumptions:** | | The user has internet access. | | |
| **Notes and Issues:** | | After rejection, the status of item will become available again. So the buyer who was rejected by the seller will have chance to reserve the item again. The seller can choose to reject the buyer’s reservation again. However, a better way is to add the buyer to the block list so that the buyer cannot reserve the item any more. Notice that block list feature is low priority and whether to implement this feature depends on the overall progress. | | |

*Table 13: Exchange User-Case: Reject Reservation*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | 11 | | | |
| **Use Case Name:** | Delete Posting | | | |
| **Actors:** | | Users as sellers | | |
| **Description:** | | This use case allows the user (seller) to delete an item he/she poste. | | |
| **Frequency of Use:** | | Every time the user (seller) decides not to sell the item. | | |
| **Preconditions:** | | The item was posted for sale. | | |
| **Postconditions:** | | The item is no longer posted for sale. | | |
| **Normal Flow:** | | 1. The user (seller) visits the posted item page. 2. The user (seller) clicks on delete posting button. 3. A notification about the change will be sent the use (buyer) who has already reserved item. | | |
| **Exceptions:** | | None | | |
| **Includes:** | | None | | |
| **Special Requirements:** | | None | | |
| **Assumptions:** | | The user has internet access. | | |
| **Notes and Issues:** | | None | | |

*Table 14: Exchange User-Case: Delete Posting*

## 4.3 Events Requirements

4.3.1 Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Code** | **Description** | **Priority** |
| FR-19 | The user (participant) should be able to view and search campus events that are yet to be held. | High |
| FR-20 | The user (participant) should be able to join an open event and add it to calendar with join button. | High |
| FR-21 | The user (participant) should be able to cancel his/her join of the event any time before the event starts. | High |
| FR-22 | The user (participant) should not see events that have already occurred on their main event page. | High |
| FR-23 | The user (participant) should be able to indicate interest in events. | High |
| FR-24 | The user (participant) should be able to view events he/she is interested in the a separate events page. | High |
| FR-25 | The user (event organizer) should be able to add, edit, delete event information (features including event name, organization posters, time, address, availability, hashtag, word descriptions, attachments). | High |
| FR-26 | The user (event organizer) should be able to see joined participants for each events they created. | Medium |
| FR-27 | The user (event organizer) should be able to give permission to joined participants and add them to participants list after permission is assigned. | Medium |
| FR-28 | The user (participant) should be able to view their permission status of joined events in joined events page. | Medium |
| FR-29 | The user (event organizer) should be able to scan user’s QR code in order to grant a user access at event time. | Medium |
| FR-30 | The user (participant) should be able to launch Google Maps from Student Portal with routed destination set to the event location. | Medium |
| FR-31 | The user (participant) should be able to launch Uber from Student Portal with destination set to the event location. | Low |

*Table 15: Event Functional Requirements*

4.3.2 Use-Case Reports

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 12 | |
| **Use Case Name:** | View (Open, Joined, Interested) Events | |
| **Actors:** | | All Users |
| **Description:** | | This allows users to view events with selected category. |
| **Frequency of Use:** | | Every time the user enters events page or clicked on selected category. |
| **Preconditions:** | | The user has logged in on events page. |
| **Postconditions:** | | Events of selected category are displayed on the page. |
| **Normal Flow:** | | 1. The user is directed to the open events page after clicked on events button. 2. The user clicks on category(joined, interested). 3. The user is directed to the event page including events of selected category(Joined, Interested). 4. The user clicks on an event. 5. The user is directed to the page displaying the details of the events. 6. The user clicks back button. 7. The user is directed to previous events page. |
| **Exceptions:** | | None |
| **Includes:** | | Sorting Algorithm |
| **Special Requirements:** | | 1. Events are sorted by user preference or campus trending(first-time user). 2. Events past start time or application deadlines will not be displayed on open events page. Users can view them if they added it use the interested events or joined events. |
| **Assumptions:** | | The user has internet access . |
| **Notes and Issues:** | | For further development, the location where the user views on the events page should be saved so that when users move back from event detail page, they should be directed to the previous position on the events page. |

*Table 16: Event User-Case: View Events*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 13 | |
| **Use Case Name:** | Search Events | |
| **Actors:** | | All Users |
| **Description:** | | This use case allows users to search for events with certain keywords or hashtags. |
| **Frequency of Use:** | | Every time when the user want to view selected events.. |
| **Preconditions:** | | The user has logged in. |
| **Postconditions:** | | Searched results are displayed |
| **Normal Flow:** | | 1. The user clicks on search bar on events page. 2. The user types in any interested keywords or hashtag. 3. Matched events are displayed on events page. |
| **Exceptions:** | | The keyword is too long, or the hashtag doesn’t exist. |
| **Includes:** | | Matching Algorithm; Sorting Algorithm |
| **Special Requirements:** | | The results should be displayed in sorted order(best match first). |
| **Assumptions:** | | The user has internet access.. |
| **Notes and Issues:** | | For further development, keyword suggestion should be implemented. |

*Table 17: Event User-Case: Search Events*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 14 | |
| **Use Case Name:** | Join Events | |
| **Actors:** | | Users as participants; Users as event organizers |
| **Description:** | | This use case allows the user (participant) to confirm their participation of certain events. |
| **Frequency of Use:** | | Every time the user (participant) wants to notify the event organizer that he/she is going to an event. |
| **Preconditions:** | | The user is logged in and the user is on Events page. |
| **Postconditions:** | | 1. The event will be listed in user’s joined events and added to the calendar. 2. Group manager will be notified about the user’s participation and be able to see the user in joined users list. |
| **Normal Flow:** | | 1. The user clicks on join button beside the event information. 2. The event is archived in user’s joined events and added to the calendar. 3. The event is now in the user’s joined events list with name, time, permission, location, direction information listed. 4. A notification about user’s participation sent to the event organizer. 5. The user (participant) is in the name list of the joined users of this event. 6. The available spots of the event minus one. |
| **Exceptions:** | | Events reach capacity limitations. |
| **Includes:** | | Capacity Status Verification |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access and the user (participant) only joins an event if he/she is actually coming.. |
| **Notes and Issues:** | | None |

*Table 18: Event User-Case: Join Events*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 15 | |
| **Use Case Name:** | Cancel Join | |
| **Actors:** | | Users as participants; Users as event organizers, |
| **Description:** | | This use case allows the user (participant) to cancel a join of certain events. |
| **Frequency of Use:** | | Every time the user (participant) wants to notify the event organizer that he/she not going to an event. |
| **Preconditions:** | | The user (participant) is logged in on Events page, and the user has joined in certain events. |
| **Postconditions:** | | 1. The event will be removed from user’s joined events and be removed from the calendar. 2. The user (event organizer) will be notified about the user’s cancellation and the user (participant) is removed from joined users list. |
| **Normal Flow:** | | 1. The user (participant) clicks on cancel join button beside the event information. 2. The event is removed from user’s joined events and removed the calendar. 3. The user (participant) is not in the joined user list. 4. A notification about the change is sent to the user (event organizer). 5. The available spots of the event plus one. |
| **Exceptions:** | | None |
| **Includes:** | | Status Update  Synchronization |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | None |

*Table 19: Event User-Case: Cancel Join*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 16 | |
| **Use Case Name:** | Interested/Add to Calendar | |
| **Actors:** | | All Users |
| **Description:** | | This use case allows users to bookmark an event or add an event to the calendar. |
| **Frequency of Use:** | | Every time the user wants to save an event. |
| **Preconditions:** | | The user is logged in on Events page. |
| **Postconditions:** | | The event will be listed in interested events and/or added to the calendar. |
| **Normal Flow:** | | 1. The user clicks on interested to button beside the event information. 2. The event is archived in user’s interested events. 3. In interested events, the user can click on add to calendar button to archive the event to the calendar. |
| **Exceptions:** | | 1. The user does not provide permission to access google calendar. 2. The event information is deleted during the time. |
| **Includes:** | | Status Verification |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | None |

*Table 20: Event User-Case: Interested/Add to Calendar*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 17 | |
| **Use Case Name:** | Add/Edit/Cancel Event | |
| **Actors:** | | Users as event organizers |
| **Description:** | | This use case allows the user (event organizer) to post/change/delete event information. |
| **Frequency of Use:** | | Every time the user (event organizer) wants to post event information. |
| **Preconditions:** | | The user (event organizer) has logged and is on the Events page. |
| **Postconditions:** | | The event information is posted/changed/deleted. |
| **Normal Flow:** | | 1. The user (event organizer) posts(2.1), edits(2.2) or deletes(2.3) an event information.    1. The user (event organizer) wants to create an event.       1. The user (event organizer) clicks on create event button and fills an event information:          1. Required: time, location, event name, price for tickets, available spots          2. Optional: attachments(jpg, url), text description,ticket sales information, hashtag, organization name       2. The user (event organizer) submits event information.       3. Events are displayed and can be viewed by other users (participants).    2. The user(event organizer) wants to change event information.       1. The user (event organizer) on homepage or Events page click on manage button.       2. Created events will be displayed.       3. The user (event organizer) clicks on specific events to change.       4. Details of the events are displayed with editable information.       5. The user (event organizer) changes the event information and submits.       6. Events with updated information is displayed and ready to be viewed.       7. The user (participant) who joined this event will receive notification of change (email).    3. The user (event organizer) wants to cancer an event.       1. The user (event organizer) on homepage or event page click on manage button.       2. Created events will be displayed.       3. The user (event organizer) clicks on cancel button and verifies the cancellation in message box.       4. The event information will no longer be displayed in Events page.       5. The event will be shown as cancelled in created events page. |
| **Exceptions:** | | Required information is empty.  Number of joined users exceeds the updated capacity. |
| **Includes:** | | Input Verification |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access and the event information posted is real. |
| **Notes and Issues:** | | For future development, the event information should be verified(inappropriate information filtered) by the system before viewed by other users. |

*Table 21: Event User-Case: Add/Edit/Cancel Event*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 18 | |
| **Use Case Name:** | Permission Management | |
| **Actors:** | | Users as event organizers; Users as participants. |
| **Description:** | | This use case allows the user (event organizer) to manage the permission for event participants. |
| **Frequency of Use:** | | Every time the user (event organizer) wants to update permission list for an event he/she created. |
| **Preconditions:** | | The user (event organizer) is logged in. |
| **Postconditions:** | | The permission status for certain users for certain event is changed. |
| **Normal Flow:** | | 1. The user (event organizer) clicks on manage button and is directed to all the events created 2. The user (event organizer) clicks on a specific event. 3. The user (event organizer) wants to manage the permission of a certain user.    1. Add the user (participant) as permitted participants.       1. If the user (participant) is in the joined list, the user (event organizer) selects user in the joined list and clicks the give permission button. The user (participant) is now added to the permitted lists.       2. If the user (participant) is not in the joined list, the user (event organizer) clicks on add user button in the permitted list, enters the username/campus Id. The user (participant)is now in the permitted list.    2. Cancel the permission for a user (participant).       1. If the user (participant) is in the joined list, the user (event organizer) selects user in the joined list and clicks on cancel permission button. The user (participant) is removed from permitted list.       2. If the user (participant) is only in the permitted list, the user (event organizer) clicks on remove button and the user (participant) is removed from the permitted list. 4. The permitted list is updated. 5. The user (participant) receives a notification about their permission status change. 6. The user (participant) will receive a notification if he/she is not permitted to joined an event at least one day before the event starts. |
| **Exceptions:** | | The user (participant) is not found, if the username or the campus ID the organizer entered is incorrect. |
| **Includes:** | | Account Verification |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access and the user (event organizer) has created events. |
| **Notes and Issues:** | | None |

*Table 22: Event User-Case: Permission Management*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 19 | |
| **Use Case Name:** | Check-in | |
| **Actors:** | | Users as participants; User as organizers |
| **Description:** | | This use case allows fast event check-in with the app. |
| **Frequency of Use:** | | Every time an event needs check-in. |
| **Preconditions:** | | The user is logged in. |
| **Postconditions:** | | The user (participant) is checked in and the user (organizer) can scan the next participant. |
| **Normal Flow:** | | 1. The user (participant) clicks on the QR code in their profile and QR code is displayed. 2. The user (event organizer) clicks on manage events and clicks on QR scan icon beside certain event in his/her created events list. QR scan UI is displayed. 3. The user (event organizer) scans the user QR code.    1. If the user (participant) is in the permitted list, a √ will be shown on the user (event organizer) UI.    2. If the user (participant) is not in the permitted list, a × will be shown on the user (event organizer) UI. 4. The user (event organizer) clicks on back to scan button and returns to QR scan interface. 5. The user (event organizer) is now ready to scan next user. |
| **Exceptions:** | | QR code not recognized. |
| **Includes:** | | QR Code Scanner |
| **Special Requirements:** | | None |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | None |

*Table 23: Event User-Case: Check-in*

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | 20 | |
| **Use Case Name:** | Direction | |
| **Actors:** | | Users as participants |
| **Description:** | | This use case allows the user (participant) to be directed to an interest events or a joined events. |
| **Frequency of Use:** | | Every time the user (participant) wants to go to an event. |
| **Preconditions:** | | The user is logged in on Events page. |
| **Postconditions:** | | The user is directed by Google Maps or the user can order rides in Uber. |
| **Normal Flow:** | | 1. The user (participant) clicks on joined event button.. 2. The user (participant) selects the joined event he wants to be directed to.    1. The user (participant) clicks on the location button, and Google Maps will be opened, showing the location of the event.    2. The user (participant) clicks on the Uber button, and Uber will be opened. The destination address will be filled in by the system. |
| **Exceptions:** | | Location not found or not provided.  Google Maps or Uber is not installed. |
| **Includes:** | | None |
| **Special Requirements:** | | Uber; Google Maps |
| **Assumptions:** | | The user has internet access. |
| **Notes and Issues:** | | None |

*Table 24: Event User-Case: Direction*

# 5. **Other Nonfunctional Requirements**

## 5.1 **Performance Requirements**

PR-1: The system should accommodate 200 users during the peak usage time window of 8:00 am to 8:00 pm local time, with an estimated average session duration of 10 minutes.

PR-2: The system should display confirmation messages to users within 5 seconds after the user submits information to the system.

PR-3: Search results should be displayed within 5 seconds.

## 5.2 **Security Requirements**

SR-1: The system must be secured to ensure that unauthorized users cannot login and access the system.

SR-2: The system will use OAuth2 for accessing Google APIs, and administrators of the system will not have access to user login credentials.

SR-3: The system will not reveal the private information of other users.

**5.3 Software Quality Attributes**

SQ-1: The Student Portal system should be available to users on the corporate internet 24 \* 7.

SQ-2: The system should generate a backup copy of the application data and log files.

SQ-3: If the connection between the user and the system is broken prior to an post being either posted or deleted, the Student Portal system should enable the user to recover an incomplete post.

**Appendix A: Glossary**

**Amazon EC2:** Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. [1]

**Android:** Android is a mobile operating system developed by Google, which is based on the Linux kernel. [2]

**API:** The application programming interface (API) is a set of commands, functions, protocols, and objects that programmers can use to create software or interact with an external system. [3]

**Cluster analysis**: Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). [4]

**Collaborative Filtering**: Collaborative filtering methods are based on collecting and analyzing a large amount of information on users’ behaviors, activities or preferences and predicting what users will like based on their similarity to other users. [5]

**FTP:** The File Transfer Protocol (FTP) is a protocol designed for transferring files over the Internet. [6]

**Hashtag:** The hashtag is a type of label or metadata tag used on social network and microblogging services which makes it easier for users to find messages with a specific theme or content. [7]

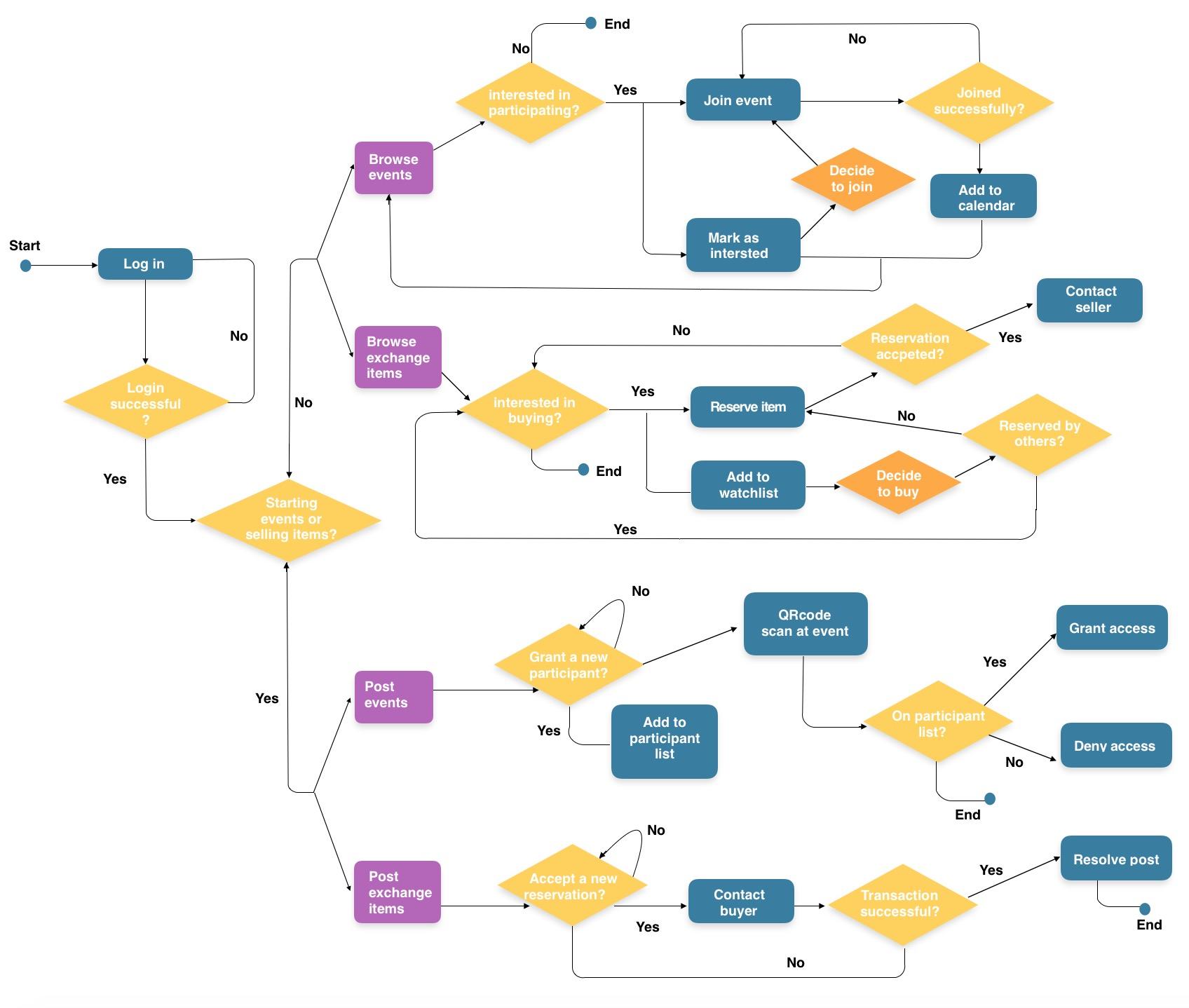
**HTTP:** The Hypertext Transfer Protocol (HTTP) is the protocol used to transfer data over the web. [8]

**OAuth:** OAuth (Open Authorization) is an open standard for token-based authentication and authorization on the Internet. [9]

**QR code:** The QR code (quick response code) is a type of barcode that contains a matrix of dots. It can be scanned using a QR scanner or a smartphone with built-in camera. [10]

**XML:** The Extensible Markup Language (XML) is used to define documents with a standard format that can be read by any XML-compatible application. [11]

**Appendix B: Analysis Models**



*Figure 4: Student Portal basic function activity diagram*

**Appendix C: Figure and Table Lists**

|  |  |
| --- | --- |
| **Section Number** | **Figure Name** |
| 2.2 Product Functions | *Figure 1: External Entity of the Student Portal* |
| 3.1 User Interfaces | *Figure 2: Prototype sample view of the homepage & Sliding Menu* |
| 3.1 User Interfaces | *Figure 3: Prototype sample view of the Manage Page & Event Page* |
| Appendix B: Analysis Models | *Figure 4: Student Portal basic function Activity Diagram* |

|  |  |
| --- | --- |
| **Section Number** | **Table Name** |
| 4.1 Homepage Requirements  4.1.1 Functional Requirements | *Table 1: User Classes and Characteristics* |
| 4.1.2 Use-Case Reports | *Table 2: Homepage Functional Requirements* |
| 4.1.2 Use-Case Reports | *Table 3: Homepage User-Case: Register* |
| 4.1.2 Use-Case Reports | *Table 4: Homepage User-Case: Sign In* |
| 4.1.2 Use-Case Reports | *Table 5: Homepage User-Case: Log out* |
| 4.2 Exchange Requirements  4.2.1 Functional Requirements | *Table 6: Exchange Functional Requirements* |
| 4.2.2 Use-Case Reports | *Table 7: Exchange User-Case: Search Items* |
| 4.2.2 Use-Case Reports | *Table 8: Exchange User-Case: Reserve Items* |
| 4.2.2 Use-Case Reports | *Table 9: Exchange User-Case: Watch Items* |
| 4.2.2 Use-Case Reports | *Table 10: Exchange User-Case: Cancel Reservation* |
| 4.2.2 Use-Case Reports | *Table 11: Exchange User-Case: Cancel Watch* |
| 4.2.2 Use-Case Reports | *Table 12: Exchange User-Case: Post Items for Sale* |
| 4.2.2 Use-Case Reports | *Table 13: Exchange User-Case: Reject Reservation* |
| 4.2.2 Use-Case Reports | *Table 14: Exchange User-Case: Delete Posting* |
| 4.3 Events Requirements  4.3.1 Functional Requirements | *Table 15: Event Functional Requirements* |
| 4.3.2 Use-Case Reports | *Table 16: Event User-Case: View Events* |
| 4.3.2 Use-Case Reports | *Table 17: Event User-Case: Search Events* |
| 4.3.2 Use-Case Reports | *Table 18: Event User-Case: Join Events* |
| 4.3.2 Use-Case Reports | *Table 19: Event User-Case: Cancel Join* |
| 4.3.2 Use-Case Reports | *Table 20: Event User-Case: Interested/Add to Calendar* |
| 4.3.2 Use-Case Reports | *Table 21: Event User-Case: Add/Edit/Cancel Event* |
| 4.3.2 Use-Case Reports | *Table 22: Event User-Case: Permission Management* |
| 4.3.2 Use-Case Reports | *Table 23: Event User-Case: Check-in* |
| 4.3.2 Use-Case Reports | *Table 24: Event User-Case: Direction* |