Software Design Document

for

Student Portal

Proposed by Yang Ding, Peiyan Duan, Xiaoying Ji,

Duan Li, Tianlin Lu, Chris Tsuei

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
SDD for Student Portal	10-14-16	Initial Release	v1.0

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to describe the aspects of the Student Portal system, whose features include trading secondhand items under Exchange section, and providing all means from registering towards arriving at on-campus events with Events functionalities. This document will describe the components of our application, specifically the Home, Exchange, Events and Manage activities as well as how each component responds to user interaction and interact with the others components of our system.

1.2 Scope

The Student Portal system is designed to be an Android application that provides a campus and community information platform for college or university students. Having a unified platform that includes both a community exchange as well as a campus events system will allow students to further interact with their peers as well as keep informed about events that campus and local organizations may be hosting.

1.3 Overview

This document is intended to describe how each component of the Student Portal will interact with the user as well as with other components of the system. Section 2 (System Overview) provides a brief description of each component of the Student Portal. Section 3 (System Architecture) will provide a detailed description module structure. Section 4 (Data Design) will describe how the information used by the system and entered by the user is interpreted and converted into an interaction with the system as well as other users. Section 5 (Component Design) describes component functions in details. Section 6 (Human Interface Design) describes the various user interactions with the application and how the app responds to user interaction. Finally, Section 7 (Requirements Matrix) shows how the components will satisfy the requirements of the System Requirements Document.

1.4 Reference Material

[1] *Student Portal System Requirements Document,* 2016
[2] "Android (operating system)", *Wikipedia*, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Android_(operating_system). [Accessed: 09- Oct- 2016].
[3] "Fragments | Android Developers", *Developer.android.com*, 2016. [Online]. Available: https://developer.android.com/guide/components/fragments.html. [Accessed: 09- Oct- 2016].
[4] "Activity | Android Developers", *Developer.android.com*, 2016. [Online]. Available: https://developer.android.com/reference/android/app/Activity.html. [Accessed: 09- Oct- 2016].
[5] C. Drawer, "Creating a Navigation Drawer | Android Developers", *Developer.android.com*, 2016. [Online]. Available:https://developer.android.com/training/implementing-navigation/nav-rawer.html. [Accessed: 09-Oct-2016].

[6] "Model-View-Controller", *Wikipedia*, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Model–view–controller. [Accessed: 08-Oct-2016].
[7] N. Bhatt, "MVC vs. MVP vs. MVVM", *Wordpress.com*, 2009. [Online]. Available: https://nirajrules.wordpress.com/2009/07/18/mvc-vs-mvp-vs-mvvm/.[Accessed: 08-Oct-2016].

1.5 Definitions and Acronyms

Android - operating system based off of the Linux kernel designed primarily for touchscreen devices
Activity - single thing a user can do
Coupling - coupling is the degree of interdependence between software modules; a measure of how closely connected two routines or modules are
Fragments - behavior or a portion of the user interface in an Activity
Navigation Drawer - panel that displays the application's main options on the left edge of the screen. This is hidden most of the time unless the user swipes from the left edge of the screen or touches the app icon.
MVC - Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.
MVP - Model–view–presenter (MVP) is a user interface architectural pattern engineered to facilitate automated unit testing and improve the separation of concerns in presentation logic

2. SYSTEM OVERVIEW

Our product is an Android application created using Android Studio with the target audience being college or university students. After the user logs into the system with a valid Case ID and associated password using the Oauth verification system, they will be able to access the main components of our application.

On the title bar of the home screen, there will be a sliding menu in the upper left corner and a search bar. The sliding menu can be opened when the user taps the upper left corner of the application in any window of our application. It will allow the user to change the screen to access their profile page, the manage page, which deals with user created event and exchange listings, the exchange page, the campus events page, as well as logout of the app if the user desires to. The search bar will allow the user to quickly search for a campus event or exchange listing after they type what they wish to search.

The rest of the space will display a preview of the exchange page as well as the events page. The user will be able to browse events and exchange items by touching the arrows on the application without needing to open those pages. Also, users will be able to create events or exchange listings by tapping the plus icon in either section of the exchange or events previews.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

The Student Portal System has two subsystems: the Event Subsystem and the Exchange Subsystem. The two subsystems share the same account so the user only need to login to the account once to access both subsystems.

3.1.1 Event Subsystem

Event Subsystem enables users to post or manage events as organizers and search for or join events as participants simultaneously. 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 and used when displaying popular events on the homepage.

3.1.2 Exchange Subsystem

Exchange Subsystem allows users to sell second-hand items as sellers and search for or reserve secondhand items as buyers at the same time. 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 during the selling process and a rating system after an item is sold.

3.1.3 Architectural Pattern

The Model View Presenter (MVP) interface architectural pattern is applied to the Student Portal mobile application. The reason we choose to use MVP rather than other architectures will be explained in details in Section 3.2 Design Rationale.

The aim of MVP is to separate the components of user interface (View); core functionality and data (Model) and the user gesture handler (Presenter). The Student Portal System consists of: UI components (updating a list of featured items or events and displaying these to the user), service components (fetching data from other services), data components (storing and retrieving user information), and widget components (handling user clicks and inputs). As shown in Figure 1, MVP is applied to Student Portal mobile application.

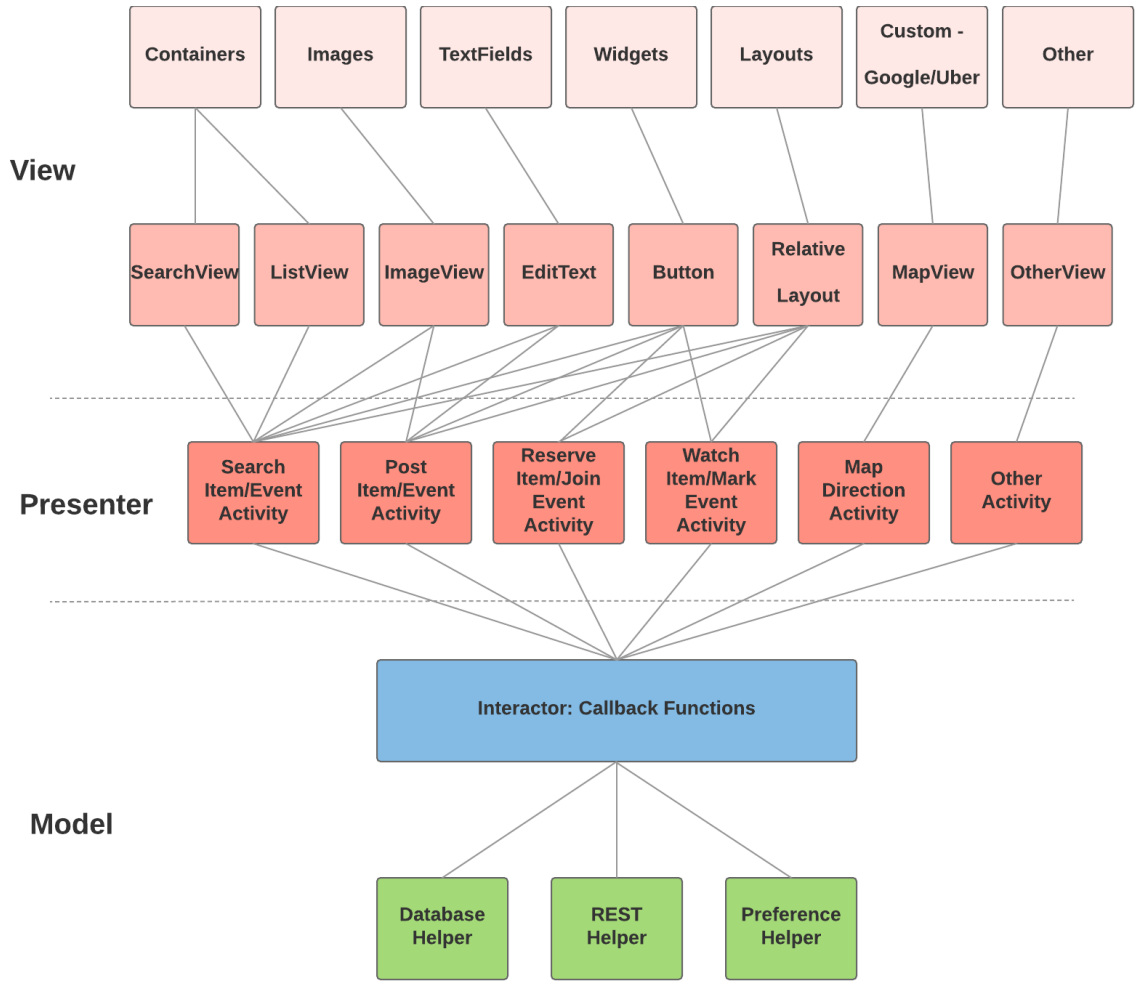

Figure 1: Student Portal MVP Architecture

3.1.4 Sequence Diagrams

Buyer: Login, Search item, Reserve item/Watch item
Participant: Login, Search event, Join event/Mark event

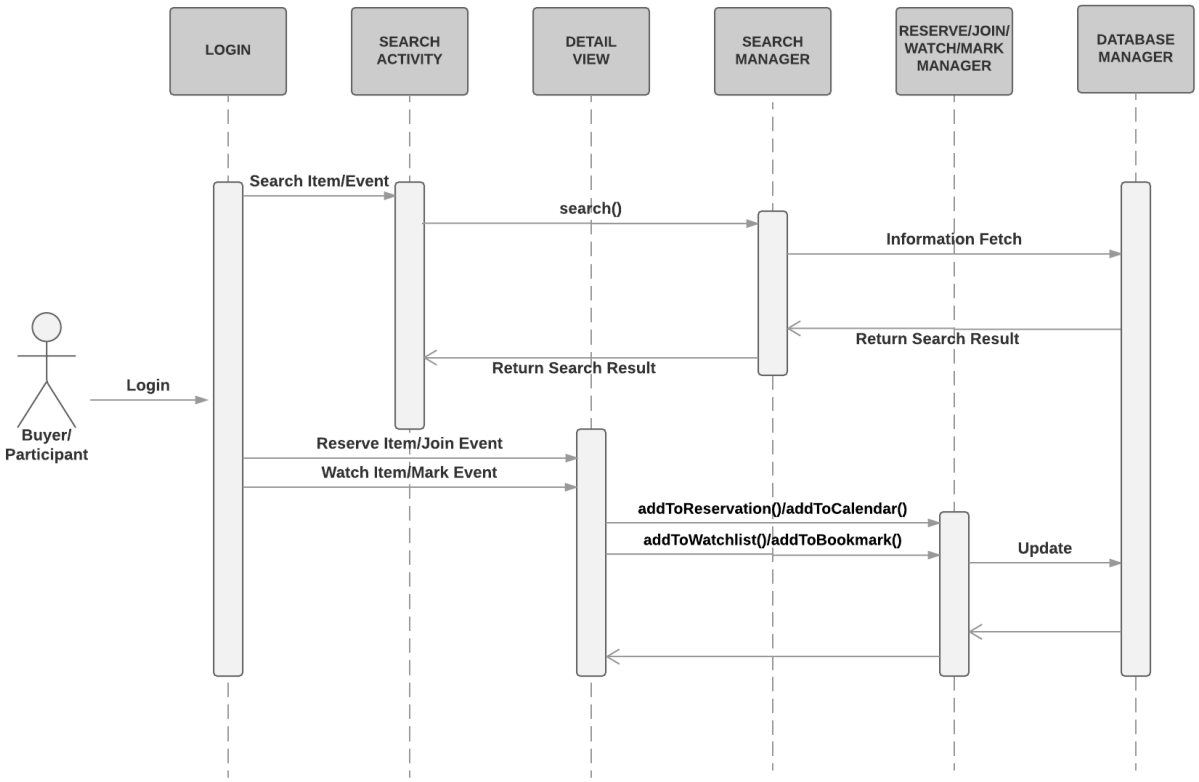

Figure 2: Buyer/Participant Major Actions Sequence Diagram

Seller: Login, Create New Item, Resolve Item after transaction
Organizer: Login, Create New Event, Grant participant access

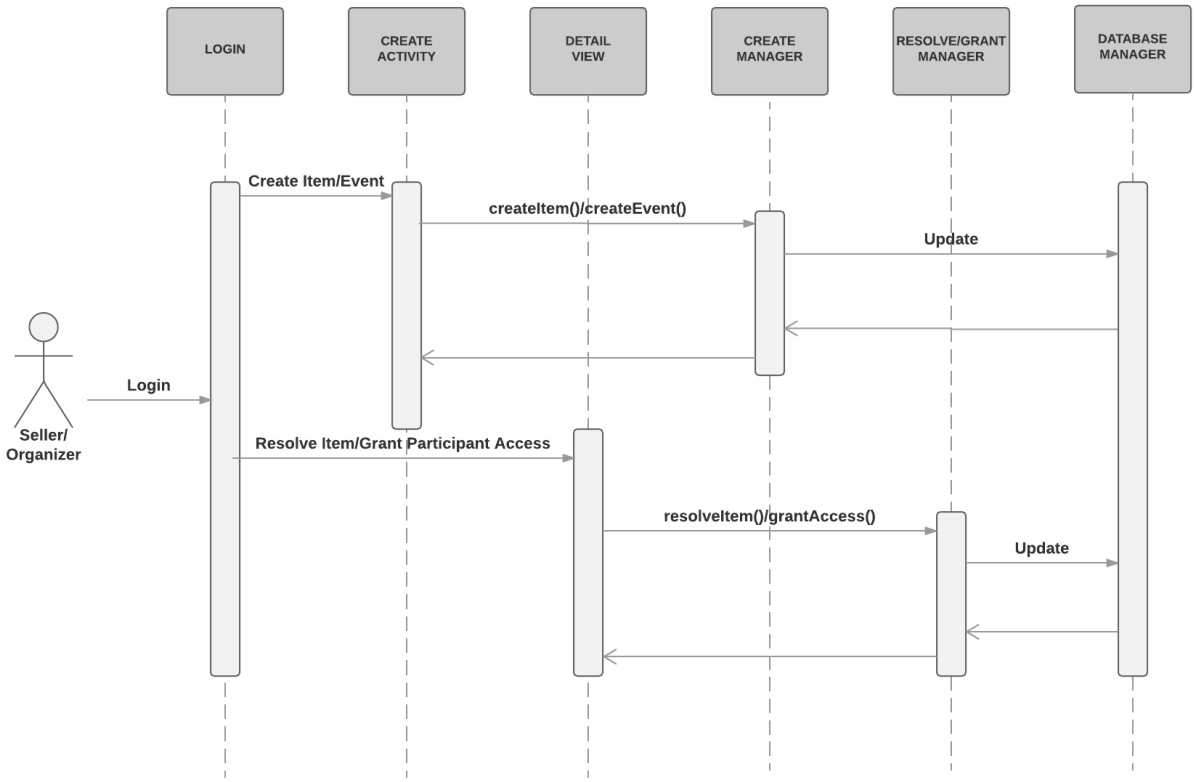

Figure 3: Seller/Organizer Major Actions Sequence Diagram

3.2 Design Rationale

3.2.1 Model-View-Controller Pattern

Model-View-Controller (MVC) is a widely used software architecture pattern. MVC divides a given software application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to or accepted from the user [6]. The three components are View, Model, and Controller. View contains the UI component. Model contains the entities and/or the data that the view is displaying to the user. Controller involves the actions that modify the model based on the user interaction.

The main issue with MVC is that the View layer has too many responsibilities. This will make

maintain Activities and Fragments difficult. The code will be ugly and difficult to understand due to too many nested callbacks and a high degree of coupling, which makes changing or adding new features very painful. More than that, heavy logic living within the Activities or Fragments makes unit test challenging and arduous.

3.2.2 Model-View-Presenter Pattern

Model-View-Presenter (MVP) is a derivation of the MVC architectural pattern. As mentioned in Section 3.1.3 Architecture Pattern, MVP has three components: View, Model, and Presenter. But dependencies change by replacing Controller with Presenter. The main difference between both is that Presenter refers back to the view while Controller doesn't [7]. MVP could bring very valuable improvements to our existing approach. Because our current architecture was divided in two layers (view and data), adding MVP felt natural. We simply had to add a new layer of presenters and move part of the code from the view to presenters.

3.2.3 Why MVP?

MVP perfectly solves the main issues of MVC. First, Activities and Fragments become very lightweight. Their only responsibilities are to set up or update the UI and handle user events. Separating interface from logic in Android is not easy, but MVP prevent our activities end up degrading into very coupled classes consisting of too many lines of code. Therefore, they become easier to maintain.

Second, MVP takes most of logic out from the activities so that we can test it without using instrumentation tests. We can now easily write unit tests for the presenters by making the view layer. The whole architecture becomes very test-friendly. If the data manager becomes bloated, we can mitigate this problem by moving some code to the presenter. Overall, MVP makes views independent from our data source.

4. DATA DESIGN

Our database model is based on SQLite, which is widely used for Android Application. SQLite has built-in connection between SQL queries with JAVA language, and therefore would be reliable and easy to implement for the student portal system. The input data comes from front-end user input and the output data is the query result of the user request.

The database has three major entities: user, item and event. User actions are specified by the relationships between three entities, including reserve, sell, block, watch, organize, join, bookmark and permit. As shown in Figure 4, the relation schemes of the database and the details of attributes in entities and relationships are displayed.

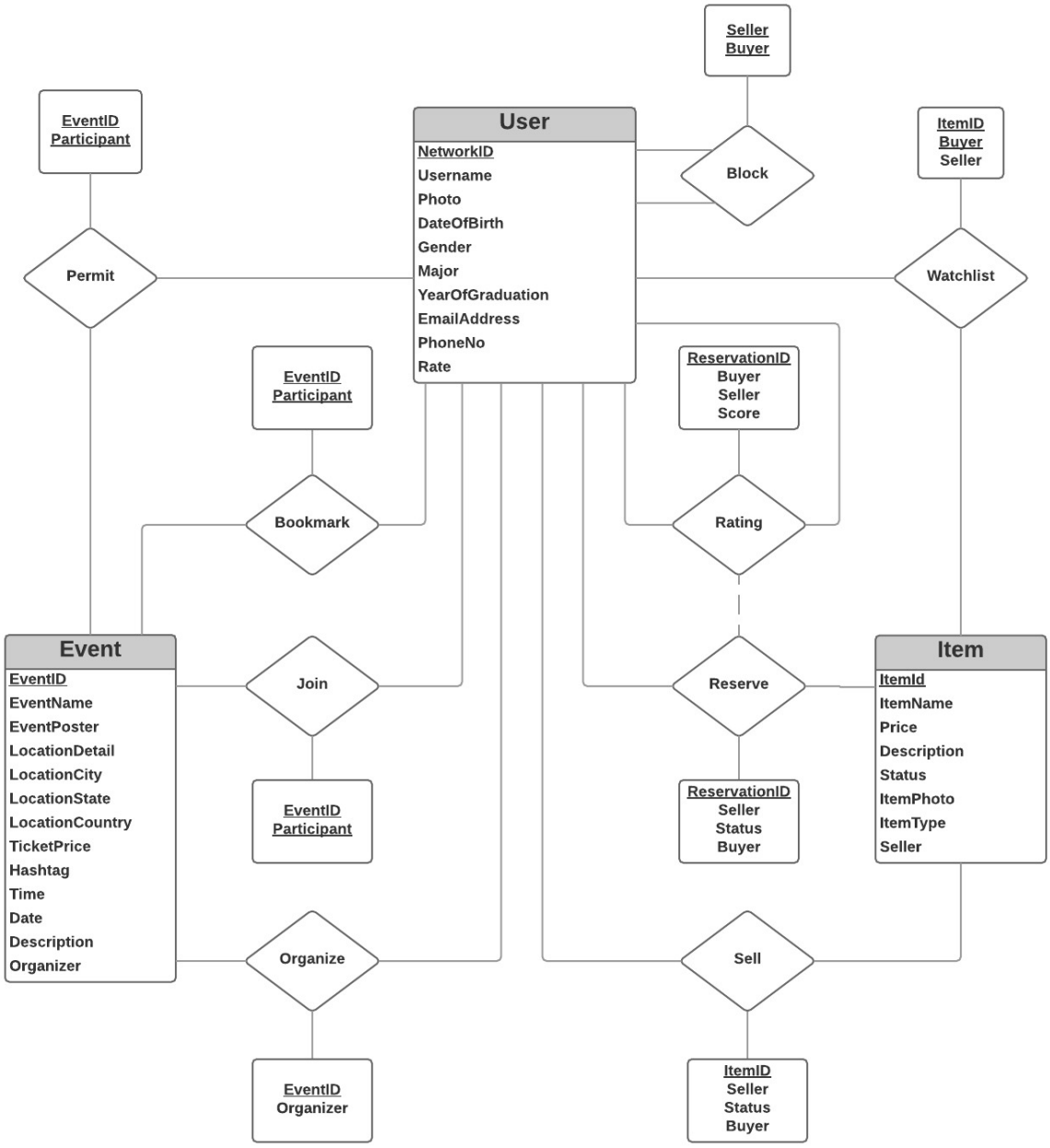

Figure 4 Student Portal ER Diagram

Entities:

1 User

- NetworkID:
  - A non-zero 4 to 6 digit **Integer** as primary key.
  - The user is not able to change the network ID once the account is created.
- Username:
  - A **Varchar** name less than 20 characters.
  - The default is the user's network ID.

    - The user can edit this attribute.
- Photo:
    - An accessible **Varchar** picture url.
    - The user can edit this attribute.
- DateOfBirth
    - A **Date** variable.
    - This attribute can be used for collaborative filter system.
- Gender
    - A 1 digit **Integer** indicating the gender of the user for collaborative filter system.
    - 0: Not provided. 1: Male. 2: Female. 3: Other.
    - The attribute can be used for collaborative filter system.
    - The user can edit this attribute.
- Major
    - A **Varchar** predefined value that can be selected by the user.
    - The user can edit this attribute.
    - The attribute can be used for collaborative filter system.
- YearOfGraduation
    - A 4 digit **Integer** that can be selected by user.
    - The attribute can be used for collaborative filter system.
- EmailAddress
    - A **Varchar** email address that is not modifiable once the account is created.
- PhoneNo
    - A 9 digit **Integer**, assuming the user provide real information.
    - The user can edit this attribute.
    - The attribute can be used for exchange system.
- Rate
    - A **Double** from 0 to 5 and -1 indicating the rate of the user as a seller
    - The default is -1 indicating the user has not finish any exchange transaction as a seller.
    - The user cannot edit this attribute.
    - The user can view this attribute in his/her profile.

2 Item

- ItemID
    - A **Char** variable of length 10 as primary key.
    - The user cannot edit this attribute once created.
- ItemName
    - A **String** variable less than 30 characters.
    - The user can edit this attribute.
- Price
    - An **Integer** less than 10 digits.
    - The user can edit this attribute.
- Description
    - A **String** variable less than 600 characters.
    - The user can edit this attribute.

- Status
    - A 1 digit **Byte** indicating the status of the item.
    - The user cannot access this attribute.
    - This attribute is determined by user action.
    - 0: The item is unreserved.
    - 1: The item is reserved.
- ItemPhoto
    - An accessible **Varchar** picture url.
    - The user can edit this attribute.
- ItemType
    - A **Varchar** predefined value that can be selected by the user.
    - The attribute can be used for collaborative filter system.
- Seller
    - A foreign key from userID in user entity.

3 Event

- EventID
    - A **Char** variable of length 10 as primary key.
    - The user cannot edit this attribute once created.
- EventName
    - A **String** variable less than 30 characters.
    - The user can edit this attribute.
- EventPoster
    - An accessible **Varchar** picture url.
    - The user can edit this attribute.
- LocationDetail
    - A **String** variable less than characters.
    - The user can edit this attribute.
    - For further development, the user can select location based on Google map recommendation with the help of Google map api.
- LocationCity
    - A **Varchar** predefined value that can be selected by the user.
- LocationState
    - A **Varchar** predefined value that can be selected by the user.
- LocationCountry
    - A **Varchar** predefined value that can be selected by the user.
- TicketPrice
    - An **Integer** less than 10 digits.
    - The user can edit this attribute.
- Capacity
    - An **Integer** less than 10 digits.
    - The user can edit this attribute.
- Hashtag
    - A **String array** indicating the hashtag of the event.
    - The user can edit this attribute.

- Time
  - A **Time** variable indicating the time the check-in starts.
  - The user can edit this attribute.
- Date
  - A **Date** variable indicating the date the event happens.
  - The user can edit this attribute.
- Description
  - A **String** variable less than 600 characters.
  - The user can edit this attribute.
- Organizer
  - A foreign key from userID in user entity.

Relationships:

1 Sell
- ItemID from item entity as primary key
- Seller: Network ID from user entity
- Status from item entity
- Buyer: Network ID from user entity
  - The buyer should be the one reserve the item when the seller indicate the transaction is finished.
  - The default value is -1 indicating unfinished transaction.

2 Reserve
- ReservationID
  - A **Char** variable of length 10 as primary key.
  - The user cannot edit this attribute once created.
- ItemID from item entity as primary key
- Buyer: Network ID from user entity
- Seller: Network ID from user entity
- Status: A 1-digit **Integer** indicating the status of the reservation
  - 0: indicating the reservation is wait to be viewed
  - 1: indicating the reservation is rejected
  - 2: indicating the reservation is confirmed

3 Watchlist
- ItemID from item entity as primary key
- Buyer: Network ID from user entity as primary key
- Seller: Network ID from user entity

4 Block
- Seller: Network ID from user entity as primary key, indicating the owner of the blocklist
- Buyer: Network ID from user entity as primary key, indicating people being blocked

5 Rating
- ReservationID from reserve relationship as primary key
  - Only reservation with status=2 can be rated.
- Buyer: Network ID from user entity
- Seller: Network ID from user entity

- Score:
    - A **Double** from 0 to 5 indicating the rate of buyer to seller in this transaction
    - The user cannot edit this attribute.
    - The user can view this attribute in his/her past transaction.

6 Organize
- EventID: EventID from event entity as primary key
- Organizer: Network ID from user entity

7 Join
- EventID: EventID from event entity as primary key
- Participant: Network ID from user entity as primary key

8 Bookmark
- EventID: EventID from event entity as primary key
- Participant: Network ID from user entity as primary key

9 Permit
- EventID: EventID from event entity as primary key
- Participant: Network ID from user entity as primary key

5. COMPONENT DESIGN

5.1 Component Function Description

Activity	Function	No.	Description	Data Flow
General	Error messages	CF01	Specifies all possible error messages that will occur during the user interaction. Stores and implements specific handles for each case.	No interaction with the system database.
	Image Write	CF02	Uploads pictures to the server for our application and returns the URL for the image uploaded.	Connects to the server, sends data snapshots.
	Image Read	CF03	Retrieve the image specified by the URL through our server.	Connects to the server, retrieves data.
	Information Write	CF04	With built in SQL functions in Android studio, the function stores information in Java or XML into database.	Stores data into the SQL database.

	Information Read	CF05	With built in SQL functions in Android studio, the function retrieves information in Java or XML into database.	Retrieves data from SQL database.
	Verify	CF06	Verifies the input data meets the constraints of database entries and entities.	Matching input information with data.
	Populate	CF07	Displays the initial list and information on creating the activity.	Reads information and picture from database.
Login Activity	Register	CF08	With the help of OAuth2 service, the function creates user tokens for our system based on the Google account the user provided. If the user grants permission, the Google Authorization Server will send our application an access token, or an authorization code that our application can use to obtain an access token.	No interaction with the system database.
	Login	CF09	The function matches the username and password with the information in the database. If corresponding information is found, a copy of user information is stored locally, otherwise, the user interface will indicate the mismatch.	A query with username and password, asking for more user information of certain tuple, is sent to the database.
Profile Activity	Edit Profile	CF10	All user information is displayed as either editable information or permanent information. Edited information will be verified and written to the database system.	Verifies that the input data conforms to the data constraints and writes to database.
Menu Activity	Logout	CF11	User can logout through clicking on the logout button at the end of	No interaction with the database.

			the sliding menu.	
MainPage Activity	Sliding	CF12	The main page is divided into two section including Event and Exchange. User can browse through different events and items by right/left swiping the sliding menu.	Reads event and exchange information from the database
ListView Activity	Update	CF13	ListView is a viewgroup that displays a list of scrollable items. The list items are automatically inserted to the list using an adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.	Browses a list of scrollable event and exchange information from the database
NewEvent Activity	NewEvent	CF14	Creates new tuple in the event table with event ID, event name, time, location, capacity, description, and item status.	Writes new data into the database
NewItem Activity	NewItem	CF15	Creates new tuple in the item table with item ID, item name, price, description, seller contact, and item status.	Write new data into the database
Event Detail Activity	Join	CF16	With the help of Google API, creates new event on the user's GoogleC calendar.	No interaction with the system data.
	Bookmark	CF17	Creates new tuple in the bookmark relationship with user ID and event ID.	Writes new relationship into the database.
	Cancel Join	CF18	Delete the event from user's Google Calendar	No interaction with the system data
	Cancel Bookmark	CF19	Deletes the tuple in the bookmark relationship with user ID and event ID	Deletes existing relationship in the database
Exchange	Watchlist	CF20	Creates new tuple in the watchlist	Writes new

Detail Activity			relationship with buyer ID, item ID.	relationship into the database.
	Reserve	CF21	Creates new tuple in the reservation relation scheme with buyer ID, item ID and seller ID.	Writes new relationship into the database.
	Cancel Watchlist	CF22	Deletes the tuple in the watchlist relationship with buyer ID, item ID	Deletes existing relationship in the database
	Cancel Reservation	CF23	Cancel the reservation created by buyer with buyer ID, item ID	Delete existing relationship in the database
Manage Activity	Manage	CF24	For events joined/ items bought, retrieves all tuples in the database whose participant/buyer attribute equals the network ID of the current user.	Reads data from the database.
Permission Activity	Modify Permission	CF25	After the CF25permission status is changed, a new tuple with updated participant information should be inserted to the database.	Writes new relationship into the database.
	Add User	CF26	Creates new tuple in the GuestList relation with permission status attribute as granted.	Writes new relationship into the database.
Search Activity	Search	CF27	The algorithm analyzes the user input, separating the words with pre-loaded stop words or spaces. Then the values in the analyzed string array will be compared to the attributes: event name, item name, keywords in the database. The found string will be returned as an integer array.	Reads data and compares the value in the database.
	Sort	CF28	Sorts the search result array or list array displayed in one activity by the value user chooses, or by the	Compares the attribute values in the database.

			maximum match for the search result. By value: Takes the value user selects and compares the corresponding attributes for all the items or events in the list and sort them in the order of the value of selected attributes. By maximum match: records number of keywords matched and stores in descending order in the array.	
	Filter	CF29	Filters values of certain attributes, either eliminate or select attribute with selected values.	Reads and filters data in the database.
Google Map Activity	Google Map	CF30	With Google map API, verifies user permission of location access, then collects user location and target location, finds the route and guides the user.	No interaction with the database system.
QR Activity	QR Display	CF31	Displays a QR code for the user by calling the Encode method	No interaction with the database
	Scan	CF32	With QR API, scans a QR code provided to the event organizer by the user attending the event and compares the decoded user id to the user ids in the attendee list.	Reads QR code from attendee, compares attendee information with attendance list from the database
	Encode	CF33	Takes the user ID and event information and encodes it, generates a QR code to be displayed (No API usage)	No database interaction
	Decode	CF34	Translates a QR code to plain text (No API usage)	No database interaction
Uber	Uber Request	CF35	With Uber API, requests Uber for user to go to the event location	No database interaction
Rate	Rate	CF36	Creates a pop-up window for	Tracks item status in

Activity			buyer and seller to evaluate each other whenever transaction is completed	the database (changes from reserved to resolved)
Reminder Activity	Reminder	CF37	Sends an email to the user when the event or item status is changed.	Tracks event or item status changes in database

5.2 Class and Variable list

5.2.1 Login Activity

Variable Name	Variable Type	Variable Description
bLogin	Button	A button for user to click and log in.
bRegister	Button	A button for user to click and register.
etUsername	EditText	A text field for user to enter username.
etPassword	EditText	A text field for user to enter password.
ivProfile	ImageView	A profile picture that is set by the user.
ivBackground	ImageView	A picture of the background of login page.
tvUsername	TextView	A text that display “username”.
tvPassword	TextView	A text that display “password”.

Method Name	Method Description
onCreate	This method initializes the login page and all of the items that will be displayed on the login page.
onClick	This method handles user’s click on login or register button by calling login, or register method.
login	This method authenticate the user’s login request by retrieving user information from the database.
register	This method create a new entry of user and related information in the

	database.

5.2.2 Profile Activity

Variable Name	Variable Type	Variable Description
ivProfile	ImageView	A profile picture that can be edited by the user.
bEdit	Button	A button for user to click and edit personal information.
bSave	Button	A button for user to save all the edited information.
bCanel	Button	A button for user to cancel edit request.
etPhone	EditText	A text field for user to edit phone information.
etEmail	EditText	A text field for user to edit email information.
etUsername	EditText	A text field for user to edit username information.
etPassword	EditText	A text field for user to enter password.
tvPhone	TextView	A text that display “phone”.
tvEmail	TextView	A text that display “email”.
tvUsername	TextView	A text that display “username”.
tvPassword	TextView	A text that display “password”.

Method Name	Method Description
onCreate	This method initializes the profile page and all of the items that will be displayed on the login page.
onClick	This method handles user’s click on edit profile and edit information by calling editProfile, or editInformation method.
editProfile	This method enables the user to change their profile picture.
editInformation	This method enables the user to change their personal information.

5.2.3 Menu Activity

Variable Name	Variable Type	Variable Description
toProfile	Button	A button for user to switch pages to the User's profile.
toExchange	Button	A button for the user to switch pages to the Exchange main page.
toEvents	Button	A button for the user to switch pages to the Events main page.
toManage	Button	A button for the user to switch pages to the Manage page.
Logout	Button	A button for the user to logout of the application.

Method Name	Method Description
onCreate	This method initializes the sliding menu with the five different buttons.
onClick	This method handles user's clicks on various buttons and changes the page according to the button that was clicked.
logout	This method handles the user's logout of the application.
navigation	This method is called by the onClick method and changes the current page to the page selected by the user clicking a button.

5.2.4 MainPage Activity

Variable Name	Variable Type	Variable Description
fEventPreview	Fragment	Android fragment that holds the Events preview section of the homepage.
fExchangePreview	Fragment	Android fragment that holds the Exchange preview section of the homepage.
ndSlidingMenu	Navigation Drawer	Navigation Drawer on every page that allows the user to switch between all pages.
tvSearchQuery	Text View	Search query that takes user input to search for a word or phrase.
fabSearch	Button	A button for the user to initiate a search query of the

		Events and Exchange entries.
fabCreateEvent	Button	A button for the user to create an event.
fabCreateItem	Button	A button for the user to create an Exchange listing.
fabEventLeft	Button	A button for the user to view a more popular event.
fabEventRight	Button	A button for the user to view a less popular event.
fabItemLeft	Button	A button for the user to view a more popular item.
fabItemRight	Button	A button for the user to view a less popular item.

Method Name	**Method Description**
onCreate	This method initializes the homepage and all of the items that will be displayed on the homepage.
onClick	This method handles user's clicks on various buttons and textView input changes the display according to the button that was clicked.
createEvent	This method handles the user's logout of the application.
createItem	This method is called by the onClick method and changes the current page to the page selected by the user clicking a button.
search	This method will search for a phrase or string that the user will enter view the tbSearchQuery using a search engine.
slide	This method allows the user to browse the top events or items, depending on the fragment slide is called in.

5.2.5 ListView Activity

Variable Name	**Variable Type**	**Variable Description**
sData	String[]	An array of data retrieved from the database.
aaDataList	ArrayAdapter <String>	An adapter to display the list's data.
sQuery	String	A string of select criteria

Method Name	Method Description
onCreate	This method initialize the ListView activity.
onItemClick	This method handles user's click on the ListView activity and ListView output changes the display according to the button that was clicked.
update	This method updates the ListView information after refreshing.

5.2.6 NewEvent Activity

Variable Name	Variable Type	Variable Description
createEvent	Button	A button for user to create an event after entering all the necessary event information.
etEventName	EditText	A textbox for user to enter event name.
etTime	EditText	A text field for user to enter event time.
etLocation	EditText	A text field for user to enter event location.
etCapacity	EditText	A text field for user to enter event capacity.
etEventDescription	EditText	A text field for user to enter event description.
ivEventImage	ImageView	An image field to upload event image.

Method Name	Method Description
onCreate	This method initialize the NewEvent Activity.
onClick	This method handles user's click on createEvent button by calling newEvent method.
newEvent	This method creates an new item in the database with event name, time, location, capacity, description, and image entered or uploaded by user.

5.2.7 NewItem Activity

Variable Name	Variable Type	Variable Description
createItem	Button	A button for user to create an new item after entering

		all necessary information.
etItemName	EditText	A text field for user to enter item name.
etPrice	EditText	A text field for user to enter item price.
etItemDescription	EditText	A text field for user to enter item description.
etContact	EditText	A text field for user to enter contact.
ivItemImage	ImageView	An image field for user to upload item image.

Method Name	**Method Description**
onCreate	This method initializes the NewItem Activity.
onClick	This method handles user's click on createItem button by calling newItem method.
newItem	This method creates a new item in database with item name, price, description, seller contact, and image entered or uploaded by user

5.2.8 Event Detail Activity

Variable Name	**Variable Type**	**Variable Description**
fabBookmark	FloatingActionButton	A button for participant to add the event to their bookmark.The button will appear as cancel bookmark if the user has already marked.
fabJoin	FloatingActionButton	A button for participant to join the event and add to calendar. The button will appear as cancel join if the user has already joined.
miPermission	MenuItem	A button in the action bar menu for the organizer to grant permit to the joined users.
miEdit	MenuItem	A button in the action bar menu for the organizer to edit the event information.
miDelete	MenuItem	A button in the action bar menu for the organizer to delete the event.
tvEventName	TextView	A text view that displays event name.

tvHashtag	TextView	A text view that displays event hashtag.
tvEventTime	TextView	A text view that displays event time.
tvEventDesceiption	TextView	A text view that displays event description.
tvLocation	TextView	A text view that displays event location.
tvCapacity	TextView	A text view that displays event capacity.
ibNavigate	ImageButton	A button that directs user to the Google map page.
ibUber	ImageButton	A button that directs user to the uber page.
ivEventImage	ImageView	An image view that displays the event poster.

Method Name	**Method Description**
onCreate	This method initializes the EventDetail Activity.
onClick	This method handles the user's click on the buttons in the page to call corresponding function or direct to corresponding page directly.
OnOptionsItemSelected	This method handles the user action in the action bar.
addToJoin	This method creates a new relationship in the join.
cancelJoin	This method deletes an existing relationship in the join.
addToBookmark	This method creates a new relationship in the bookmark and adds the event to the user calendar.
cancelBookmark	This method deletes the existing relationship in the bookmark and removes the event from the user calendar.
deleteEvent	This method deletes the event information.

5.2.9 Exchange Detail Activity

Variable Name	**Variable Type**	**Variable Description**
fabWatchlist	FloatingActionButton	A button for buyer to add the event to their watchlist.The button will appear as cancel watchlist if

		the user has already joined.
fabReserve	FloatingActionButton	A button for buyer to join the event and add to calendar. The button will appear as cancel join if the user has already joined.
fabReject	FloatingActionButton	A button for the seller to reject the current reservation.
fabConfirm	FloatingActionButton	A button for seller to confirm the current reservation.
miBlock	MenuItem	A button for seller to block the buyer who current reserves the item.
miEdit	MenuItem	A button in the action bar menu for the seller to edit the item information.
miDelete	MenuItem	A button in the action bar menu for the seller to delete the item.
tvItemName	TextView	A text view that displays item name.
tvType	TextView	A text view that displays item type.
tvPrice	TextView	A text view that displays item price.
tvSeller	TextView	A text view that displays seller and his/her rate.
tvDescription	TextView	A text view that displays the description of the item.
ivItemImage	ImageView	An image view that displays the item image.

Method Name	**Method Description**
onCreate	This method initializes the Exchange Detail Activity.
onClick	This method handles user's clicks on the buttons in the page to call corresponding function or direct to corresponding page directly.
OnOptionsItemSelected	This method handles the user action in the action bar.
addToWatchlist	This method creates a new relationship in the watchlist.

cancelWatchlist	This method deletes an existing relationship in the watchlist.
addToReservation	This method creates a new relationship in the reserve.
cancelReservation	This method deletes an existing relationship in the reserve.
rejectReservation	This method sets the status in the tuple in the reserve relationship and item entity.
confirmReservation	This method sets the status in the tuple in the reserve relationship and item entity.
block	This method creates a new relationship in the block.
delete	This method deletes the item information from the database.

5.2.10 Manage Activity

Variable Name	**Variable Type**	**Variable Description**
imEventJoined	ImageView	A cover image for the user's joined event section.
imEventFaved	ImageView	A cover image for the user's favorited event section.
imEventCreated	ImageView	A cover image for the user's created event section.
imItemBought	ImageView	A cover image for the user's bought item section.
imItemWatchlisted	ImageView	A cover image for the user's watchlisted item section.
imItemSold	ImageView	A cover image for the user's watchlisted item section.
numTabs	Int	A count indicating the number of tabs of the manage page, which should always equal to two in our case.
currentPage	viewPager	A reference to the current tab.

Method Name	**Method Description**
onCreate	This method initializes the Manage Activity
onClick	This method handles click events in the page to direct user to corresponding

	page directly.
getItem	Fetch the corresponding page based on user's position on tab
onTabSelected	Invoked when a new tab is selected in this page.
onItemSelected	Invoked when a section in a particular tab is selected.

5.2.11 Permission Activity

Variable Name	**Variable Type**	**Variable Description**
mAdapter	SimpleCurserAdapter	A adapter to bind views and get data from the database to ListView.
projection	String[]	An array of String to store projected attributes retrieved from the database.
selection	String[]	An array of String to store selected attributes retrieved from the database.

Method Name	**Method Description**
onCreate	This method initializes the Permission Activity.
grantAccess	This method is invoked whenever a user clicks on grant permission button for some participant.

5.2.12 Search Activity

Variable Name	**Variable Type**	**Variable Description**
abSearchBar	ActionBar	A input field for user to enter searching keywords.
fabSearch	Floating Action Button	A button beside the search bar to call the search function.
sUserInput	String[]	A local copy of user's input.
iResult	Integer List	A list of item/event ID's in the display order.

Method Name	Method Description
onCreate	This method initializes the Search Activity.
onClick	This method handles user's click on the buttons in the page to call corresponding function or direct to corresponding page directly.
inputAnalyze	This method analyzes the user input and separates it into strings.
search	This method performs the search algorithm.
sort	This method performs the sorting algorithm.
filter	This method performs the filter algorithm.

5.2.13 Google Map Activity

Variable Name	Variable Type	Variable Description
gacClient	GoogleApiClient	A local copy of Google map API.
lCurrentLocation	Location	The location of the user in Google map form.
lTargetLocation	Location	The location of the event in Google map form.

Method Name	Method Description
onViewCreated	This method initializes the Google Map when the activity is created.
Navigate	This method directs the user to the target location.
OnStart	This method calculates the route when the navigation starts.
OnStop	This method ends the navigation when the user reaches the location or stops navigation.
OnConnected	This method updates the user's location on the map and compares it to the route.

5.2.14 QR Activity

Variable Name	Variable Type	Variable Description

apiQRC	QRCAPI	A local copy of the QRC scanner API.
create	Button	A button that allows the user to generate a QRC code via a createQRC class.
scan	Button	A button that allows the user to scan a QRC code through the QRCAPI and display the encoded text.

Method Name	**Method Description**
onCreate	This method initializes the QRC scanner API.
onClick	This method processes the button that was clicked and calls the appropriate method.
display	This method calls the encode method and takes that output and displays it on the screen as an image.
scan	This method uses the user's phone and scans a QR code using the QRC scanner API.
decode	This method takes a valid QR code and outputs the text that was encoded.
encode	This method takes text and encodes the text into a QRC code.

5.2.15 Uber Activity

Variable Name	**Variable Type**	**Variable Description**
ubClient	UberApiClient	A local copy of Uber API.
lCurrentLocation	Location	The location of the user in Uber.
lTargetLocation	Location	The location of the event in Uber.

Method Name	**Method Description**
onCreate	This method initializes the Uber request when the activity is created.
Request	This method directs the user to the target location.

5.2.16 Rate Activity

Variable Name	Variable Type	Variable Description
Submit	Button	A button for user to submit the score.
Score	RatingBar	An rating bar for user to evaluate the buyer/seller.

Method Name	Method Description
onCreate	This method initializes the Rate Activity.
rate	This method is invoked whenever the item status changes from reserved to resolved.

5.3 API's Utilized

OAuth 2.0

OAuth 2.0 is a protocol that Google APIs use for authentication and authorization. In our application, with the implementation of OAuth 2.0 in our login page, we should be able to redirect our application window to a Google URL when users log in with their Case accounts. The URL includes query parameters that indicate the type of access being requested. Google handles the user authentication, session selection, and user consent. The result is an authorization code, which our application can then exchange for an access token. With this token stored in our application, we can access Google APIs such as Maps and Calendar mentioned below.

Google Maps

Google Maps is a standard API provided within android studio, which is available as a fragment inside an arbitrary activity. We will use this API in our application for creating a fragment in the events detail page whenever user wish to navigate to the specific event inside our application.

Uber

Uber provides a developer API named Deep Linking, which is for a non-native mobile application to launch specific actions inside the Uber app installed on that mobile device. We will use this API in our application to provide Uber with the drop-off location information, and launch the setPickup query, when our user choose to request Uber to the event location specified in our application.

QRScanner (Google Charts)

The QR scanner is a free API that is provided through Google in order to encode and decode QR

codes. In our application, we will only be decoding QR codes since we will manually encode a QR code when providing an identification code to the participants of events.

Google Calendar

Google Calendar is a free API that Google provides for access to a user's calendar. In our application, we will use this API to get the user's current calendar that is associated with their network ID. We will also be calling the API in order to add events that the user has chosen to attend to the user's associated network ID.

6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

To any user, aside from the login screen, our application will appear as a typical Android application. When logged in, the user will easily recognize the sliding menu button in the upper left corner as well as the search bar on the top of the screen. Most user interaction will involve touching the buttons on the screen, or in the case of forms and the search button, typing out information on the create form for the exchange and events create page and tapping the submit button. All user interaction on valid parts of the screen will correspond to a change on the display.

6.2 Screen Images

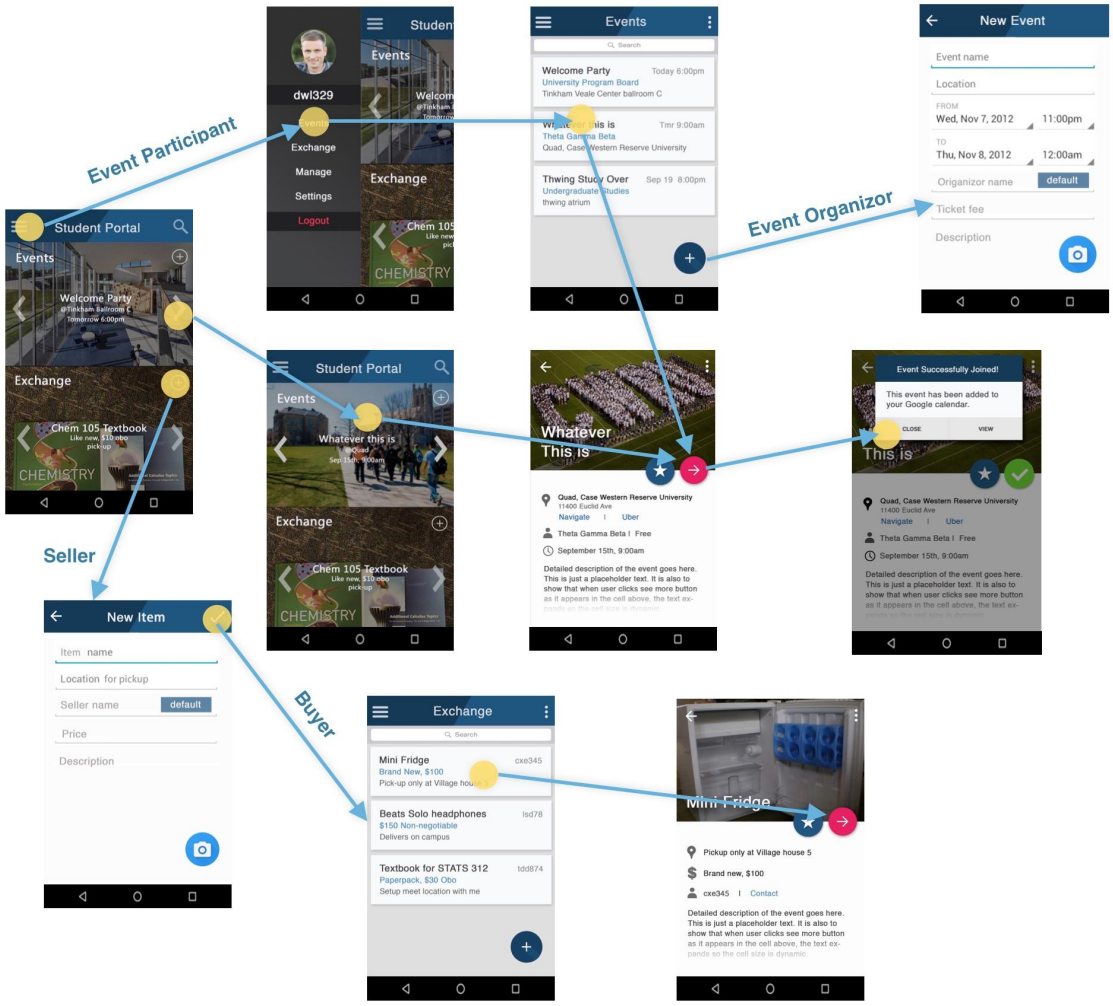

Figure 5: Student Portal UI Mockups

6.3 Screen Objects and Actions

6.3.1 Home Page

Screen Object	Location on page	Action
Sliding menu button	Action bar, upper left corner	Brings out the sliding menu (navigation drawer), navigates user to other pages.
Search button	Action bar upper right corner	Brings out the search page that allows user to search all entries across the platform.
Events navigation arrows	Inside events section	Within the homepage, navigates the user through featured events recommended for that specific user.
Events add-new button	Upper right corner of	Directs the user to the create new event page.

	events section	
Events image and text display	Inside events section	Directs the user to the corresponding detail view page of the event displayed on the homepage.
Exchange navigation arrows	Inside exchange section	Within the homepage, navigates the user through featured items recommended for that specific user.
Exchange add-new button	Upper right corner of exchange section	Directs the user to the create new exchange item page.
Exchange image and text display	Inside exchange section	Directs the user to the corresponding detail view page of the item displayed on the homepage.

6.3.2 Events / Exchange list view page

Screen Object	**Location on page**	**Action**
Sliding menu button	Action bar, upper left corner	Brings out the sliding menu (navigation drawer), navigates user to other pages.
Drop down button	Action bar, upper right corner	Brings out the drop down menu, which allows user to navigate to event /items they have joined, favorited or created.
List cells	Inside Listview fragment	Directs the user to the corresponding detail view page of the event/item displayed in that cell.
Search bar	Below action bar	Prompt user for search keywords. Directs user to the detail view page with search results.
Add button	Lower right corner	Directs the user to the create new event or exchange item page depending on the section the user is originally under.

6.3.3 Events / Exchange detail view page

Screen Object	**Location on page**	**Action**
Back button	Upper left corner	Navigates the user back to the page before the detail view page is visited.
Drop down button	Upper right corner	Brings out the drop down menu, which allows user to cancel join/reservation, contact seller (buyer only), edit post information (seller and organizer only), view guest list and scan participant QR codes (organizer only).
Star button	Left button under	Allows user to favorite/add to watchlist an

	image view	event/item depending on the section the user is under.
Go button	Right button under image view	Allows user to join/reserve an event/item depending on the section the user is under.
TextView	Lower part of the activity	Displays the detailed description of the selected event/item.

6.3.4 Create new event / exchange item page

Screen Object	**Location on page**	**Action**
Back button	Upper left corner	Navigates the user back to the page before the create new event/item page is visited.
Editable text fields	Over the entire page	Placeholders are replaced by user input as user types in a certain field.
Date picker	Middle of the page (new event page only)	Brings out a date picker view which allows user to scroll and select date / time.
Default button	Right side of the organizer / seller name text field	Auto-fills the field with current user's network ID for the convenience of user.
Camera button	Lower right of the page	Launches the image picker that prompts the user to choose a photo from camera roll and upload.

7. REQUIREMENTS MATRIX

Req. ID	Requirement Description	Requirement Subtype	Actors	Component ID
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.	Homepage	Logged In Uers	CF04, 08, 09
2	Account usernames must be unique.	Homepage	Logged In Uers	CF09
3	The user should be able to change the username provided it is unique.	Homepage	Logged In Uers	CF09
4	After the login, the homepage should	Homepage	Logged In Uers	CF11

	display the sliding menu button on the upper left corner, which can navigate to the Events, Exchange, Manage and Settings page.			
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.	Homepage	Logged In Uers	CF14
6	The Events section will consist of current event's name, location and time, overlaid on event's cover image.	Homepage	Logged In Uers	CF03, 14
7	The Events section will also consist of a left and right arrow to allow users to browse other featured events.	Homepage	Logged In Uers	CF12
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.	Homepage	Logged In Uers	CF15
9	The Exchange section will consist of current item's name, price and a brief description, overlaid on the item's picture.	Homepage	Logged In Uers	CF15
10	The Exchange section will also consist of a left and right arrow to allow users to browse other items on sale.	Homepage	Logged In Uers	CF12
11	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.	Homepage	Logged In Uers	CF03, 05,15
12	The user (seller) should be able to edit or update item information. The system will send notifications to other users	Exchange	Sellers	CF21,25

	who have reserved the item.			
13	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.	Exchange	Sellers	CF15
14	The user (seller) should be able to receive notifications from the system automatically if the item posted is reserved.	Exchange	Sellers	CF15
15	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.	Exchange	Sellers/Buyers	CF21, 25
16	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.	Exchange	Sellers/Buyers	CF04, 05, 15
17	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.	Exchange	Sellers/Buyers	CF15
18	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.	Exchange	Sellers/Buyers	CF15
19	The user (buyers) 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 user to use the system, the list will display the latest	Exchange	Sellers/Buyers	CF27, 28

	items posted by other users (sellers) to sell.			
20	The user (buyer) should be able to search an item by keywords and check all the available and reserved items related to the keyword.	Exchange	Buyers	CF27, 29
21	The user (buyer) should be able to filter the search results by price or time.	Exchange	Buyers	CF28,29
22	The user (buyer) should be able to filter the search results by seller rate.	Exchange	Buyers	CF28,29
23	The user (buyer) should be able to reserve an available item. The status of item will change from available to reserved.	Exchange	Buyers	CF21
24	The user (buyer) should be able to cancel the reservation on items. The status of the item will be changed from reserved to available.	Exchange	Buyers	CF23
25	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.	Exchange	Buyers	CF20
26	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.	Exchange	Buyers	CF37
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.	Exchange	Buyers	CF36
28	The user (participant) should be able to	Events	Participants	CF27,28

	view and search campus events that are yet to be held.			
29	The user (participant) should be able to join an open event and add it to calendar with join button.	Events	Participants	CF17
30	The user (participant) should be able to cancel his/her join of the event any time before the event starts.	Events	Participants	CF18
31	The user (participant) should not see events that have already occurred on their main event page.	Events	Participants	CF18
32	The user (participant) should be able to indicate interest in events.	Events	Participants	CF17
33	The user (participant) should be able to view events he/she is interested in the a separate events page.	Events	Participants	CF17
34	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.	Events	Event Organizers	CF14
35	The user (event organizer) should be able to see joined participants for each events they created.	Events	Event Organizers	CF25
36	The user (event organizer) should be able to give permission to joined participants and add them to participants list after permission is assigned.	Events	Event Organizers	CF25
37	The user (participant) should be able to view their permission status of joined events in joined events page.	Events	Participants	CF25,26

38	The user (event organizer) should be able to scan user's QR code in order to grant a user access at event time.	Events	Event Organizers	CF31,32,33,34
39	The user (participant) should be able to launch Google Maps from Student Portal with routed destination set to the event location.	Events	Participants	CF30
40	The user (participant) should be able to launch Uber from Student Portal with destination set to the event location.	Events	Participants	CF30,35
41	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.	Performance	System	CF24
42	The system should display confirmation messages to users within 5 seconds after the user submits information to the system.	Performance	System	CF10
43	Search results should be displayed within 5 seconds.	Performance	System	CF27,29
44	The system must be secured to ensure that unauthorized users cannot login and access the system.	Security	System	CF01, 06, 09
45	The system will use OAuth2 for accessing Google APIs, and administrators of the system will not have access to user login credentials.	Security	System	CF01, 06, 09
46	The system will not reveal the private information of other users.	Security	System	CF06, 09
47	The Student Portal system should be available to users on the corporate internet 24 * 7.	Software Quality	System	CF07

48	The system should generate a backup copy of the application data and log files.	Software Quality	System	CF04
49	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.	Software Quality	System	CF03, 05