KSU GO

Software Design Document

Version 3.0

Team 5

Albert Lim

Chase Godwin

Patrick Hilerio

Dayton Chamberlin

Anthony Schell

Nick Wilson

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 23/Feb/18 | 1.0 | Initial SDD is established | Albert Lim  Chase Godwin  Patrick Hilerio  Dayton Chamberlin  Anthony Schell  Nick Wilson |
| 26/Feb/18 | 2.0 | The SDD is revised. | Albert Lim  Chase Godwin  Patrick Hilerio  Dayton Chamberlin  Anthony Schell  Nick Wilson |
| 27/Feb/18 | 3.0 | SDD reviewed & complete | Albert Lim  Chase Godwin  Patrick Hilerio  Dayton Chamberlin  Anthony Schell  Nick Wilson |

Table of Contents

[**Introduction**](#_y7qhj7njqye3) **5**

[Purpose](#_ly360g3sd5m4) 5

[Scope](#_n0uz1wt7jeze) 5

[Definitions, Acronyms, and Abbreviations](#_i0to7h8yct5l) 5

[References](#_n28l1dth03i) 6

[**Design Overview**](#_ya2r8yr236g0) **6**

[Introduction](#_x9smzfmj01z) 6

[System Architecture](#_m017twbs61z0) 7

[System Interfaces](#_vmia88jd4428) 8

[Constraints and Assumptions](#_kkkxingp782d) 8

[**System Object Model**](#_j9mtt0cpi91r) **8**

[Introduction](#_e99vgsilu88n) 8

[Subsystems](#_61mifs4vvcys) 9

[Subsystem Interfaces](#_yj6zq5i1hdoi) 28

[**Object Descriptions (Object Design)**](#_rynk1eesyn82) **31**

[Objects in Login subsystem](#_8vpa559n0uvm) 31

[Objects in Home subsystem](#_prwsyfh30dfo) 32

[Objects in Bus subsystem](#_iz60t0m5xth1) 33

[Objects in Contact Directory subsystem](#_kxma41rdu9jc) 34

[Objects in Emergency subsystem](#_b8quyckntf64) 34

[Objects in D2L subsystem](#_qn7hq9us18z7) 35

[Objects in Owl Life subsystem](#_615ylt3u9ajy) 37

[Objects in Handshake subsystem](#_o8auoiexc1c0) 38

[Objects in NewsFeed subsystem](#_tusnvkaucbs5) 39

[Objects in Basic Campus Map subsystem](#_b3k2s6dgxh18) 40

[Objects in Interactive Campus Map subsystem](#_myyt0d7m2ass) 40

[Objects in All Events subsystem](#_5l8t648s1qg) 42

[**Object Collaboration (Process View)**](#_us8zscf4v7d8) **43**

[Objects in Login subsystem](#_iqapbbs3ifyd) 43

[Objects in Emergency subsystem](#_tgy2kzbvhaqk) 43

[Objects in Contact Directory subsystem](#_2evws3b15klc) 44

[Objects in D2L subsystem](#_1w4f354ogt1) 44

[Objects in BOB Bus subsystem](#_5c2sbtcghhef) 44

[Objects in News Feed subsystem](#_m3bltzzfdz5a) 45

[Objects in Owl Life subsystem](#_qfjc5gnzxrou) 45

[Objects in Handshake subsystem](#_bdo037x70bz8) 45

[Objects in Basic Map subsystem](#_5u8rh5ua82es) 46

[Objects in Interactive Map subsystem](#_tkzg1kqa3ds) 46

[Objects in All Events subsystem](#_pfqt6mkdckou) 46

[**Dynamic Model**](#_x2a8p85ko2ya) **47**

[State Diagrams](#_tir6u2eayzge) 47

[Sequence Diagrams](#_vcyr3n2tjlo2) 49

[**Data Design**](#_b2gu04tt8mjf) **56**

[Entity Relationship Diagram](#_zch34xyqty2p) 56

[Relational Schema](#_6o4gvclwth1z) 57

[**Non-functional Requirements**](#_pwksqn3ywyvd) **58**

[**Supplementary Documentation**](#_n9t0mu4pi77u) **58**

Software Design Document

# Introduction

The software requirements of KSU Go cover a number of in-app functions including interactive maps, campus emergency services, and real-time transportation information. The scope of these functions is defined by our RDD, comprising many useful outlets provided by KSU for students and faculty alike. The purpose of these applications is to provide ease of access to the aforementioned outlets, with a user interface implemented to house and display certain facets of the apps.

## Purpose

The software exists as a mobile application on an Android device. The execution of said app primarily loads with a login screen, with prompts for the user’s KSU ID and password relative to the app. The login screen contains alternative options for forgotten passwords and usernames, and should redirect the user in the case of the app being unavailable. Following logging in, the system will provide a menu housing the main applications of the project, as well as an “announcements” area for pertinent/subscribed information. The current interface to interact with the program’s apps is a single contained activity panel, with links to the following to be fully implemented: emergency campus contact, contact directory, D2L, KSU social feeds, OwlLife (Clubs & Organizations), Handshake (Career Services), BOB, and standard Kennesaw/Marietta Maps. Bonus functionality will contain interactive campus map and all events/ announcements notification (collecting all emails regarding events into on collection). Contingent on the accessibility of the features, certain aspects of the application may be comprised of hard-coded or mock data to simulate, such as emergency services or direct access to student grades and schedules.

## Scope

The SRS is applicable to any KSU students, faculty, or guests, and is accessible through Android mobile devices. The application and its features consist of a structure in Android Studio, implementations from Google mapping and database subsystems, and is associated with the KSU Go Use Case model (Fig. 1.0) below in the SRS. All entities affected and influenced by this document are contained within the application, and cover basic.

## Definitions, Acronyms, and Abbreviations

BOB - Big Owl Bus

D2L - Desire 2 Learn

GPS - Global Position Service

ID - Identification

KC - Kennesaw Campus

KSU - Kennesaw State University

MC - Marietta Campus

RDD - Requirements Description Document

UI - User Interface

UITS - University Information Technology Services

## References

Android Studio & SQLite -

* <https://developer.android.com/studio/index.html>
* <https://www.sqlite.org/releaselog/3_22_0.html>

Google Maps Android API Help -

* <https://developers.google.com/maps/documentation/android-api/start>
* <https://www.androidtutorialpoint.com/intermediate/android-map-app-showing-current-location-android/>

Android Studio Research & Troubleshooting -

* <http://www.c-sharpcorner.com/article/how-to-be-working-with-multiple-activities-and-navigate-the-activities-in-androi/>
* <https://developer.android.com/training/basics/firstapp/starting-activity.html>
* <https://developer.android.com/training/basics/firstapp/running-app.html>
* <https://developer.android.com/training/keyboard-input/commands.html>
* <https://stackoverflow.com/questions/11531845/how-to-go-about-creating-an-interactive-map-in-android>

Amazon Relational Database Service -

* <https://aws.amazon.com/rds/?ft=n>

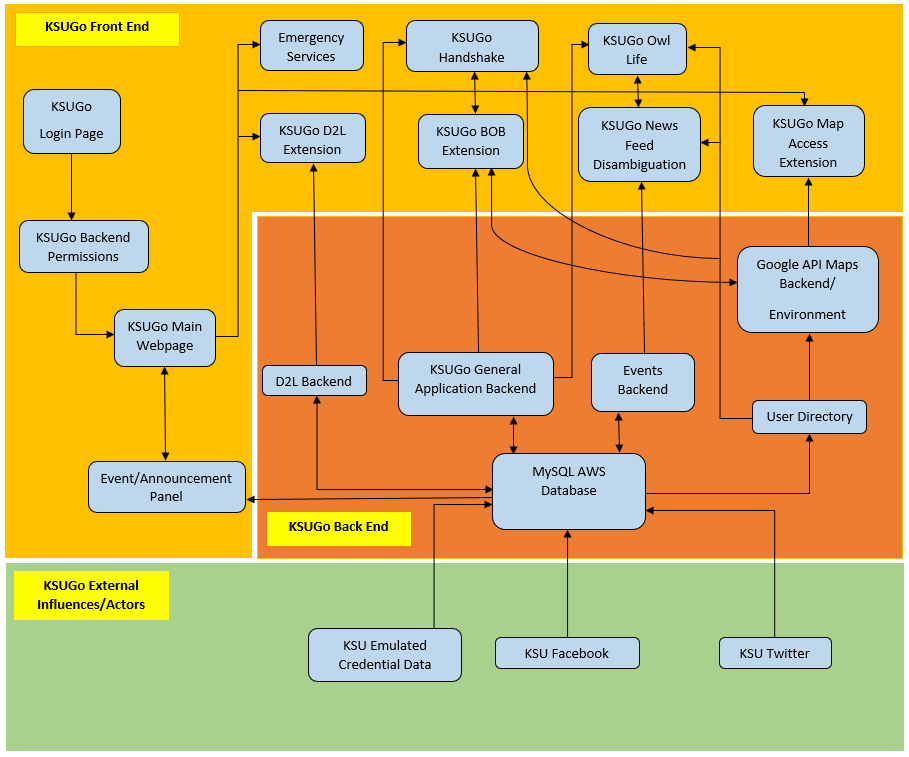
# Design Overview

## Introduction

The design scheme of KSU Go was decided to be the Android Studio-based platform backed by an SQL database, with a host of applications within all supported through a collective homepage. All content is tied to the homepage as well as the database, with users entering through the login page and external content being collected through the database for display. The System architecture diagram illustrates the functional objects at play in the project, as well as the relationships they have with other objects; each of these objects is detailed in the use cases provided in the System Object Model. The system interfaces all operate through Android phones, and thus are composed of the variety of UI that are part of KSU Go. Each backend function is tied to its respective UI outlets and is further detailed in the analysis classes regarding their implementation.

## System Architecture

The System Architecture of KSU Go is based off of the front-end applications that make up the majority of the users’ experience, and the backend processes associated with the database and executing the application’s primary tasks. The frontend is made up of the login, the homepage and its applications, and any transition screens relative to applications or UI. The database acts as the primary information network that pulls and displays information throughout the program. It also acts as the gateway between the user experience and the external content, with the database updating the information as it checks each outlet website or function.

**System Architecture Diagram**

The design of the program and its subsystems is shown above, illustrating the three primary regions that our application covers; the diagram also utilizes arrows to portray which entities are communicating in which direction. The execution of the application always begins at the Login Page for KSUGo, followed by a classification of the account based on the permissions of the user relative to Kennesaw State University. The method to change the activity to the main menu is invoked, and the user is presented with the basic applications within KSUGo. Each application is independent of the others on the main menu, but communicates with a number of backend objects that handle the functionalities of each activity. Both the interactive map and BOB features utilize the Google API backend unique to programs implementing a map, inheriting the user directory for information relative to the campus and students. All information from the backend is handled by the database, which receives and stores data from external sources connected to KSU. The database communicates with nearly every aspect of the application, and is essential for the retrieval of outputs in nearly every user activity.

## System Interfaces

System Interfaces of KSU Go Application is mainly frontend talking to back end. As features need information the will interact with the application. The application will fulfill this request by requesting data from the database through an API. Other interfaces include interactive map using Google Maps API to show current location and other notable locations.

## Constraints and Assumptions

KSU Go is a mobile application on an Android System (At Least Android OS Version:5.0/ Lollipop). Assumption is users and database can work with this. Mobile Phone must be able to perform application performance as well as have access to key mobile phone functions, such as GPS, ability to call, and ability to access data over carrier services and/or wireless connectivity. If phone does not meet these assumptions and/or dependencies the application may not perform as intended or perform at all.

# System Object Model

## Introduction

This section of the document goes into the specifics of our system. The interactions of users and subsystems within the application will be discussed through a Use-Case Model, as well as an explanation of who each of our end users are. This includes both surface users (such as people using the application) and back end users (External Systems connected to the application). We layout assumptions and dependencies for our application to function properly so that we can design with them in mind. Section 2.3 goes into the specific design functionalities of the system and how those systems are made up, as well as how they specifically interact with each other in technical terms rather than functions as described in Section 2.2. We also explore how the actual design UI looks for each function of the application. Last but not least, we have other technical requirements and support information provided.

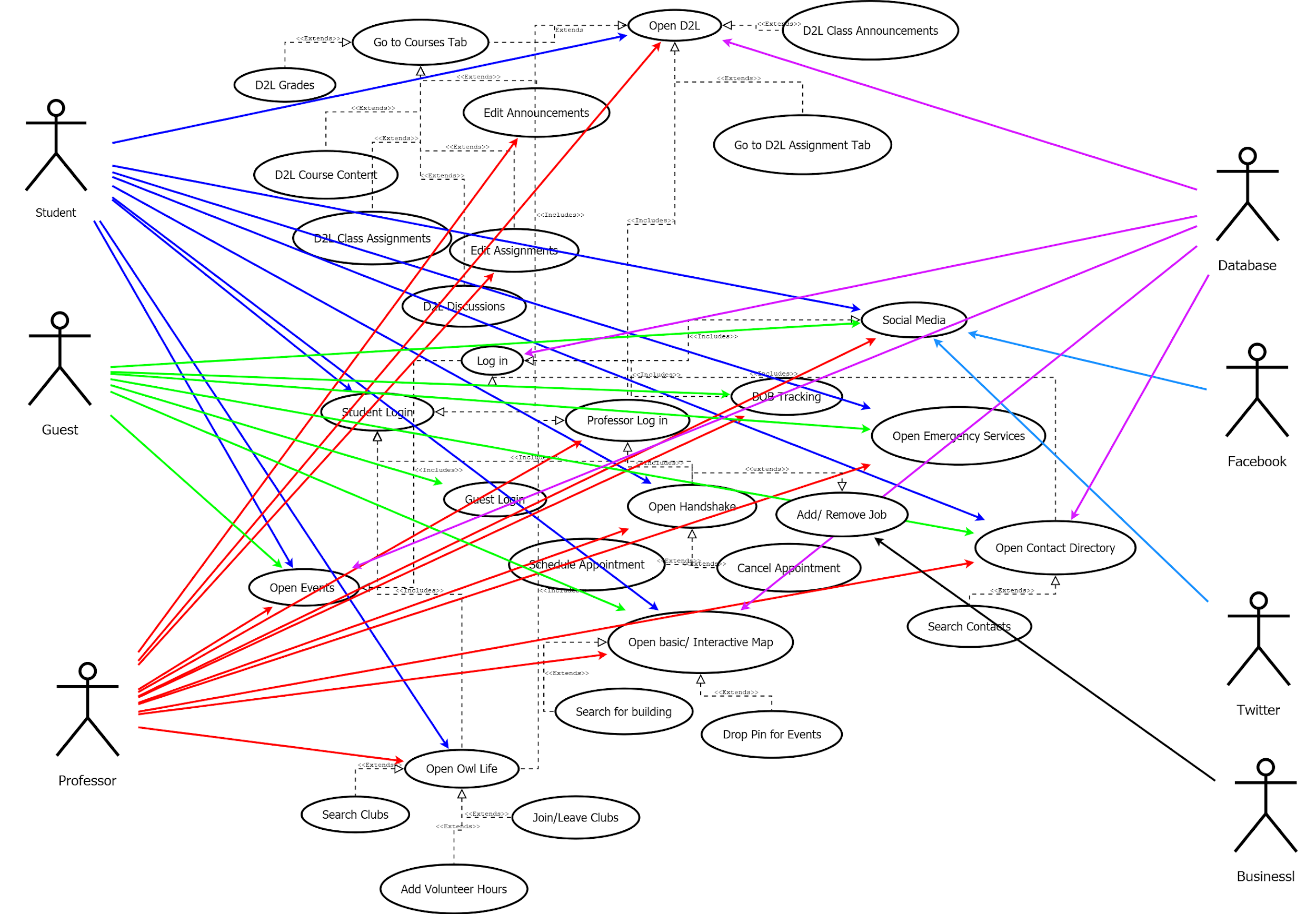
## Subsystems

This section provides us with a Use Case Model for the entirety of the Application including actors that interact with the system. Actors include KSU Students, KSU Staff, Non-Students, Database, and Twitter/Facebook. Subsystems include D2L, Emergency Services, Contact Directory, Basic Campus Map, Interactive Campus Map, Handshake, KSU Social Feeds, BOB Bus Routing, Event/Notification Center, Owl Life, and Login System.

KSU Students are users that interact with all Functionalities of the application. KSU Faculty are users that interact with each functionalities to customize what KSU Students see, as well as using some functionalities as all other Users would. Non-Students/Faculty are users with limited scope of the applications as only select functionalities would be present for them to use. Anything that requires authentication would be unavailable to these users. Database would be an actor that is notified by the application when the application is in need of some information. This can include login in information, class schedule data, and information regarding functions.

NOTE:

* Blue arrows are for Student (actor).
* Green arrows are for Guest (actor).
* Red arrows are for Staffs (actor).
* Purple arrows are for Database (actor).
* Light blue arrows are for Facebook and Twitter (actors).
* Black arrow is for the business (actor)



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access Login | | **ID:** A001 | | **Priority:** High |
| **Primary actor:**  Application users, students/faculty, guests | **Source:**  Application execution | | **Use case type:**  Business | |
| **Interested Stakeholders:** N/A | | | | |
| **Brief description:**  Primary interface for user to input credentials and access application, or to recover username or password. | | | | |
| **Precondition:**  User has an Android Phone.    **Trigger:**  App startup | | | | |
| **Relationships:** Access to main menu and features  **Include:** N/A  **Extend:** N/A  **Depends on:** Main menu | | | | |
| **Typical flow of events:**  User opens KSU Go application on their phone, and is prompted to enter their username and password on the screen. The user enters the information, and is then either granted access to the main menu and ancillary applications, or is told their information is invalid. The user also has the option to recover their username or password via button. | | | | |
| **Assumptions**  Assumes keyboard/touch screen access | | | | |
| **Implementation Constraints and Specifications:**  Students and staffs must use their netID and password. Since actual user data from UITS are unavailable, dummy accounts will be used for KSU GO implementation. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access Homepage | | **ID:** A002 | | **Priority:** High |
| **Primary actor:**  All application users | **Source:**  Access Login page | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  Acts as the main menu for our KSU Go Application | | | | |
| **Precondition:**  The user must have a valid account with KSU or log in as a guest to be able to access this page.  **Trigger:**  Once the user enters valid credentials on the login page, authentication will then redirect the user to this page. | | | | |
| **Relationships:** Holds the directories for all other use cases, with exception to the login page  **Include:** N/A  **Extend:** N/A  **Depends on:** Login page | | | | |
| **Typical flow of events:**  After logging in, the user will be met with a screen displaying the applications described in our proposal and this SRS. The user will see a dialog window at the top of the page displaying announcements depending on their subscriptions in-app, and can scroll down to access the other applications. The user can also access the emergency services, which is on the side of the page and can be accessed at any time if the user chooses. | | | | |
| **Assumptions**  The announcement tab must be integrated and connected to the database as well as the internet to allow synchronization of current information. | | | | |
| **Implementation Constraints and Specifications:**  All applications aside from the announcement tab (on account of being implemented as a portion of the menu) must be contained on one consistent page, such that the user does not have to look for apps in different locations. The menu must be able to accommodate for different dimension screens to maintain the organizational structure of the menu & announcements. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access D2L | | **ID:** A003 | | **Priority:** High |
| **Primary actor:**  Student/Professor | **Source:**  Student/Professor selecting D2L function | | **Use case type:**  Business | |
| **Interested Stakeholders: N/A** | | | | |
| **Brief description:**  Student or professor opens up the D2L within KSU Go. Students and professors will be able to check assignment due dates, course content, any announcements, etc. | | | | |
| **Precondition:**   * User has to be logged in as either a student or Professor * App is connected to the internet * User has android phone     **Trigger:**  Student/professor selects D2L on the homepage of the app | | | | |
| **Relationships:**  **Include:** Log in Professor/Student  **Extend:** Homepage use Case  **Depends on:** Login Use Case | | | | |
| **Typical flow of events:**   1. Student or professor opens D2L from within KSU Go 2. The application verifies with server that the student or teacher is able to use D2L 3. Application then checks to see if user is a student or professor 4. Application with then pull information from the server that is required for D2L, such as class list, announcements, etc. 5. After verification, both students and professors will be shown the News tab of D2L 6. Student:    1. Student goes to Assignment tab       1. Tab displays all assignments from each class organized by date then by time    2. Student Goes to course tab       1. Student can then select class       2. After selection of class, app will all student to select one of the following: Content, Assignments for that class, Discussion Board, Announcements for that class, and grades for that class       3. The content page will allow the student to see all modules for the selected class. 7. Professor:    1. Professor can make announcement to all classes here    2. Professor Goes to assignment tab       1. Tab displays all assignments due for each of his/her classes       2. Professor will be able to adjust due dates or release of each assignment as well    3. Professor goes to course tab       1. Similar to student. The Professor will be able to select course, and then go to the same functions.       2. Professors will be able to make announcements to specific classes in the announcement tab | | | | |
| **Assumptions**   1. The student or professor has an internet connection. 2. The student of professor has class or classes that they are attending 3. The database has the required information for the student/professor 4. The student/professor is logged in | | | | |
| **Implementation Constraints and Specifications:**  Since actual user data from UITS are unavailable, dummy accounts will be used for KSU GO implementation. Dummy accounts will specify courses enrolled, announcements, and schedule (such as test date). | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access BOB Bus Route | | **ID:** A004 | | **Priority:** High |
| **Primary actor:**  Application users, students/faculty, guests | **Source:**  Application user selecting the BOB Bus Route feature. | | **Use case type:** Business | |
| **Interested Stakeholders:** N/A | | | | |
| **Brief description:** Application user will open the BOB Bus Route, and will be able to see the routes and check the locations of currently running buses. | | | | |
| **Precondition:**   * User has to be logged in as either a student, professor, or guest. * App is connected to the internet * User has android phone   **Trigger:**  Application user selects the BOB Bus Route feature inside KSU Go. | | | | |
| **Relationships:**  **Include:** N/A  **Extend:** Log in Professor/Student/Guest  **Depends on:** Login User Case | | | | |
| **Typical flow of events:**   1. User opens BOB Bus Route from within KSU Go. 2. The application opens a web page to KSU’s official BOB Bus Route page. | | | | |
| **Assumptions**   1. The user has an internet connection. 2. The database has the required information for the user. 3. The user is logged in. | | | | |
| **Implementation Constraints and Specifications:**  The permission to access BOB Bus implementation is in progress. If the BOB Bus integration is not allowed. This implementation will be integrated by calling http://kennesaw.edu/mobile/bob.php. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access Directory | | **ID:** A005 | | **Priority:** High |
| **Primary actor:**  Application users, students/faculty, guests | **Source:**  Application user selecting the Contact Directory Feature. | | **Use case type:**  Business | |
| **Interested Stakeholders:** N/A | | | | |
| **Brief description:**  Application user accesses the Contact Directory feature inside KSU Go. Users will be able to search for staff and populate information based upon the search. | | | | |
| **Precondition:**   * User has to be logged in as either a student, professor, or guest. * App is connected to the internet * User has android phone   **Trigger:**  Application user selects the Contact Directory feature inside KSU Go. | | | | |
| **Relationships:**  **Include:** N/A  **Extend:** Log in Professor/Student/Guest  **Depends on:**  Login User Case | | | | |
| **Typical flow of events:**   1. User opens Contact Directory from within KSU Go. 2. User inputs desired search terms. 3. If information in the database matches the search terms, the contact information is then returned to the user. | | | | |
| **Assumptions**   1. The user has an internet connection. 2. The database has the information requested. 3. The user is logged in. | | | | |
| **Implementation Constraints and Specifications:**  The contact or department list will be populated using mock data stored in database or it will be integrated by calling <http://directory.kennesaw.edu/dir/people/>. Students, staffs, and guests will be  greatly benefited from this feature because it will help the users to find certain staffs or departments  for either help or collaboration. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access Emergency Services | | **ID:** A006 | | **Priority:** High |
| **Primary actor:**  Application users, students, faculty, guests | **Source:**  Application user selecting the Emergency Services Feature. | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  Application user accesses the Emergency Services feature inside KSU Go. Users will be able to choose between Emergency and Non-Emergency calls. | | | | |
| **Precondition:**   * User has to be logged in as either a student, professor, or guest. * App is connected to the internet * User has android phone   **Trigger:**  Application user selects the Emergency Services feature inside KSU Go. | | | | |
| **Relationships:**  **Include:** N/A  **Extend:** Log in Professor/Student/Guest  **Depends on:** Login User Case | | | | |
| **Typical flow of events:**   1. User opens Emergency Services icon in KSU Go. 2. User chooses “Emergency” or “Non-Emergency” 3. The correct number is input into the dialer. | | | | |
| **Assumptions**   1. The user is logged in. 2. The user has an internet connection. 3. The user has an android phone. 4. The user has a cellular signal. | | | | |
| **Implementation Constraints and Specifications:**  Emergency call will benefit those are in dire need for help such as ambulance or counseling. The non-emergency call will benefit those are need helps for non-emergency situation such as opening  building when user is locked out of the entrance. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Owl Life | | **ID:** A007 | | **Priority:** High |
| **Primary actor:**  Students, faculty (event planning/management) | **Source:**  Access main menu | | **Use case type:**  Business | |
| **Interested Stakeholders:**  KSU Owl Life Events/Organizations | | | | |
| **Brief description:**  Owl Life application allows the students that use KSU Go to seamlessly access KSU events and the organizations that set them up, as well as play an interactive part with the aforementioned. The application is integral to our purpose, as we seek to get students more invested and interested in university events with other students. | | | | |
| **Precondition:**  The KSU Go application must have access to Owl Life via link, as well as contain functions within the Owl Life application that allow the student to organize and interact with Owl Life provisions.  **Trigger:**  This application executes upon clicking the icon on the main menu for Owl Life. | | | | |
| **Relationships:**  **Include:** N/A  **Extend:** N/A  **Depends on:** Login use case, Interactive Map use case, Main Menu | | | | |
| **Typical flow of events:**  Upon launching the application, the user will be able to choose from a selection of options including ongoing events, events that are being planned and the ability to create an event depending on their permissions. The user will then be able to select events and view them via the Interactive Map application, and can also view upcoming events on a provided calendar. The user will also be able to view organizations that are listed by Owl Life, which can be accessed from the primary activity screen of the application. | | | | |
| **Assumptions**  This use case assumes not only that the main menu is operable, but that a connection has been established with the Interactive Map as well as the Owl Life website to provide up-to-date information on events and organizations. | | | | |
| **Implementation Constraints and Specifications:**  The Owl Life application cannot access certain personal information relative to Owl Life, as it would require for students to provide confidential credentials. The expectation of the Owl Life app is to circumvent this by making more public groups so guests and faculty are able to benefit from this. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access Handshake | | **ID:** A008 | | **Priority:** High |
| **Primary actor:**  Students | **Source:**  Access main menu | | **Use case type:**  Business | |
| **Interested Stakeholders:**  Students/faculty, employers | | | | |
| **Brief description:**  The Handshake application in KSU Go is an extension of the handshake feature provided by KSU, allowing students to keep up to date in finding job opportunities. The extent of the app for KSU Go is to include notifications, as well as interacting with other applications such as the announcements and maps to create a stronger connection between Handshake and the other opportunities present at KSU. | | | | |
| **Precondition:**  The user must be a student at KSU if they are utilizing Handshake primarily to search for jobs, but must also have an existing Handshake account to access the full features of the KSU Go application.  **Trigger:**  This application executes upon clicking the Handshake icon on the main menu. | | | | |
| **Relationships:**  **Include:** N/A  **Extend:** N/A  **Depends on:**  Interactive map use case, Login use case, Main Menu | | | | |
| **Typical flow of events:**  Upon launching the application, the user is prompted to make a Handshake account and redirected to the website to access the app. The user is then provided an activity screen displaying any notifications from handshake, as well as the ability to view their profile. The user is also able to view the website version of Handshake, and can access it below the notifications. Any events covered by Handshake will appear along with the notifications and implement the interactive map to pin locations on KSU Campuses. | | | | |
| **Assumptions**   1. User is a Student 2. User has Internet Connection 3. User has a Handshake account | | | | |
| **Implementation Constraints and Specifications:**  The application contains certain aspects that provide links to Handshake resources that are unavailable for KSU Go, along with the in-app interface and activities. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access News Feed | | **ID:** A009 | | **Priority:** High |
| **Primary actor:**  Application users, students, faculty, guests | **Source:**  Application user selecting the News Feed Feature. | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  Application user accesses the News Feed feature inside KSU Go. The user will be able to view Facebook and Twitter KSU news feeds. | | | | |
| **Precondition:**   * User has to be logged in as either a student, professor, or guest. * App is connected to the internet * User has android phone   **Trigger:**  The application user selects the News Feed feature inside KSU Go. | | | | |
| **Relationships:**  **Include:** N/A  **Extend:** Log in Professor/Student/Guest  **Depends on:** Login use case | | | | |
| **Typical flow of events:**   1. The user clicks on the News Feed icon in KSU Go. 2. The user can select or view Facebook and Twitter news feeds. 3. Optionally, the browser will open to the particular feed selected. | | | | |
| **Assumptions**   1. The user is logged in. 2. The user has an internet connection. 3. The phone is an android device. | | | | |
| **Implementation Constraints and Specifications:**  The Facebook posts will be based on the KSU Facebook account, and the Twitter tweets will be based on KSU twitter account. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access Basic Campus Map | | **ID:** A010 | | **Priority:** High |
| **Primary actor:**  All application users*.* | **Source:**  Main Menu | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  A static map provided along with the interactive map to display the visual layout of KSU rather than emphasizing database information. | | | | |
| **Precondition:**  The only precondition is the presence of up-to-date images of KSU.  **Trigger:**  Application executes upon clicking the Basic Campus Map icon on the main menu. | | | | |
| **Relationships:**  **Include:**  **Extend:** Login of Student/Faculty/Guest  **Depends on:**  Login use case, Main Menu | | | | |
| **Typical flow of events:**   1. User opens Basic Campus map Icon. 2. Picks either Marietta or Kennesaw Map from a Drop Down Menu. 3. Depending on previous choice, Map of campus appears. | | | | |
| **Assumptions**   1. Either Guest of KSU or Student/Faculty of KSU | | | | |
| **Implementation Constraints and Specifications:**  The Marietta and Kennesaw Campus Map will be integrated by calling the backend where the database contains map layouts for both campuses. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access Interactive Campus Map | | **ID:** A011 | | **Priority:** Medium |
| **Primary actor:**  Student, Professor, Guest | **Source: User selecting interactive map from home screen** | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  Students, faculty, or guests will be able to search for buildings on either the main Kennesaw campus or the Marietta campus. The interactive map will give the community of KSU and any guests a better experience as they navigate the campuses. | | | | |
| **Precondition:**  N/A  **Trigger:**  Selecting the interactive map from the home screen of KSU Go | | | | |
| **Relationships:**  **Extend:** Log in Professor/Student/Guest | | | | |
| **Typical flow of events:**   1. Student/Professor/Guest select the Interactive Map Icon from the home screen. 2. The user selects which campus he/she would like to go to. 3. Map then displays the map of selected campus. 4. Users can use search function to find campus buildings 5. Users can also pin locations that they find important 6. Users can filter the map pins by favorites, buildings, parking lots, etc | | | | |
| **Assumptions**  User wants to look at map of KSU  The app is connected to the internet | | | | |
| **Implementation Constraints and Specifications:**  The Marietta and Kennesaw Campus Map will be integrated by calling the backend where the database contains map layouts for both campuses. There will be easy search functionally that categorizes the campus by Dinings, Academic Buildings, Housings, Other. There is also going to be drop pins functionality when accessing campus map layout. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Access All Events/ Announcements Notification | | **ID:** A012 | | **Priority:** Medium |
| **Primary actor:**  Student, Professor, Guest | **Source:**  User selecting the Events feature from within KSU Go. | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  Students, faculty, or guests will be able to see any planned KSU events in a list format, with events being displayed by the day they are going to be happening on. Multi-day events will have multiple listings. | | | | |
| **Precondition:**   * User has to be logged in as either a student, professor, or guest. * App is connected to the internet * User has android phone   **Trigger:**  User selects Events from the Homescreen of KSU Go. | | | | |
| **Relationships:**  **Extend:** Log In Faculty/Student/Guest  **Depends on:**  Login use case, Main Menu | | | | |
| **Typical flow of events:**   1. Student/Professor/Guest select the Events Icon from the home screen. 2. The user selects how they wish to view events; all events, events on a specific day, or events matching a search description. 3. The app then displays the selected information in a list format to the user.. | | | | |
| **Assumptions**:  The app is connected to the internet, and the table containing events is up to date in the database. | | | | |
| **Implementation Constraints and Specifications:**  All events will be implemented by calling the backend database where ongoing events/ announcements  that are hosted based on the emails sent out from Departments or KSU. This will require staffs to input  data manually to the database on top of sending the campus or department wide emails. | | | | |

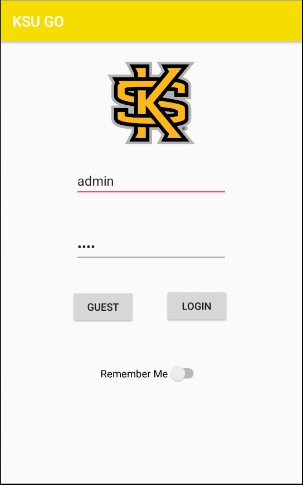
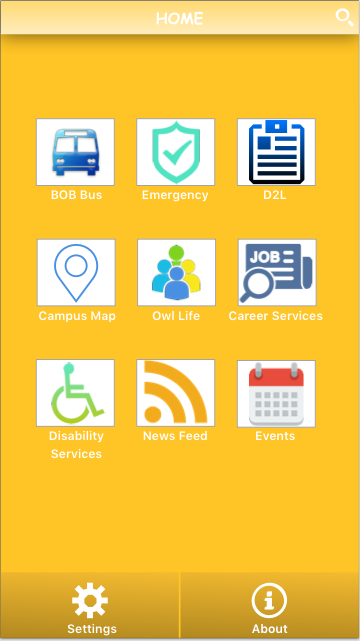
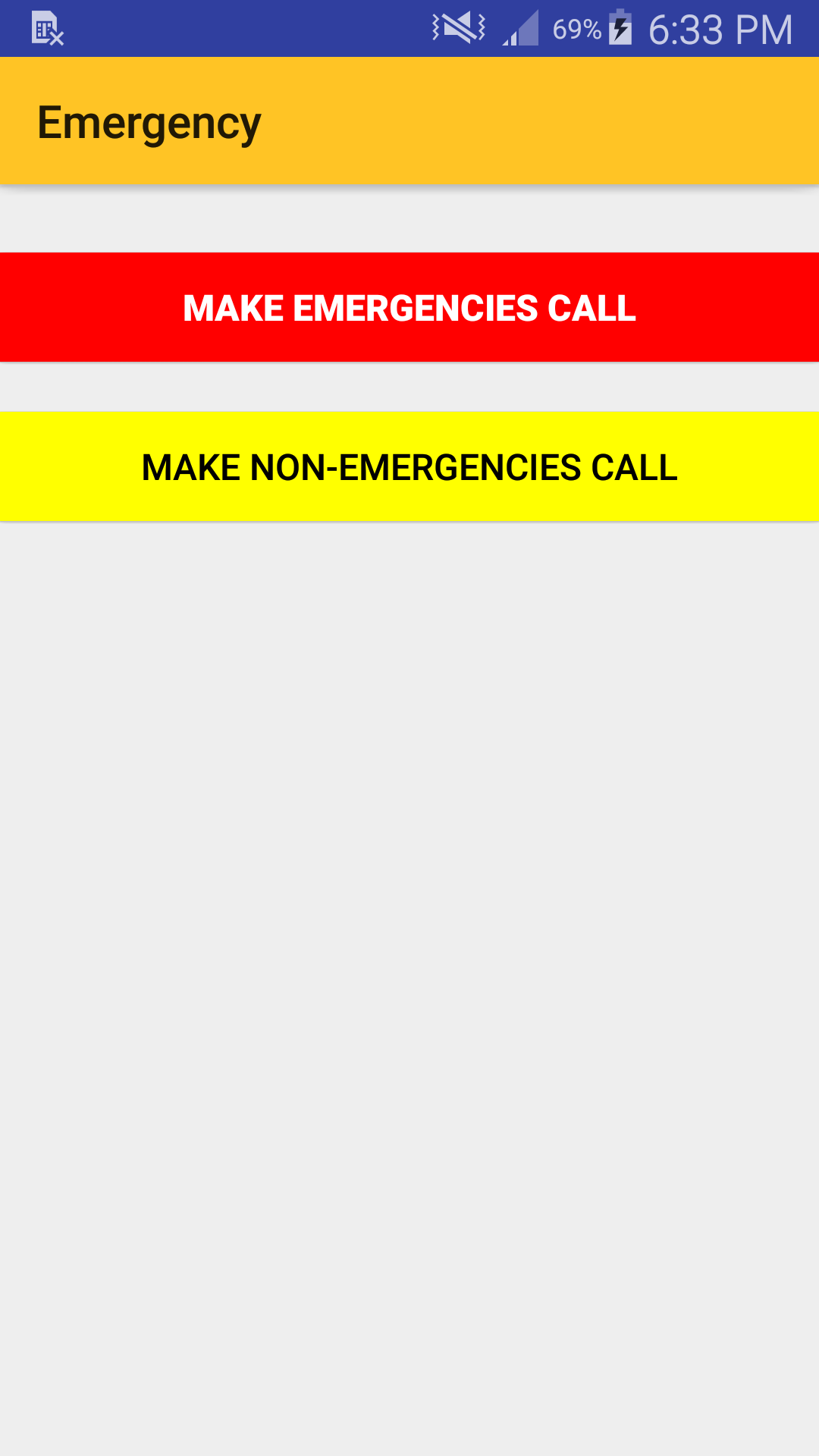
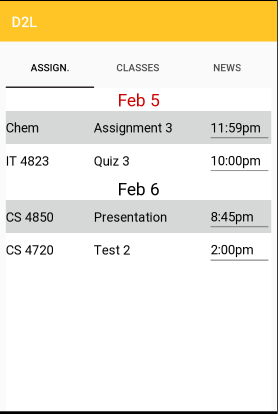
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** D2L Backend | | **ID:** A013 | | **Priority:** Medium |
| **Primary actor:**  Database | **Source:** User selecting D2L app | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  The app will connect with the database to pull in all necessary information for D2L. The app will determine whether or not the user is a professor or student. Then, display the information required for either. | | | | |
| **Precondition:**   * User has to be logged in as either a student, professor, or guest. * App is connected to the internet * User has android phone   **Trigger:**  Selecting the D2L from the home screen of KSU Go | | | | |
| **Relationships:**  **Extend:** Log in Professor/Student/Guest, Access D2L | | | | |
| **Typical flow of event:**  When the student/professor selects the D2L feature on the home screen, the application will connect to the database via a RESTful API. The database will determine if the ID of the user is a professor or student in order to set the permissions for using D2L. After determining the permissions, the app will pull in data from the database matching the student/professor’s courses that he or she is attending. The app will display the information in each of the sections within D2L. When a professor sends an announcement, the database will distribute the announcement to any students taking the course. | | | | |
| **Assumptions**  User wants to look at course information  User is attending classes at KSU  The app is connected to the internet  The database is online. | | | | |
| **Implementation Constraints and Specifications:**  N/A | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Events Backend | | **ID:** A014 | | **Priority:** Medium |
| **Primary actor:**  Database | **Source:** User selecting Events feature in KSU Go. | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  The app will connect with the database and pull in event information based upon what the user is searching for. It can pull up events based on date or the event name, which will then be displayed to the user in the app. | | | | |
| **Precondition:**   * User has to be logged in as either a student, professor, or guest. * App is connected to the internet * User has android phone   **Trigger:**  Selecting the Events feature from the home screen of KSU Go | | | | |
| **Relationships:**  **Extend:** Log in Professor/Student/Guest, Access All Events/Announcements | | | | |
| **Typical flow of event:**  When the user selects the Event feature, the application will prompt the user to either show all events or to filter events based on date or event name. Based on the user response, the app will interact with a RESTful API using a GET method, which in turn interacts with the database, and returns the requested information to the app. This information is then displayed to the user in a list format, descending in order of date. | | | | |
| **Assumptions**  User wants to look at event information.  The app is connected to the internet  The database is online. | | | | |
| **Implementation Constraints and Specifications:**  N/A | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case name:** Map Backend | | **ID:** A015 | | **Priority:** Medium |
| **Primary actor:**  Database | **Source:** User selecting the map feature in KSU Go. | | **Use case type:**  Business | |
| **Interested Stakeholders:**  N/A | | | | |
| **Brief description:**  The map feature will communicate with the database to retrieve building information and will facilitate the ability to filter by campus and by building name. | | | | |
| **Precondition:**   * User has to be logged in as either a student, professor, or guest. * App is connected to the internet * User has android phone   **Trigger:**  Selecting the map feature from the home screen of KSU Go | | | | |
| **Relationships:**  **Extend:** Log in Professor/Student/Guest, Access Basic Campus Map | | | | |
| **Typical flow of event:**  The user will be presented with a basic map of the campus of their choosing, along with the option for an interactive map. The interactive map can allow for searching by building name and filter by campus. After making a selection, the app will interact with a RESTful API to query the requested information from the database, which will return the requested information in JSON format. This information will include the building address (if it has one), its campus, and the building name. | | | | |
| **Assumptions**  User wants to look at the campus map..  The app is connected to the internet  The database is online. | | | | |
| **Implementation Constraints and Specifications:**  N/A | | | | |

## Subsystem Interfaces

Below information will contain descriptions for what the UI will contain, and some of the functionalities will have actual UI mock up as development is still ongoing.

1. Login UI
   1. 
   2. The login will require students and staffs to input their Net ID and password to access KSU GO app. The students and staffs can choose to “Remember Me” for easy access on occurences.
   3. The guests will have no login required.
2. Homepage
   1. 
   2. Students will have access to Contact Directory (through search icon), BOB Bus, Emergency, D2L (Student Mode), Owl Life (known for Clubs and Organizations - Student Mode), Handshake (known for Career Services - Student Mode), News Feed, Events (Bonus Functionality), Basic Campus Map (if possible, this will be replaced by the interactive map - bonus functionality), Disability Services (Chase’s Honors Capstone component).
   3. KSU staffs will have access to Contact Directory (through search icon), BOB Bus, Emergency, D2L (Staff Mode), Owl Life (known for Clubs and Organizations - Staff Mode), Handshake (known for Career Services - Staff Mode), News Feed, All Events/ Announcement (Bonus Functionality), Basic Campus Map (if possible, this will be replaced by the interactive map - bonus functionality), Disability Services (Chase’s Honors Capstone component).
   4. Guest will have access to Contact Directory (through search icon), BOB Bus, Emergency, News Feed, Events (Bonus Functionality), Basic Campus Map (if possible, this will be replaced by the interactive map - bonus functionality), Disability Services (Chase’s Honors Capstone component).
3. BOB Bus
   1. Students, staffs, and guests will have access to real-time bus locations based on their routes.
4. Contact Directory
   1. Students, staffs, and guests will have access to all KSU staffs information which includes Name, Associated Department, Location, and their telephone number.
5. Emergency
   1. 
   2. Students, staffs, and guests will have access to two easy dials for Emergencies and Non-Emergencies towards KSU Police.
6. D2L
   1. 
   2. Students will have access to submit their assignments, access their classes, post to the discussion board, and news notifications for emails, upcoming assignments or announcements.
   3. KSU staffs will have access to see submitted assignments, access their classes, and post and receive news notifications.
7. Owl Life
   1. Students can search, join, or quit existing clubs and organizations. They can also add their volunteer hours.
   2. KSU staffs can create, remove, search, join, or quit existing clubs and organizations. Mainly, the KSU staffs will manage the existing clubs and organizations. They can also approve submitted volunteer hours.
8. Handshake
   1. Students can search and apply for jobs, check applications, contact and set up appointments with career advisors for resume check and mock interviews.
   2. KSU staffs can add sponsors, post career events, approve and comment on appointments.
   3. Companies can post jobs, and access new applications including students’ information.
9. News Feed
   1. Students, staffs, and guests will have access to see real-time KSU Facebook posts and Twitter tweets.
10. All Events/ Announcements Notification - Bonus Feature
    1. Students, staffs, and guests will have access to see real-time on-going events/ announcements without having to look for the emails sent by KSU departments.
11. Basic Campus Map
    1. Students, staffs, and guests will have access to access both Marietta and Kennesaw Campus map.
12. Interactive Campus Map
    1. Students, staffs, and guests will have access to see the user friendly Marietta and Kennesaw Campus locations by categories, which include but not limited to Academic, Housing, Greek, Culinary, and Other. They will also have access to pin maps for directions from their current locations.

# Object Descriptions (Object Design)

## Objects in Login subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  Login | | |
| **Brief description:** This class will have the login functionality. | | if (name && pass in db)  return true;  else return false; |
| **Attributes (fields)** | **Attribute Description** |
| name (string) | Holds Username field |
| password (string) | Hold password field |
| **Methods (operations)** | **Method Description** |
| onLoginClick() (boolean) | Sends request to database to check User and pass |
| onGuestClick() (void) | Load new content to Guest Home Page |

## 

## Objects in Home subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  Home | | |
| **Brief description:** This class will have the home page functionality. | | startActivity(attribute); |
| **Attributes (fields)** | **Attribute Description** |
| D2LButton (void) | Button linked to UI |
| OwlifeButton (void) | Java attribute linked to UI |
| InteractiveMapButton (void) | Java attribute linked to the InteractiveMapButton |
| HandshakeButton (void) | Java Attribute linked to the Handshake Button |
| ContactButton (void) | Java Attribute linked to the Contact Directory Button |
| EmergencyButton (void) | Java Attribute linked to the Emergency Services Button |
| BOBButton (void) | Java Attribute linked to the BOB Tracking Button |
| **Methods (operations)** | **Method Description** |
| OnClick() (void) | Loads appropriate function depending on button clicked. |

## 

## Objects in Bus subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  Bus Route | | |
| **Brief description:** This class will have the Bus Route functionality. | | Intent browser = new Intent(Intent.ACTION\_VIEW, Uri.parse("http://kennesaw.edu/mobile/bob.php")); startActivity(browser); |
| **Attributes (fields)** | **Attribute Description** |
| KennesawMariettaRoute (route) | Button that link to bus route for Kennesaw-Marietta Route |
| BusbeeDriveRoute (route) | Button that link to bus route for Busbee Drive/ Stadium Route |
| WestCampusRoute (route) | Button that link to bus route for West Campus Route |
| FreyRoute (route) | Button that link to bus route for Frey Road Route |
| SkipSpannRoute (route) | Button that link to bus route for Skip Spann Route |
| ChastainRoute (route) | Button that link to bus route for Chastain Pointe Route |
| TownPointRoute (route) | Button that link to bus route for Town Point Route |
| **Methods (operations)** | **Method Description** |
| OnClick() (void) | Open a browser link to http://kennesaw.edu/mobile/bob.php |

## Objects in Contact Directory subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  Contact Directory | | |
| **Brief description:** This class will have the contact directory functionality. | | query = “SELECT \* FROM names WHERE professor=NameTextBox”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close(); |
| **Attributes (fields)** | **Attribute Description** |
| NameTextBox (string) | Text Box where people’s name or department’s name to search |
| OptionsDropDown (string) | Dropdown will have options for People or Department |
| SearchButton (void) | Button that link to database for searching by People or Department |
| **Methods (operations)** | **Method Description** |
| OnClickDropDown() (void) | Give options for People or Department |
| OnClickSearch() (DirectoryResult) | Search and return the findings |

## Objects in Emergency subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  Emergency | | |
| **Brief description:** This class will have the direct emergency and non emergency call functionality. | | Intent emergency = new Intent(Intent.ACTION\_DIAL, Uri.fromParts(telNumber); startActivity(emergency); |
| **Attributes (fields)** | **Attribute Description** |
| emergencyCallButton (void) | Button link to make call with KSU Emergency call. |
| nonEmergencyCallButton (void) | Button link to make call with KSU Non-Emergency call. |
| **Methods (operations)** | **Method Description** |
| OnClick() (void) | Make native call based on the telephone number linked to call button. |

## Objects in D2L subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  D2L | | |
| **Brief description:** This class will have the D2L functionality. | | AssignmentsInformation();  pull from D2L database;  return assignment;  Notification notif = new Notification(System.currentTime);  notification.setLatestEventInfo(course);  notif.notify(notification);  OnCourseDropDown();  query = “SELECT courses FROM courselist WHERE course=courseDropDown”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close();  startActivity(course);  post(discussion);  endActivity();  startActivity(course);  startActivity(grades);  startActivity(course);  startActivity(discussion);  startActivity(course);  startActivity(module); |
| **Attributes (fields)** | **Attribute Description** |
| AnnouncementsTextField (string) | Will display any announcements on D2L |
| AssignmentsTextField (string) | Will display any assignments students have due. |
| CoursesDropDown (Courses) | Dropdown menu of all course student/professor is apart of |
| ContentButton (void) | Link to activity where user can see all content for selected course |
| CourseAnnoucementButton (void) | Button that links to any announcements from specified course |
| CourseAssignmentButton (void) | Button that links to any assignments for the specified course |
| DiscussionButton (void) | Button that links to discussion board of specified course |
| GradesButton (grades) | Button that shows all grades for specified course |
| CourseModuleButton (module) | Will display any content from the specified course listed in certain module |
| AssignmentDescriptionLink (string) | TextLink that allows user to see details about selected assignments |
| AnnouncementTextBox (string) | Textbox that will give professors ability to send announcements |
| AnnouncementSend (void) | Sends announcements to either specified course or all courses |
| Discussion TextBox (string) | Here is where students will be able to type to the discussion board |
| DiscussionTextField (string) | Displays any discussions that have been started |
| Content | **Any** |
| **Methods (operations)** | **Method Description** |
| onAssignmentClick() (assignment) | Shows user more information about assignment. Also, will be where professors will be able to edit assignment dates |
| onAnnouncementSend() (void) | Sends announcements to students |
| onCourseDropDown() (course) | Selects course. This selection will be what the app uses to pull data for specified course |
| onDiscussionPost() (void) | Submits discussion that professor/student creates |
| onGradesClick() (grades) | Goes to grades of specified course |
| onDiscussionClick() (void) | Takes you to discussion board |
| onAnnouncementsClick() (void) | Takes you to announcements for specified course |
| onModuleClick() (void) | shows specified module |

## Objects in Owl Life subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  Owl Life | | |
| **Brief description:** This class will have the clubs and organizations functionality. | | if (no account){  Intent browser = new Intent(Intent.ACTION\_VIEW, Uri.parse(owlUrl)); startActivity(browser);  }  query = “SELECT \* FROM organizations WHERE organization=searchBox”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close();  query = “INSERT org INTO organizations WHERE organization=searchBox”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close();  query = “DELETE org FROM organizations WHERE organization=searchBox”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close(); |
| **Attributes (fields)** | **Attribute Description** |
| owllifeNotificationBox (string) | Dialog field to hold important information regarding Owl Life. |
| organizationBox (string) | Dialog field to hold information about which organizations the user is enrolled in. |
| searchBox (void) | Dialog field to permit searching of all KSU organizations. |
| **Methods (operations)** | **Method Description** |
| onRedirectSignIn() (void) | Redirects the user to the Owl Life website in the case that they do not currently have an account |
| searchOrgs() (string) | Searches the database of student organizations and populates the searchBox. |
| addOrg() (void) | Updates the database to add an organization to the user’s account. |
| removeOrg() (void) | Updates the database to remove an organization from the user’s account. |

## Objects in Handshake subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  Handshake | | |
| **Brief description:** This class will have the functionality for career services. | | if (no account){  Intent browser = new Intent(Intent.ACTION\_VIEW, Uri.parse(handshakeUrl)); startActivity(browser);  }  fetch event from handshake;  startActivity(map);  place event on map;  startActivity(announcements);  update page with eventList; |
| **Attributes (fields)** | **Attribute Description** |
| stuID (int) | Holds the value of the student’s identification to synchronize with Handshake website & account |
| handshakeNotificationBox (string) | Dialog field to hold important information referring to Handshake |
| eventList (string) | Table of text data representing information on events occurring at KSU via Handshake |
| eventListTemp (string) | Table of text to hold updated information whenever the application receives new information |
| **Methods (operations)** | **Method Description** |
| onRedirectSignIn() (void) | Redirects the user to the Handshake website in the case that they do not currently have an account |
| onEventLocate() (event) | Takes data from user selecting location of event through Handshake and opens the interactive map at the location with a pin. |
| onAnnouncementUpdate() (void) | Updates the main page announcements with the information from Handshake chosen by user |

## Objects in NewsFeed subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  NewsFeed | | |
| **Brief description:** This class will have the real time update and views on KSU posts from Facebook and tweets from Twitter. | | TwitterFactory twitter = new TwitterFactory(config);  Twitter twitt = tf.getInstance();  stat = twitt.getUserTimeline(ksuTwitter);  return status.getText(); |
| **Attributes (fields)** | **Attribute Description** |
| ShowMoreButton (void) | Button to load more posts and tweets |
| RefreshButton (void) | Button to refresh Facebook posts and Twitter tweets |
| **Methods (operations)** | **Method Description** |
| onClickNews() (string) | Load more posts and tweets |
| autoUpdateNews() (void) | Refresh to update based on Facebook posts and Twitter tweets |

## Objects in Basic Campus Map subsystem

|  |  |  |
| --- | --- | --- |
| **Class name:**  Basic Campus Map | | |
| **Brief description:** This class will have the campus map functionality. | | query = “SELECT campuslayout FROM campus WHERE campustype=layout”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close(); |
| **Attributes (fields)** | **Attribute Description** |
| MCViewButton (void) | Button link to open Marietta Map layout |
| KCViewButton (void) | Button link to open Kennesaw Campus Map layout |
| **Methods (operations)** | **Method Description** |
| onClickMC() (layout) | Access database for Marietta Map layout |
| onClickKC() (layout) | Access database for Kennesaw Map layout |

## Objects in Interactive Campus Map subsystem

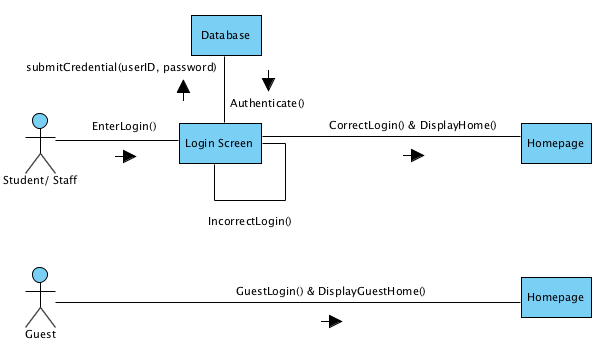
|  |  |  |
| --- | --- | --- |
| **Class name:**  Interactive Campus Map | | |
| **Brief description:** This class will have the campus map functionality, where students can get directions to the places at KSU. | | startActivity(map)  if (viewActivity == marietta)  return mariettaCampus;  if (viewActivity == kennesaw);  return kennesawCampus;  onConnected(locationServices);  latitude = valueOf(getLatitude);  longitude = valueOf(getLongitude);  return position;  query = “SELECT campuslayout FROM campus WHERE campusdescription=searchBar”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close();  updateMap(location);  getPosition(locationServices);  latitude = valueOf(getLatitude);  longitude = valueOf(getLongitude);  return position;  updateMap(position);  navigatePath(location, location);  Maps API:  startPos(location);  endPos(location);  RunRouteCalc(startPos, endPos); |
| **Attributes (fields)** | **Attribute Description** |
| mariettaViewButton (void) | Java Link to UI for Marietta View. |
| kennesawViewButton (void) | Java Link to UI for Kennesaw View |
| positionButton (void) | Java Link to UI to move Map to current Position. |
| searchBar (void) | Java link to UI to search map |
| navigateButton (void) | Java Link to UI for NavigateButton |
| **Methods (operations)** | **Method Description** |
| OnClickMariettaButton() (mariettaCampus) | Action Performed when mariettaViewButton is clicked. Shows Marietta Campus |
| OnClickKennesawButton() (kennesawCampus) | Action Performed when kennesawViewButton is clicked. Shows Kennesaw Campus. |
| OnClickShowCurrentPosition() (position) | When positionButton is clicked the map will move itself or update itself to show where you currently are. |
| OnEventSearch(description) | Does a search of the database looking for places fitting to description or name of place on either campus. Triggers UpdateMap(location) to move map to hit on database if successful. |
| UpdateMap(location) | Retrieves location needed to move too in parameters and updates map to reflect that. |
| navigatePath(location, location) | First parameter is current location and second is end results. Works with Google Maps API to find route from current location to end location |

## Objects in All Events subsystem

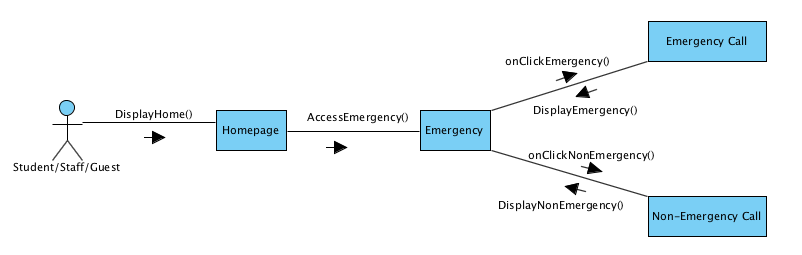
|  |  |  |
| --- | --- | --- |
| **Class name:**  All Events | | |
| **Brief description:** This class will have the events notification functionality. | | query = “SELECT event FROM events WHERE eventdescription=description”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close();  updateList();  if (searchBar is empty and match not found)  return “No Matching Events Found”;  else return returnData;  query = “SELECT event FROM events WHERE eventdescription=description AND days=day”;  Cursor cursor = db.rawQuery(query);  cursor.moveToFirst();  cursor.close();  updateList(); |
| **Attributes (fields)** | **Attribute Description** |
| searchBar (void) | Java link to UI to search events, or search days. |
| daySelect (day) | Java link to UI to select the days to display events. |
| eventList (event) | Scrollable list that will display the results of the event search. Will display all events if no search terms are submitted. |
| **Methods (operations)** | **Method Description** |
| OnEventSearch(description) | Searches the database for any event name that matches the provided description. Calls UpdateList() after successful query. |
| UpdateList(returnedData) | Populates the eventList with the returned information from the database. If search terms are provided and no match is found, displays “No Matching Events Found”. If no search terms are provided, displays all events currently in the database. |
| SelectDay(day) | Adds additional parameter to OnEventSearch, if used. Only days selected will be queried in the search. |

# Object Collaboration (Process View)

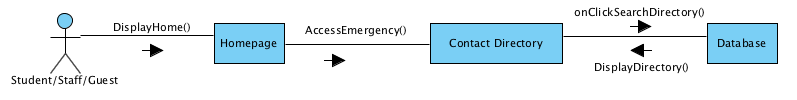
## Objects in Login subsystem



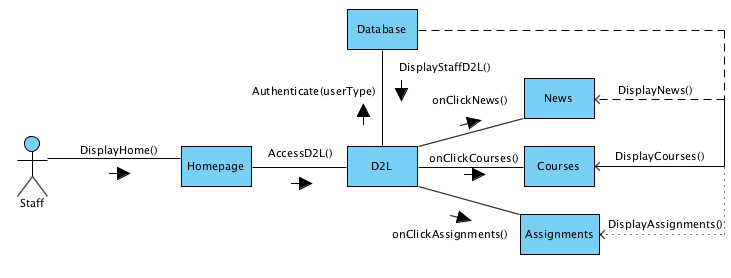
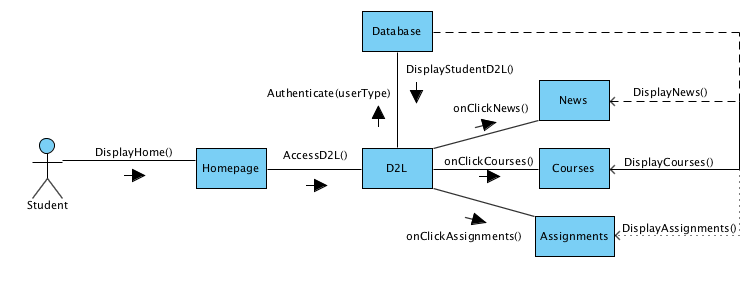
## Objects in Emergency subsystem



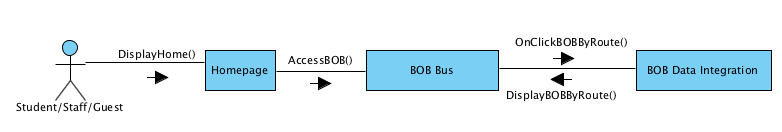
## Objects in Contact Directory subsystem



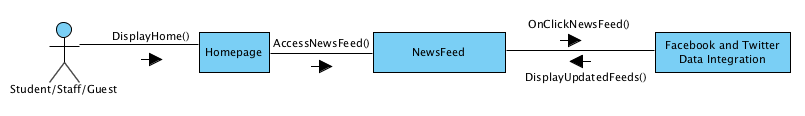
## Objects in D2L subsystem



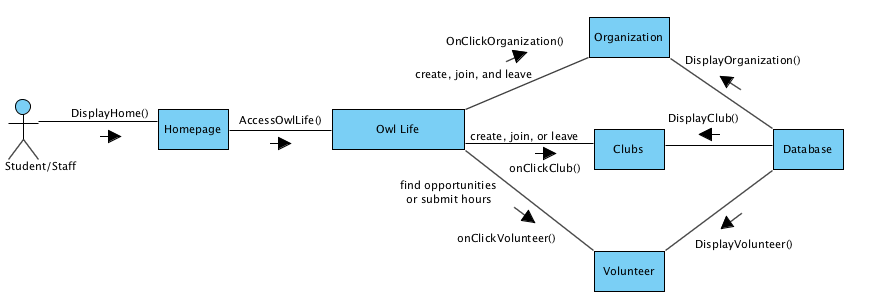
## Objects in BOB Bus subsystem



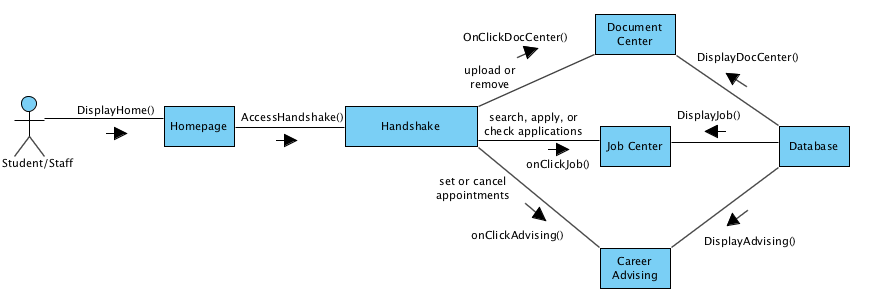
## Objects in News Feed subsystem



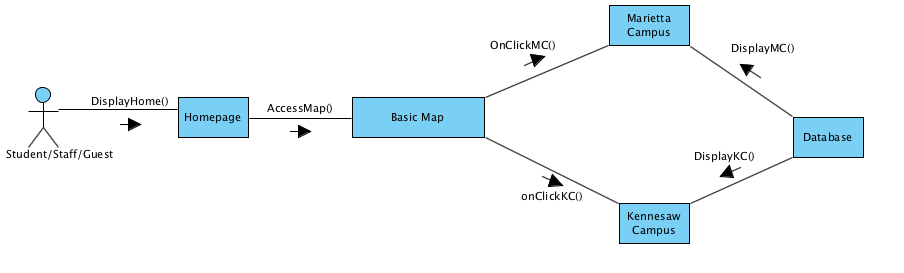
## Objects in Owl Life subsystem



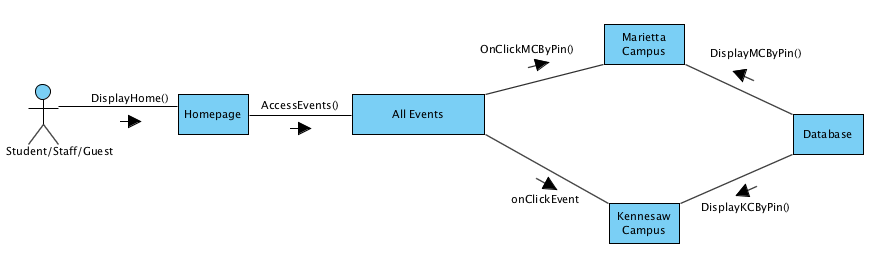
## Objects in Handshake subsystem



## Objects in Basic Map subsystem



## Objects in Interactive Map subsystem

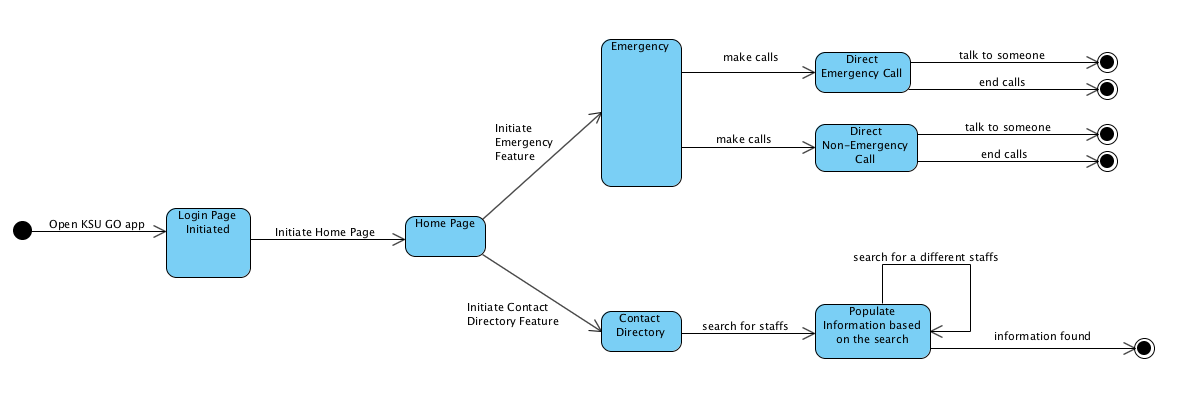


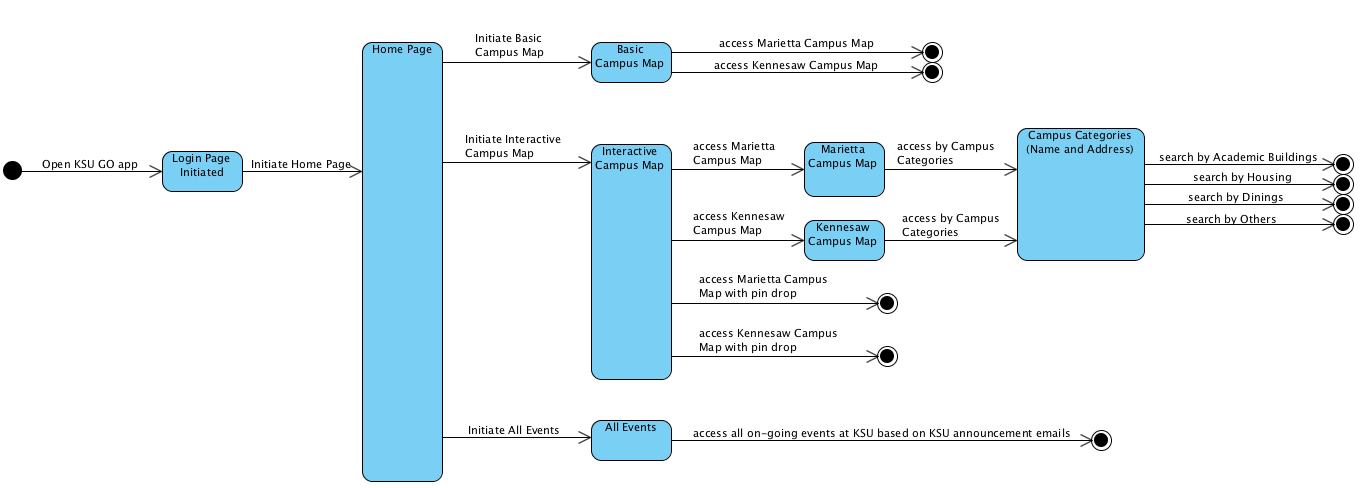
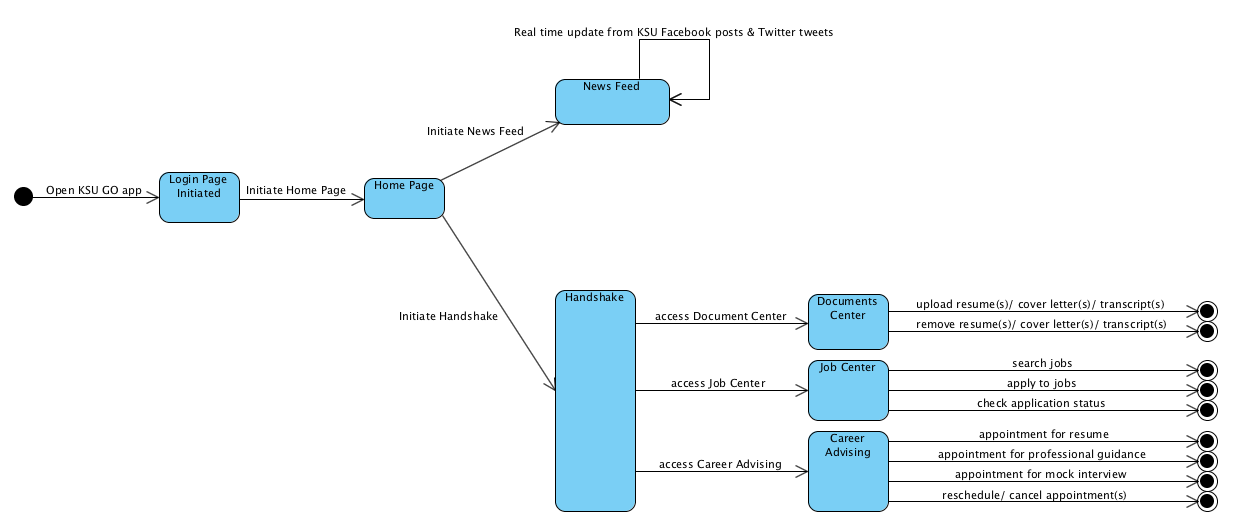
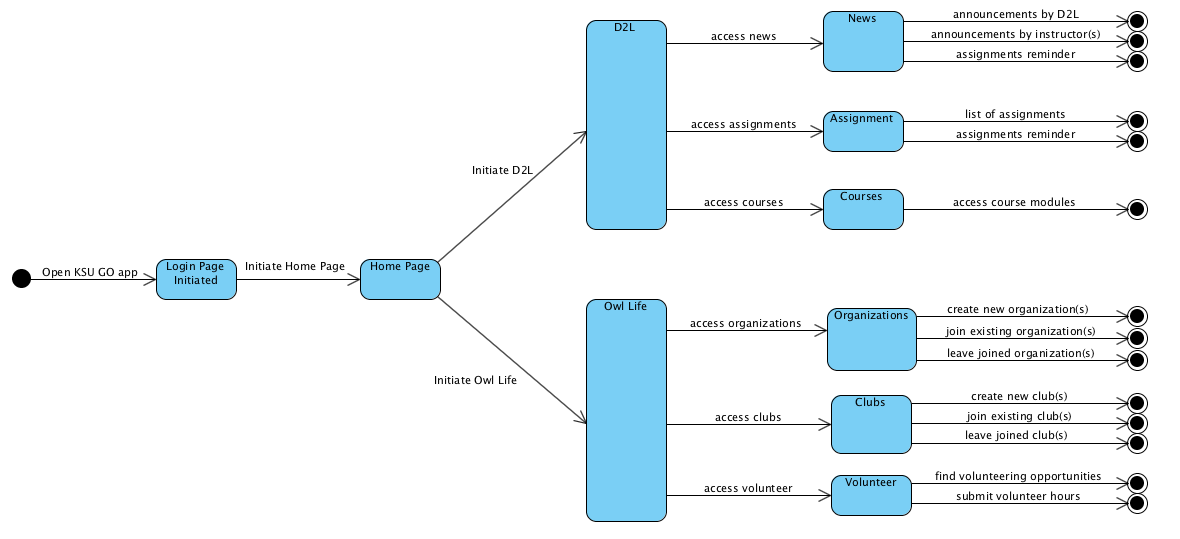
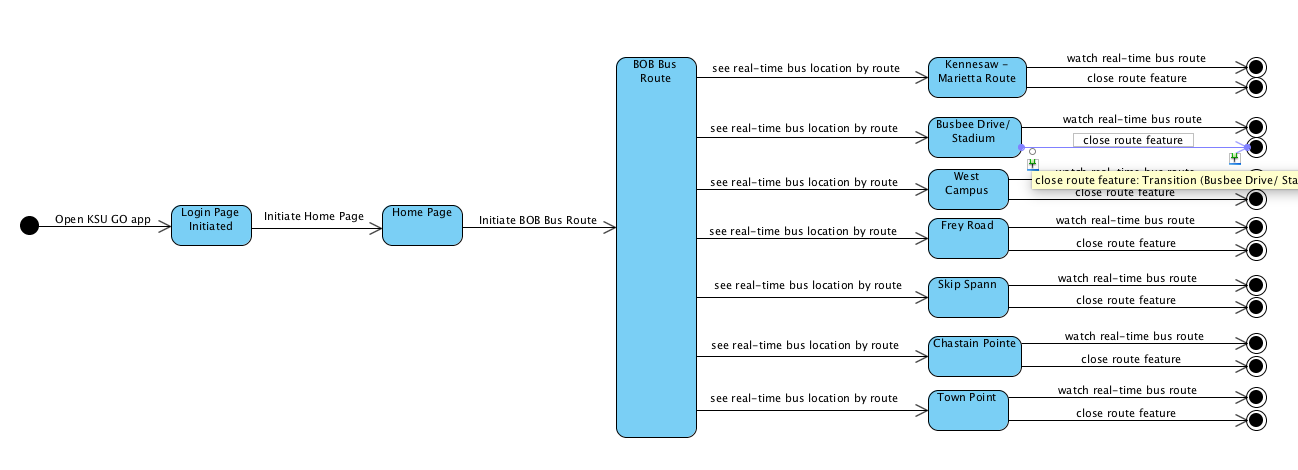
## Objects in All Events subsystem



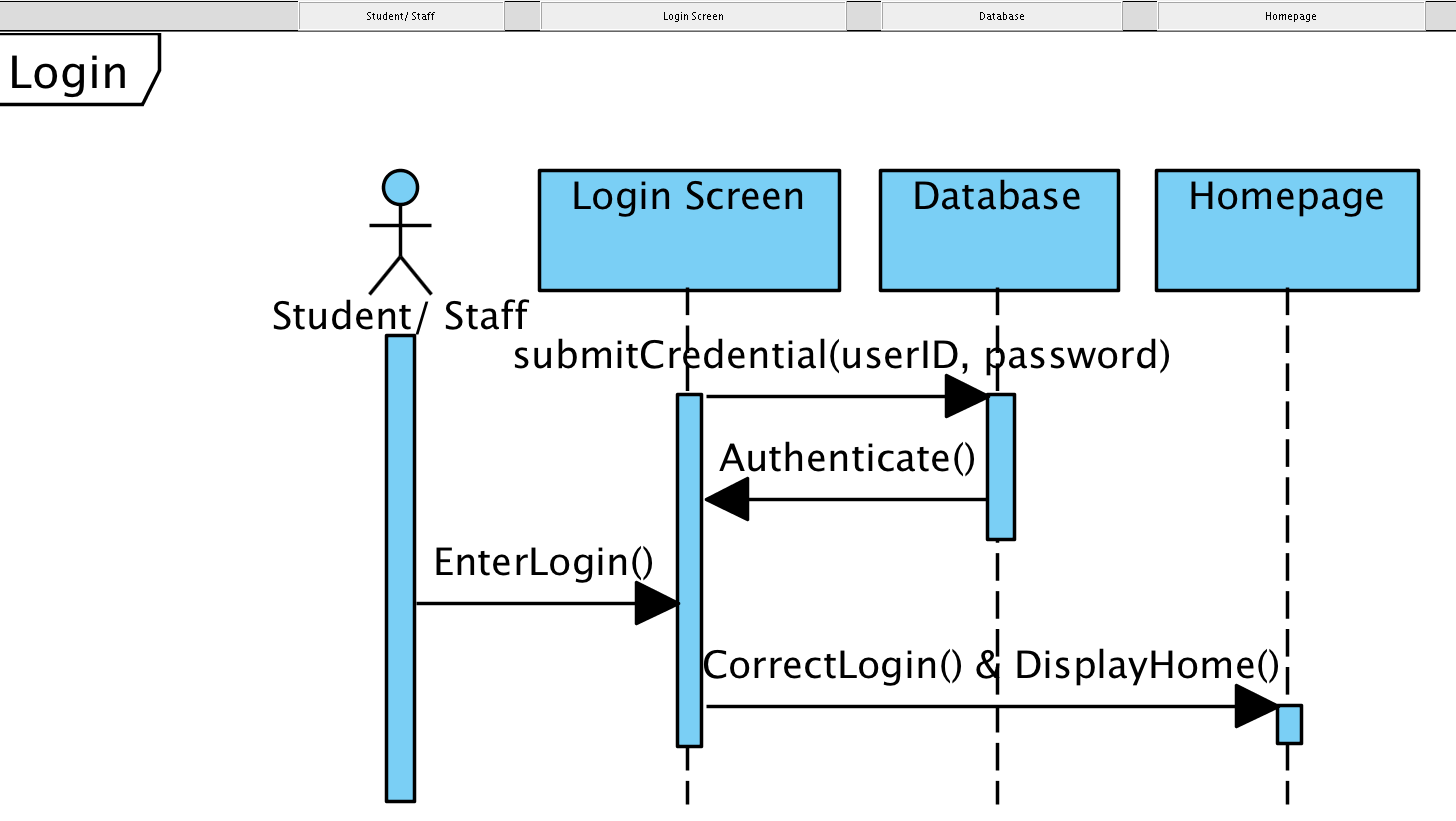
# Dynamic Model

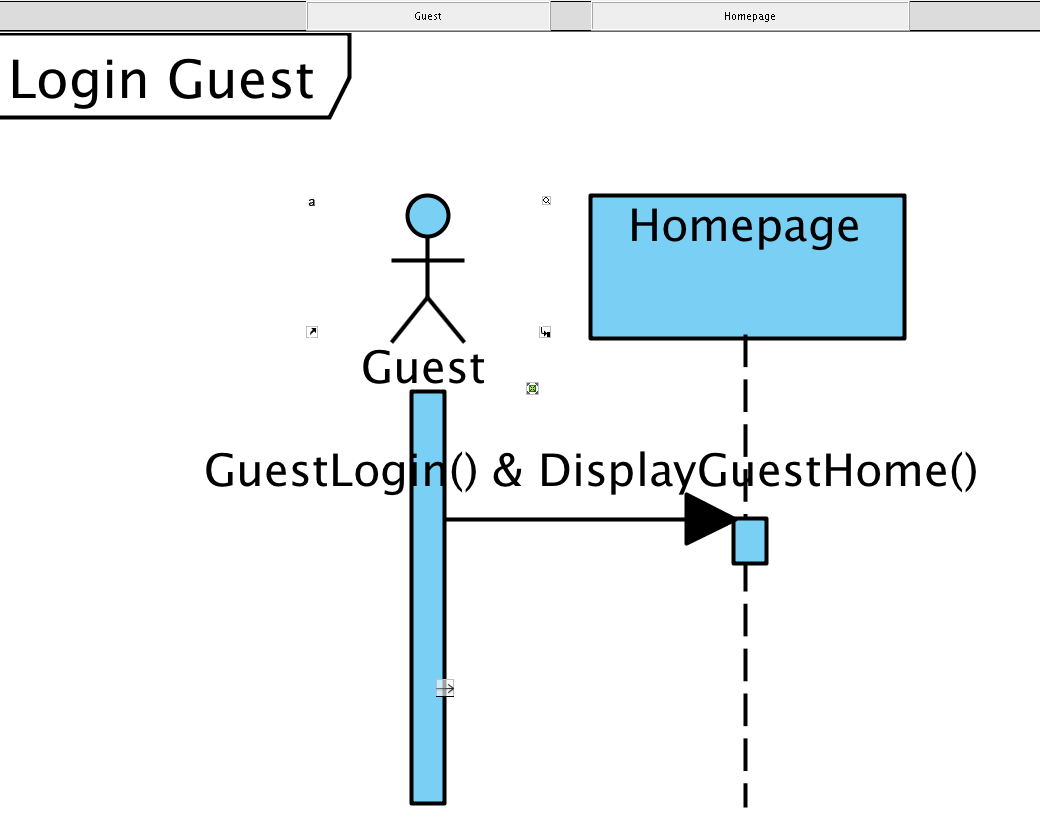
## State Diagrams

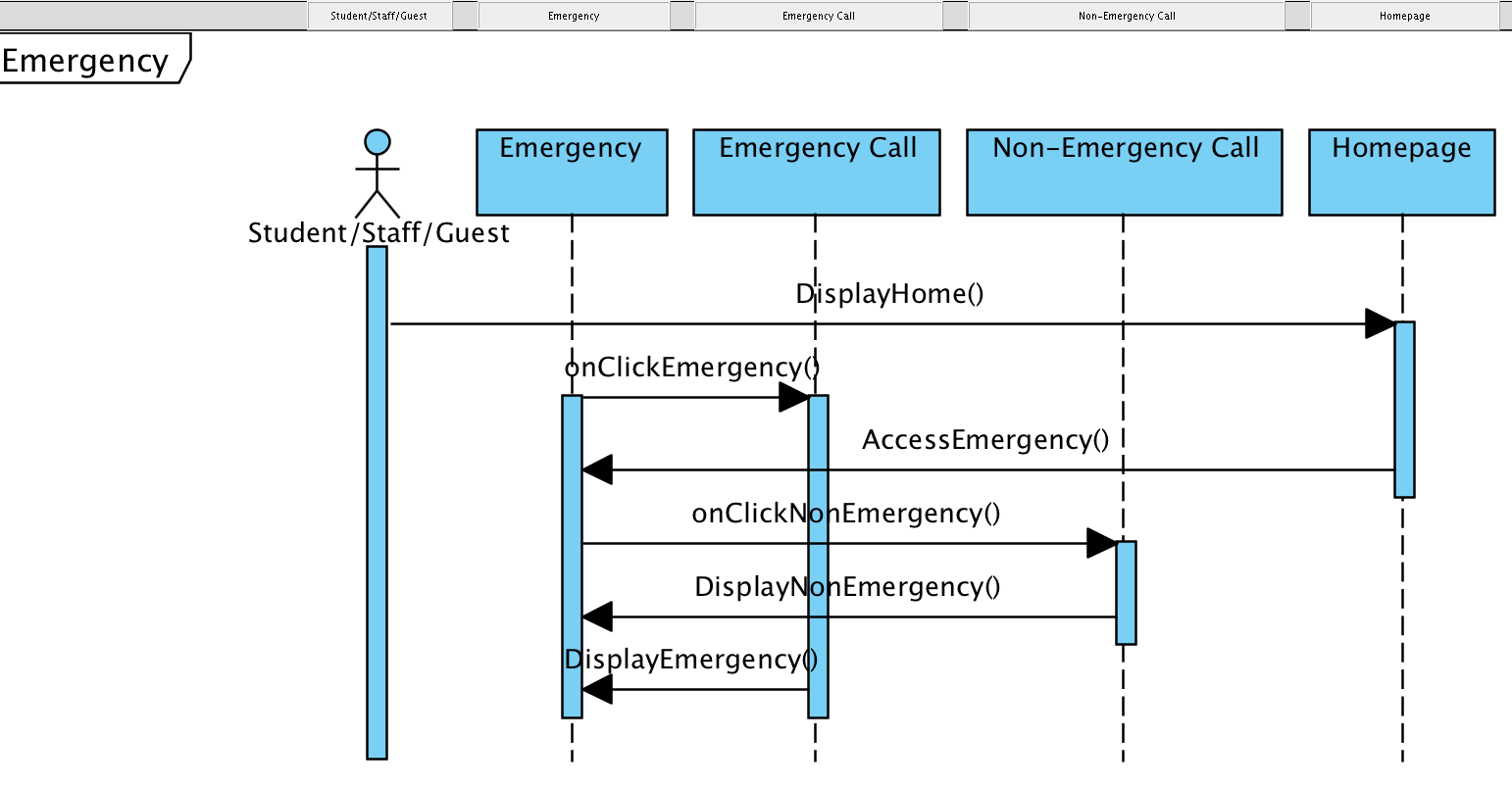


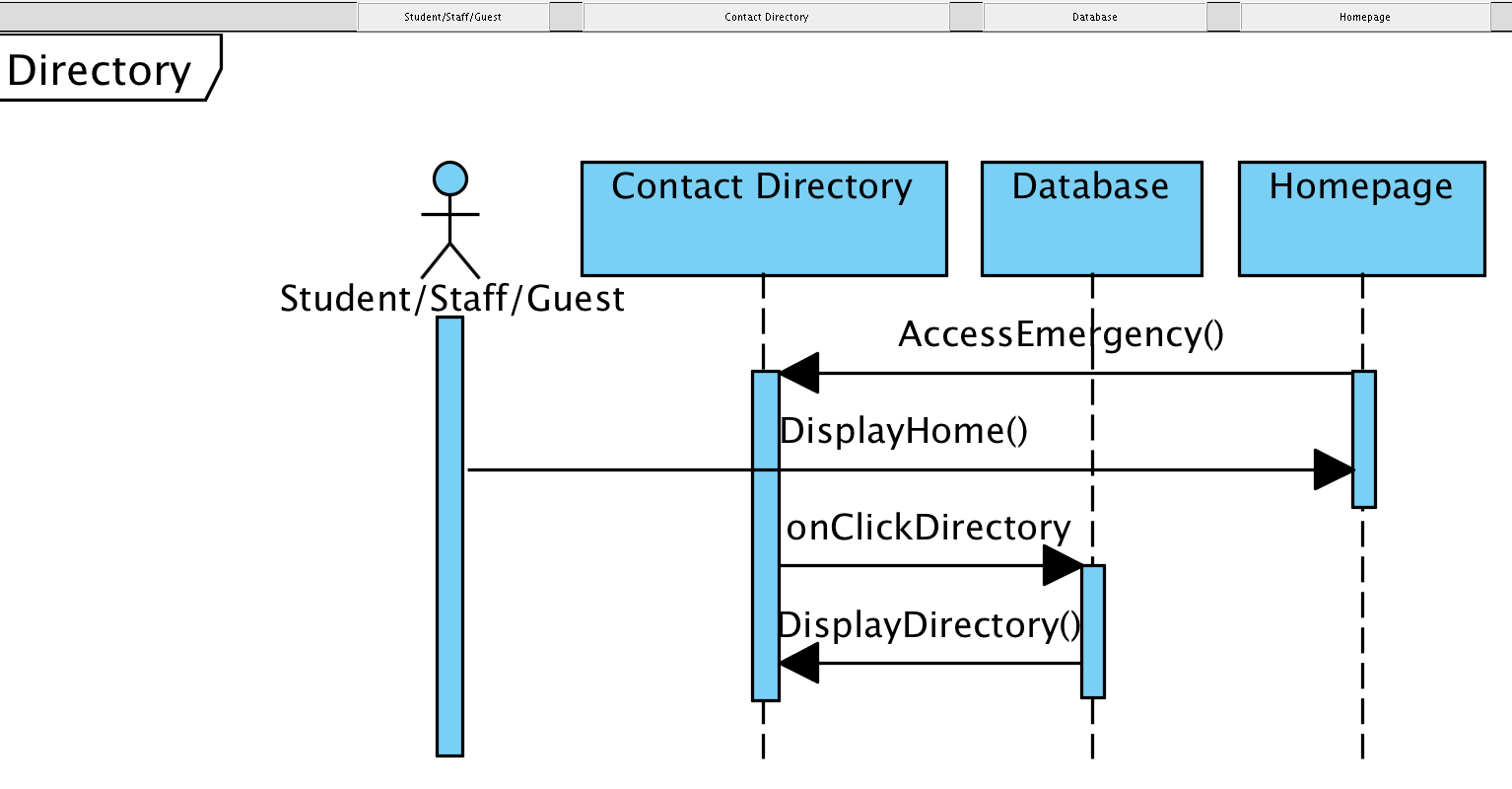


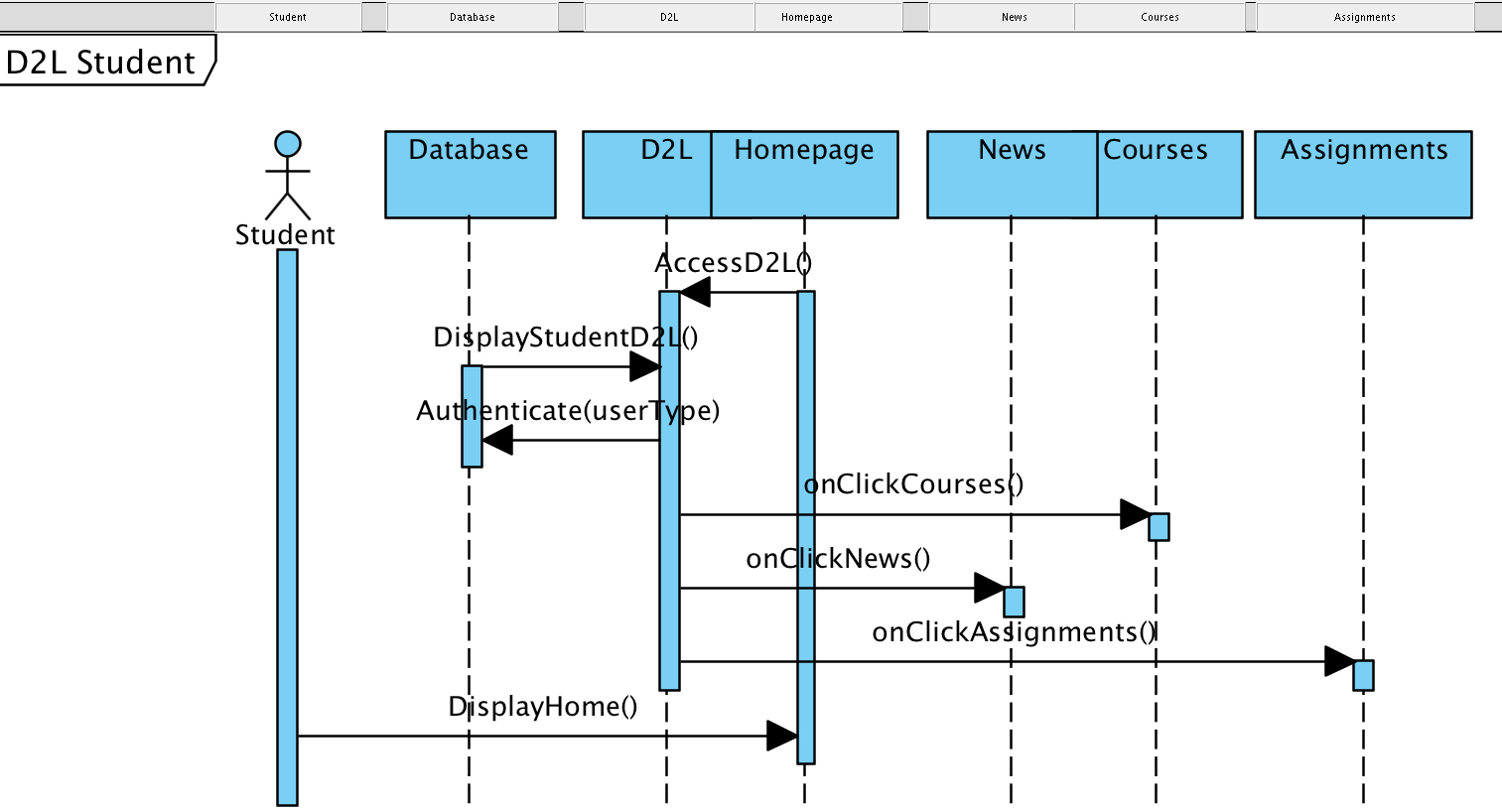
## Sequence Diagrams

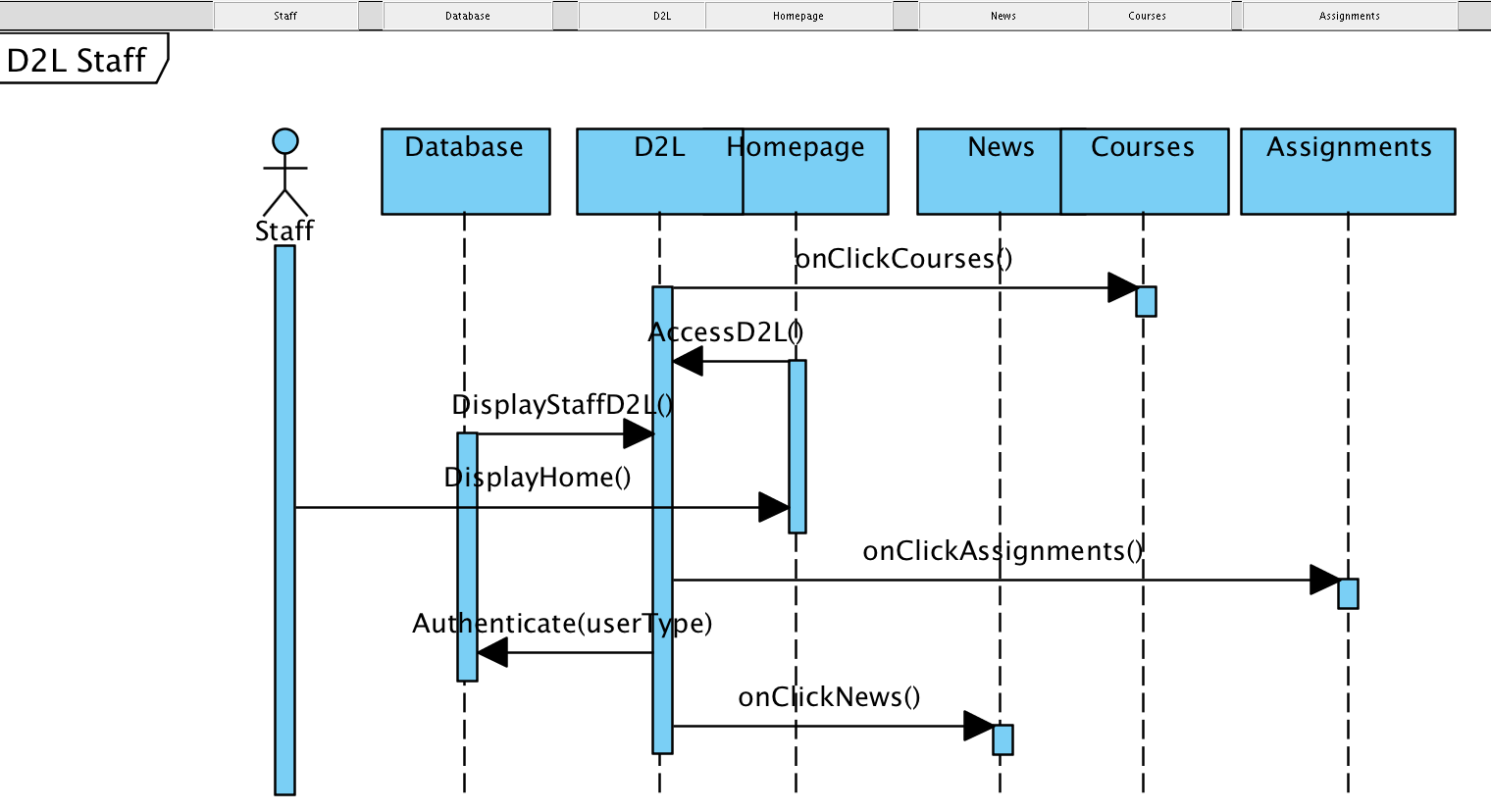


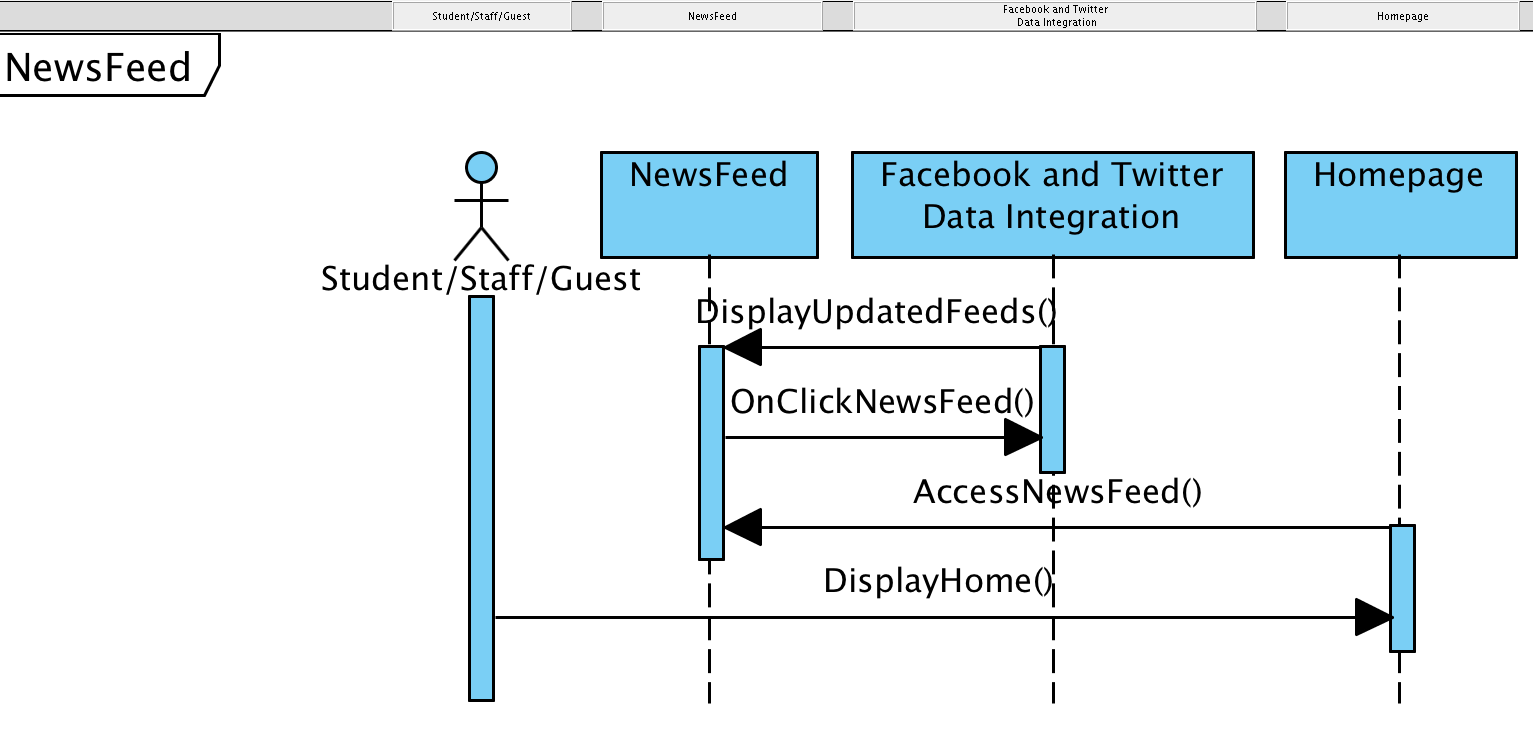
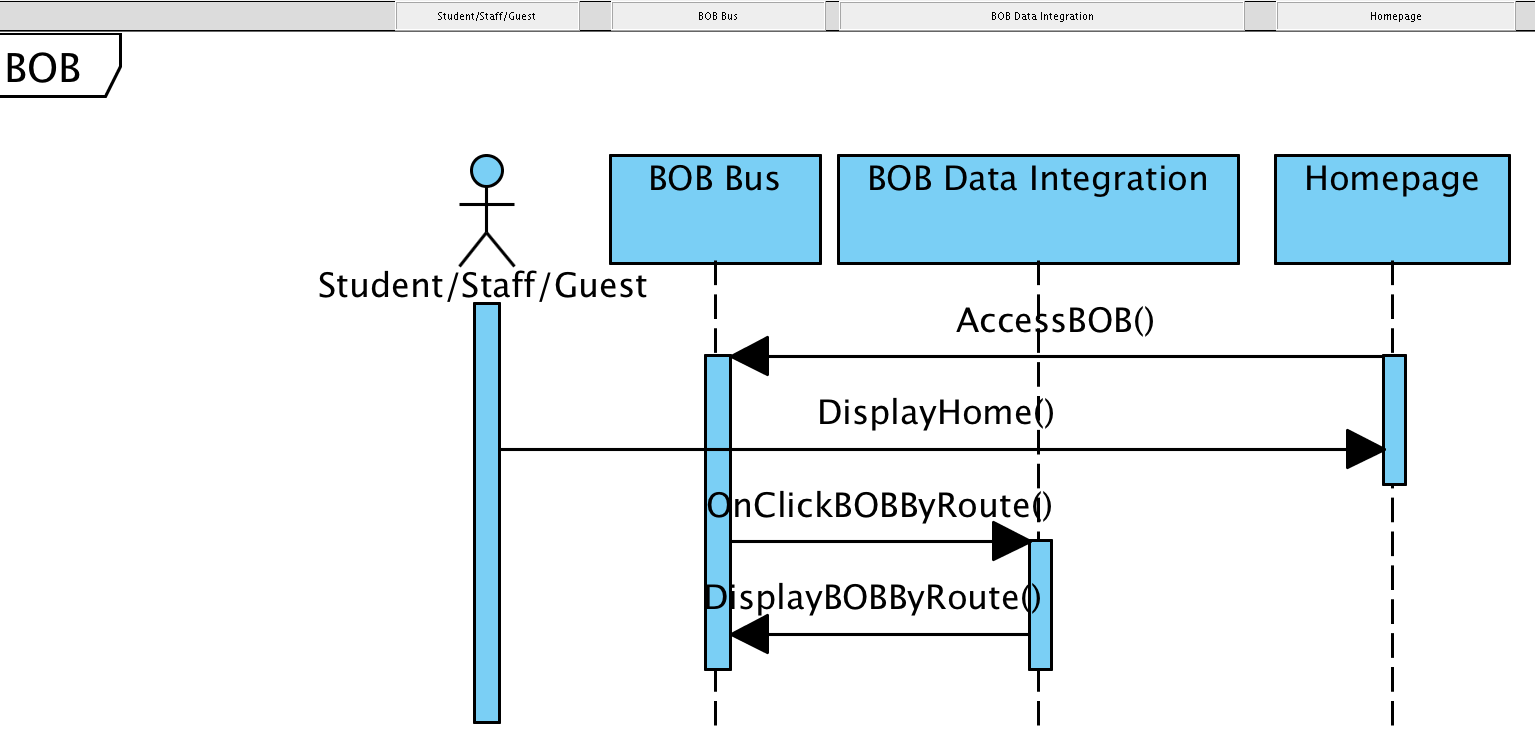


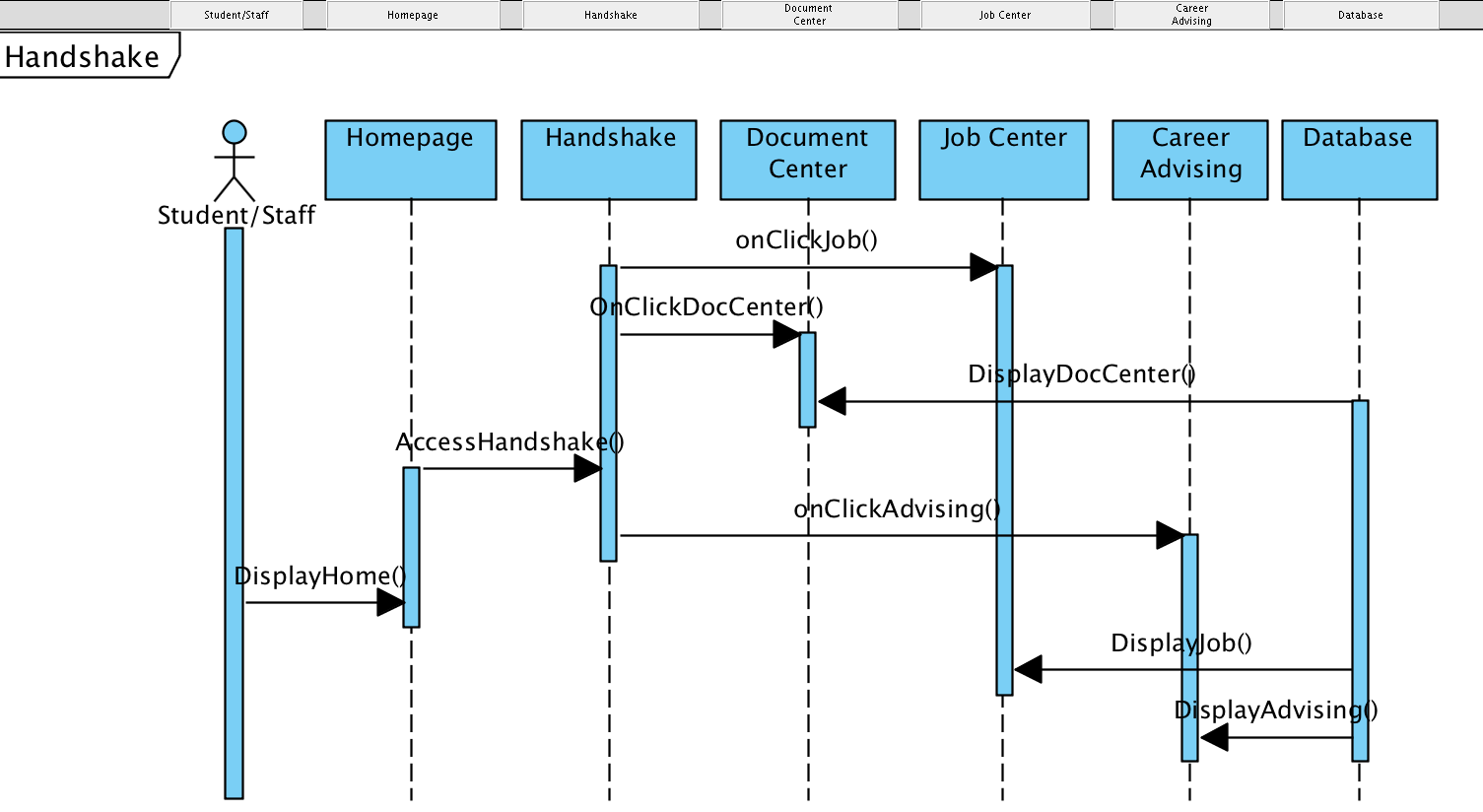
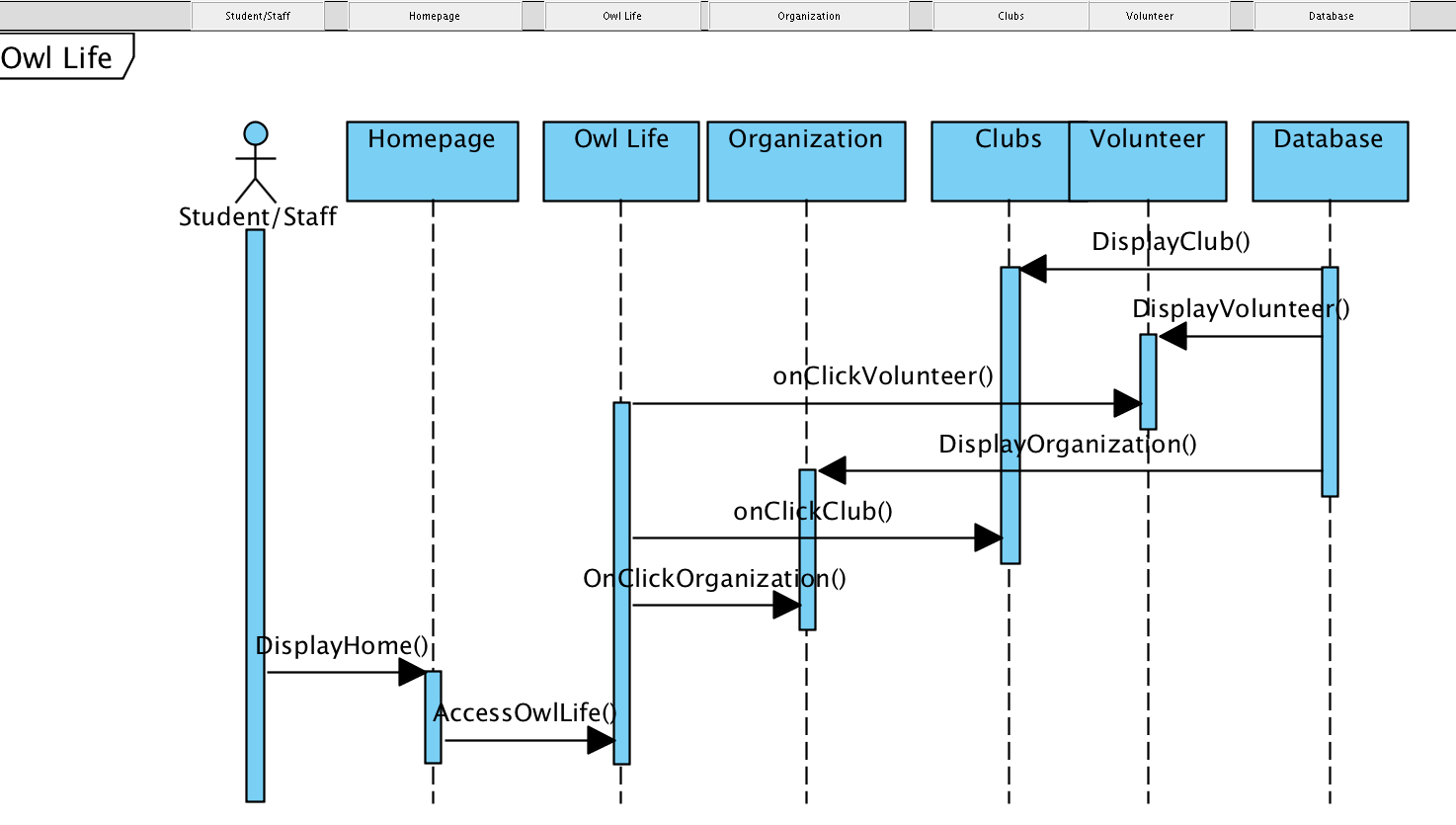


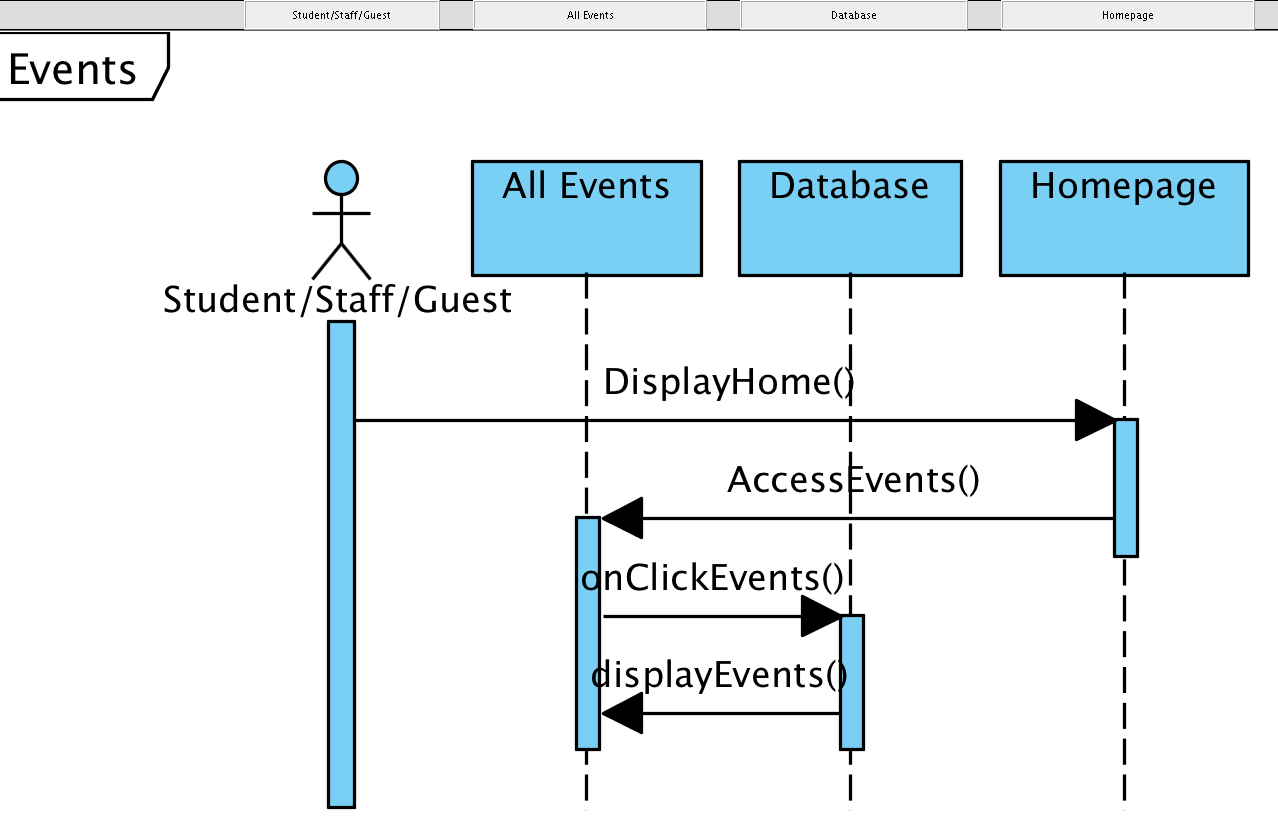
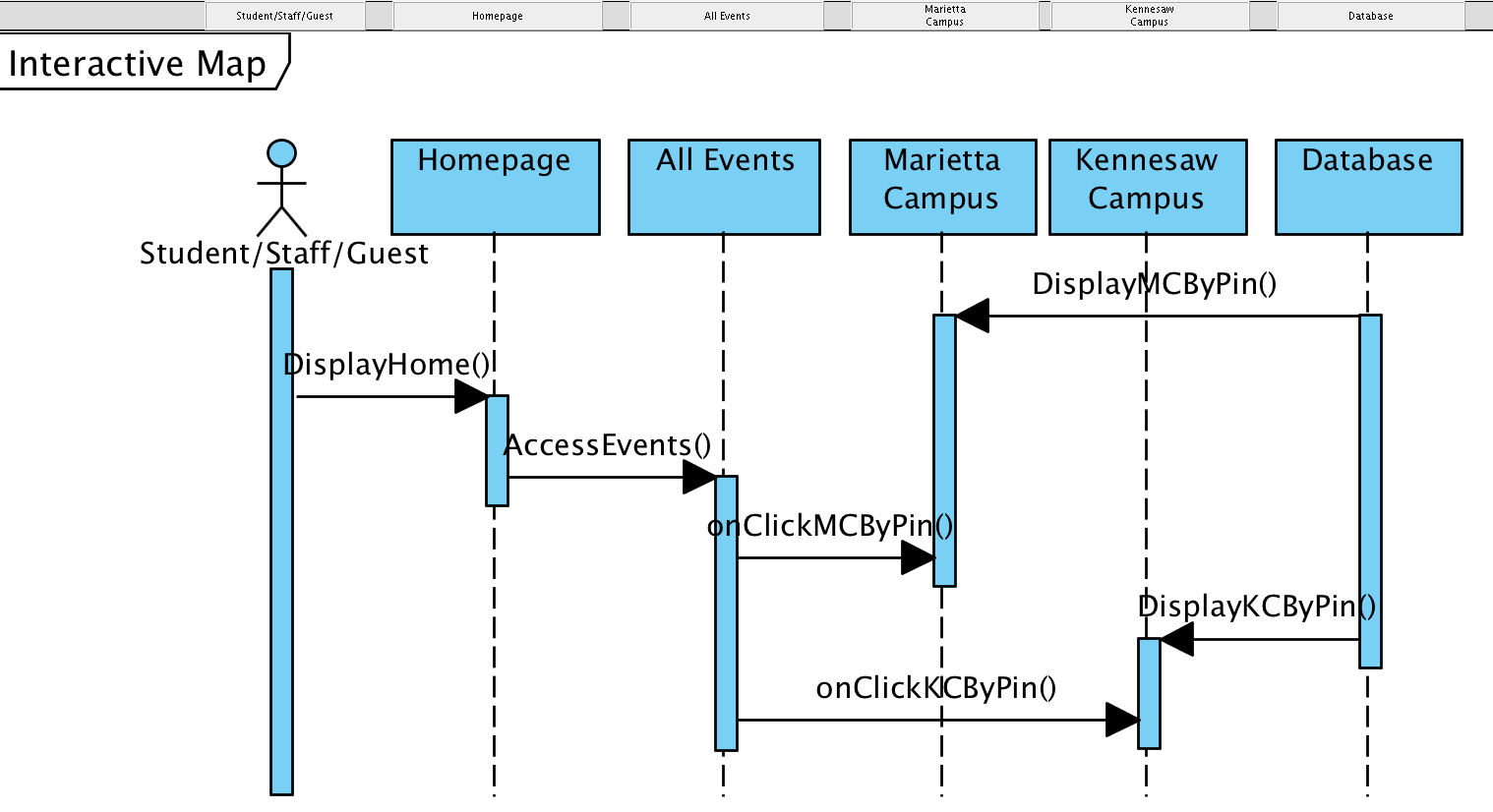
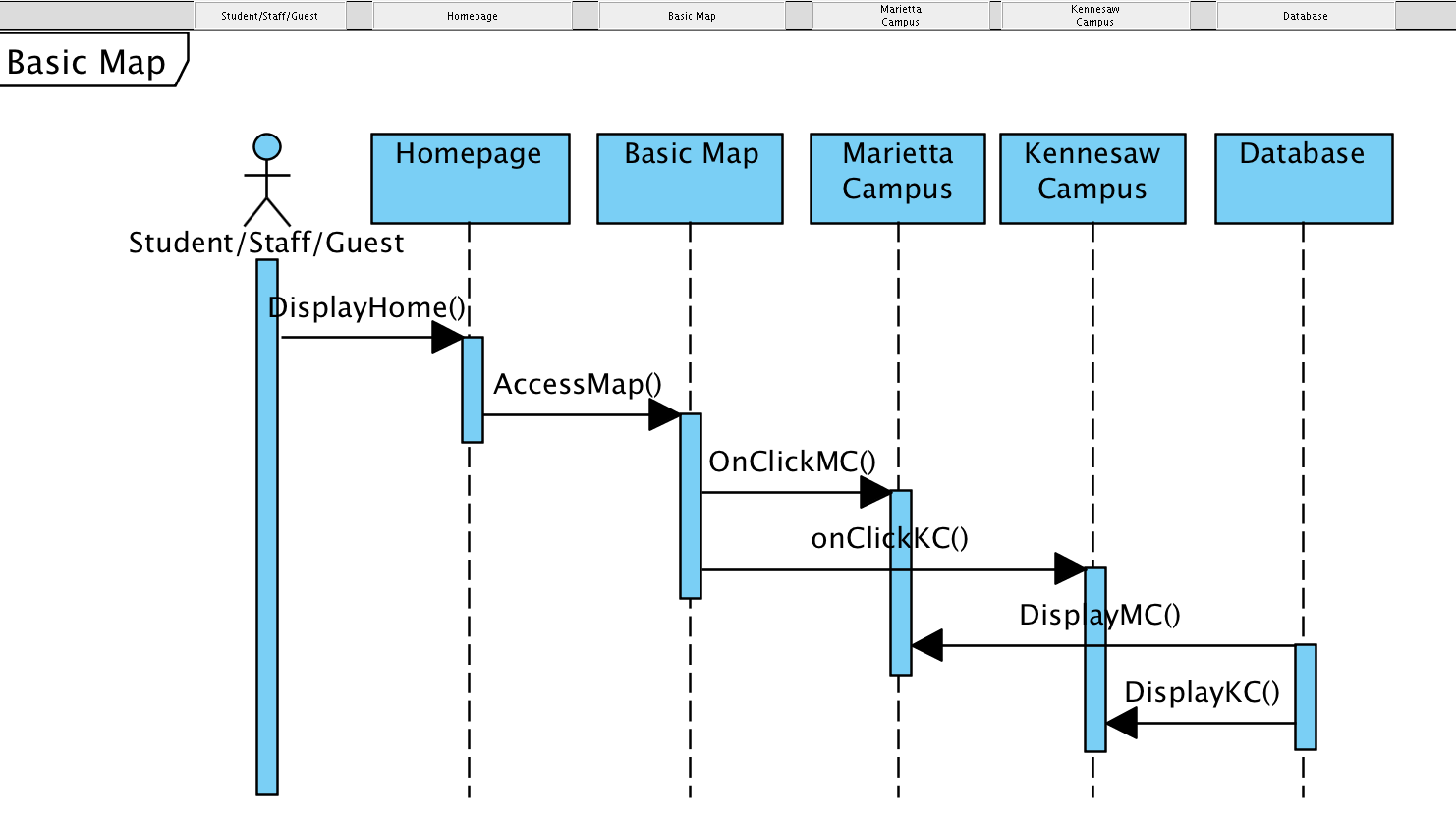






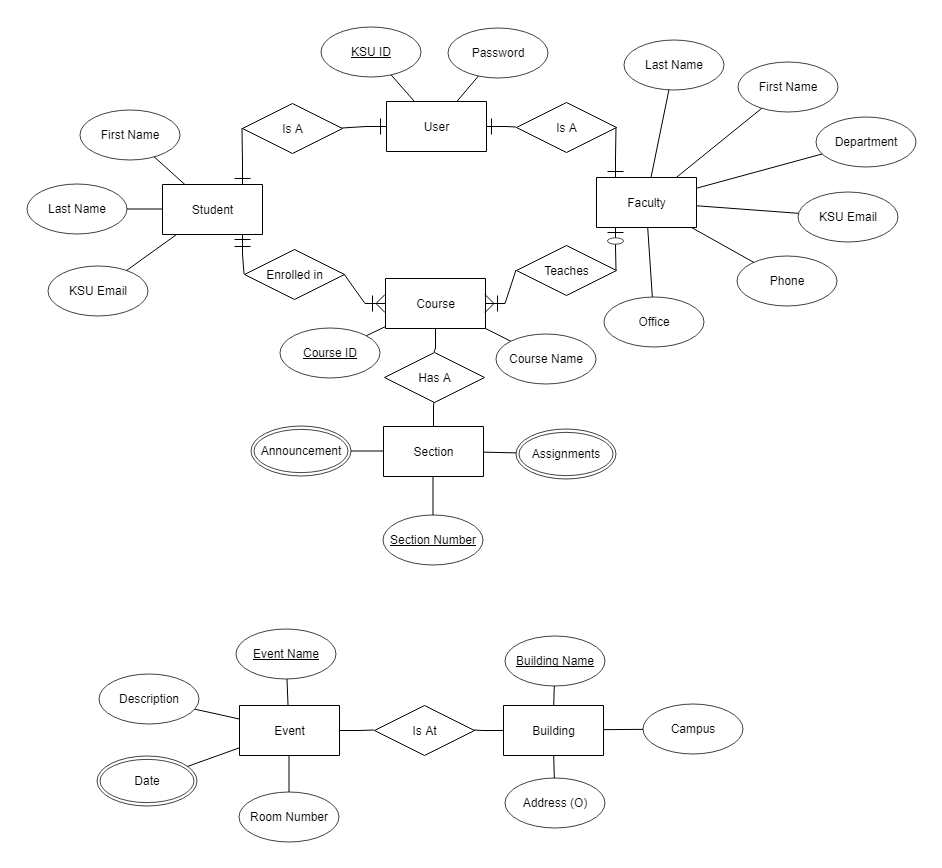




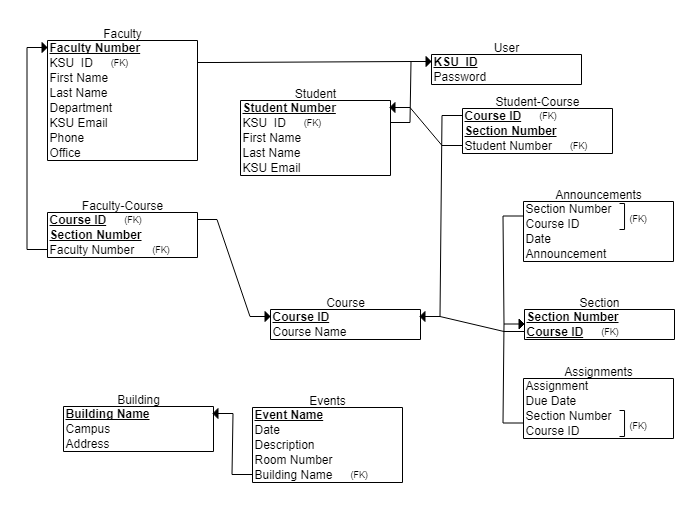


# Data Design

## Entity Relationship Diagram



## Relational Schema



As shown above, the data will be stored in a series of tables. All building data will be stored in the Building table, which will be used in both the campus map as well as the events features. The events table will hold all the data on a particular event, with the date being a multivalued attribute since it can take place over multiple days.

The other tables will hold the information for Users and Courses. The Faculty table and Student table holds only the respective information of either faculty or student, with both having their specific KSU IDs as foreign keys. The Faculty Number and Student Number primary keys are only there to serve as a unique identifier for a row and do not hold great significance. The Student-Course and Faculty-Course serve as intersection tables for Student, Faculty, and Course. This allows for querying whether or not a student belongs to a certain course, or whether or not a particular faculty teaches a course.

The Course is further specialized into sections, stored in the Section table. It’s made of a composite primary key of the Section Number and Course ID as a foreign key. This way, each section of a class is uniquely identified. The Announcements table will hold all the announcements in the system, allowing for the app to query only the announcements that pertain to the class that the user is viewing. A similar layout exists for the Assignments table; it holds all the assignments for all classes, but will only show the relevant assignments for a given section number and course ID.

All of this data is going to be stored in a MySQL database hosted by Amazon Web Services, and will communicate with the application through a RESTful API.

# Non-functional Requirements

The user must have an Android device with internet capabilities in order to fully utilize the application.

Some features simply will not load due to an internet connection requirement that we have in place.

This should not be a problem due to the fact that Kennesaw State University has a WiFi connection for students, faculty, and staff on the campus. However, the user must ensure that the WiFi connection feature of their respective device is enabled and connected to a WiFi hotspot.

User should be able to see all of their classes listed in D2L. When opening the app, all announcements should be up-to-date. All buildings will be displayed accurately on the map. The B.O.B tracker will display the location of the bus accurately within a window of 4 minutes.

# Supplementary Documentation

Supplementary documentation is not applicable towards KSU GO project.