NJIT STUDENT GUIDE FOR iOS

Documentation for Milestone 1

Team 4:
Arif Mukhthar Umarkayam Shakilabanu
Ravi Ragul Balakrishnan
Srinivasan Sundaramoorthy
Tushar Swaminathan

GIT repo:
https://github.com/arifmukhthar/NJITApp

Table of contents :

1.0 Introduction 1

2.0 Glossary 1

3.0 Requirement Specification 2

3.1 System Requirements : 3

3.2 User Requirements : 3

4.0 System Architecture : 4

5.0 Overall Sequence Diagram : 6

6.0 Network Architecture Diagram : 7

6.0 User Interaction Flow : 8

6.1 Appointment Schedule Module : 9

6.2 Know Your Professor Module: 10

6.3 Academic Calendar Module : 11

6.4 Vector Module : 12

6.5 Contacts Module : 13

6.6 Lost & Found Module : 14

6.7 ToDo Module : 15

6.8 Broadcast : 16

6.9 What's that building? 17

6.10 Shuttle : 18

6.11 Buy and Sell books: 19

6.12 NJIT MAPS 20

7.0 Data Flow Diagrams : 21

7.1 Appointment Schedule Module : 21

7.2 Academic Calendar Module : 22

7.3 Know Your Professor Module : 23

7.4 Vector Module : 24

7.5 Contacts Module : 26

7.6 Lost and Found Module : 27

7.7 Broadcast Module : 28

7.8 NJIT MAPS Module : 29

7.9 Buy and Sell books Module : 30

7.10 Shuttle Module : 31

7.11 What's that Building : 32

8.0 Test Case: 33

9.0 Reference : 37

10.0 Contribution Log 38

1.0 Introduction

NJIT Student Guide for iOS is a reference guide to students that provides comprehensive information about the campus and also acts as a tool to assist students to carry out tasks such as making an appointment, finding his/her lost item or creating a check list.

The application acts as a centralized information center that administers the students with details such as academic calendar, professor's details and also provides access to NJIT's student newspaper, Vector. NJIT student guide app caters its users by acting as a toolkit to do day-to-day activities such as making an appointment with a professor or reserve playing courts or create check lists to manage activities.

The application accomplishes the aforementioned tasks by storing and retrieving information stored in a custom database created and hosted in Amazon RDS. Further, custom Java code have been written to fetch data from websites, render it in a database storage compatible format and then PHP scripts hosted in an AFS domain have been used to acts interface between the DB and the application.
The application adapts RESTful architecture for client-server interaction and Model - View - Controller (MVC) architecture for implementing the user interface. The application is developed by adhering to extreme programming software development methodology.

2.0 Glossary

1. **Amazon RDS :** Amazon RDS distributed relational database service which runs as a web service in the cloud. This service enables the developers to setup, operate, manipulate and scale a relational database. With the aid of a web service, the information database can be relayed to a mobile application.
2. **REST :** Representational State Transfer (REST) is a platform independent, language independent software architecture style used for web services. It relies on a stateless, client-server, cacheable communications protocol(HTTP/ HTTPS)
3. **JSON :** Java Script Object Notation (JSON) is a data interchange format used by REST services. Data fetched from a DB is serialized in an order of key value pairs and used by client.
4. **Storyboard :** It is a graphical layout of the iOS application that provides a view of the finished application and also provides an overall view of the interaction flow of the app.
5. **Scene :** A scene represents an onscreen content area in an iDevice.
6. **Segue :** A scene represents the transition from one scene to the next scene.
7. **View controllers :** Every storyboard has one/many view controllers. It acts as the a middle man between the user interface and the data. It used to write a logical code which makes the app perform a particular task.
8. **GIT :** Tool used for source and version control within the team.
9. Unity Engine : A 3D game engine, used for image recognition in this project.

3.0 Requirement Specification

1. Appointment Schedule:
- Professor Appointment via email
Lists the professor based on department in a table view and by selecting the professor, a custom email compose panel is opened (NJIT webmail must be logged in the host device) and the email is directly sent to the professor's mailbox.

- Athletic Center Appointment via email
Lists the various categories of games and rooms that can be reserved and on selection, a custom email compose panel is opened (NJIT webmail must be logged in the host device) and the email is directly sent to the athletic center supervisor's mailbox.

2. Vector:
- Vector data
Data is fetched from the official FaceBook page and populated to the user. The data is fetched using HTML extraction where the content alone is extracted and tags are removed. This data is stored in the database and fetched periodically.
- User can search through the vector content and contacts of vector using the search bar on top of the page.

3. Lost and Found:
- A view to display all the lost and found items by users. The user can post a found item by filling up a form and it is posted in the thread. A 'I Lost that' button can be used by the user to retrieve the item by contacting the person who posted it.

4. Academic Calendar:
- A calendar which displays the important dates and events pertaining to NJIT.

5. Know Your Professor:
- Information about professor.
- A profile page which contains info about professor's details such as class schedules, office hours and contact information.

6. Contact information:
- A table which contains contact details about various departments.
- Search contacts by person or department of all faculty and administrative staff of NJIT.

7. Shuttle Stop:
- Show arrival times of various shuttles to and from NJIT. A real time map showing the position of each of the shuttle.

8. What's that building:
- Point the camera to any building of NJIT shows the description of the building. Currently in research for the implementation of the feature which can be realized in many ways such as image stitching and recognition in MATLAB and export it as C code and embed into SWIFT or augmentation in Unity 3D.

9. Sell and buy books:
- A scene containing details of books put up for sale by other users of the app.
- Another scene containing a form for the sellers to provide information of the books they want to put up for sale.

10. NJIT Map:
- A map of the campus through which the users can look up for routes to specific building within the campus. This is achieved by using Apple Maps API.

11. Upcoming events / Broadcast :
- A news feed which contains upcoming events and past events in NJIT where the user could scroll around the information.
- A user can subscribe to a particular event by setting remainders and receive updates.

Extra functionalities Implemented:
1. Click to call functionality in contacts module - The users can click on a contact cell and a call is placed automatically to the corresponding contact's phone number.
2. Share functionality in vector - Enables the user to share the link of an article he is reading via messages, mail, WhatsApp, Facebook, et.al.
3. Search functionality in Know your professor module - Search for professors either by name or department in the KYP module.
4. Search functionality in academic calendar - Users can key in custom search phrase like 'last day to drop classes' and the module will return the deadline to drop classes.
5. ToDo Module - This is a handy check list tool for the user. Provides interface to insert and delete a task. Tick and un-tick task to manage tasks.

3.1 System Requirements :

Mobile Devices:
1. An iOS device running iOS 7.0 or later
2. Must contain a minimum 50MB of free space

3.2 User Requirements :

1. The user must have a working internet connection and allow the app to use the same to access information from Vector or know a professor's details.
2. The app should be allowed to make use of the mail app to send appointment requests to a professor or the athletic center via email.
3. The app should be allowed to place calls while pressing on a phone number within the app.
4. The user should grant permissions for the app to access his/her photo library to select images and post it to the lost and found section.
5. The user should have an NJIT email account to post and claim lost items, this is to validate and only allow students of the NJIT community to report lost items or claim them.
6. The user should allow the user to access the phone's GPS functionality to enable modules like NJIT Maps and Shuttle Services to garner user location.
7. To access functions like identifying a building using what's that building, the application should be granted permission to make use of the phone's camera.
8. Permission to access phone's calendar application must be granted, this is used by the Broadcast module to add an event details such as date, location and time.

4.0 System Architecture :

The system employs a Model-View-Controller (MVC) architecture. The controller and view blocks do not directly interact with each other, they do it with the help of Model. The user requests are sent to the controller. The model class contains the modules needed to fetch data from an external storage and parse it in the format, the view can display. The data is then passed to the view block and this is rendered to the user. As depicted in figure 9, the view equivalent in Xcode is the storyboard, model and the controller are written in separate swift files. In this system's case, the model swift file comprises of a function which is used to make use of a php file hosted in a AFS domain, which in turn fetches data from the DB instance hosted in the cloud and serializes the data, which can be manipulated and used in the view.

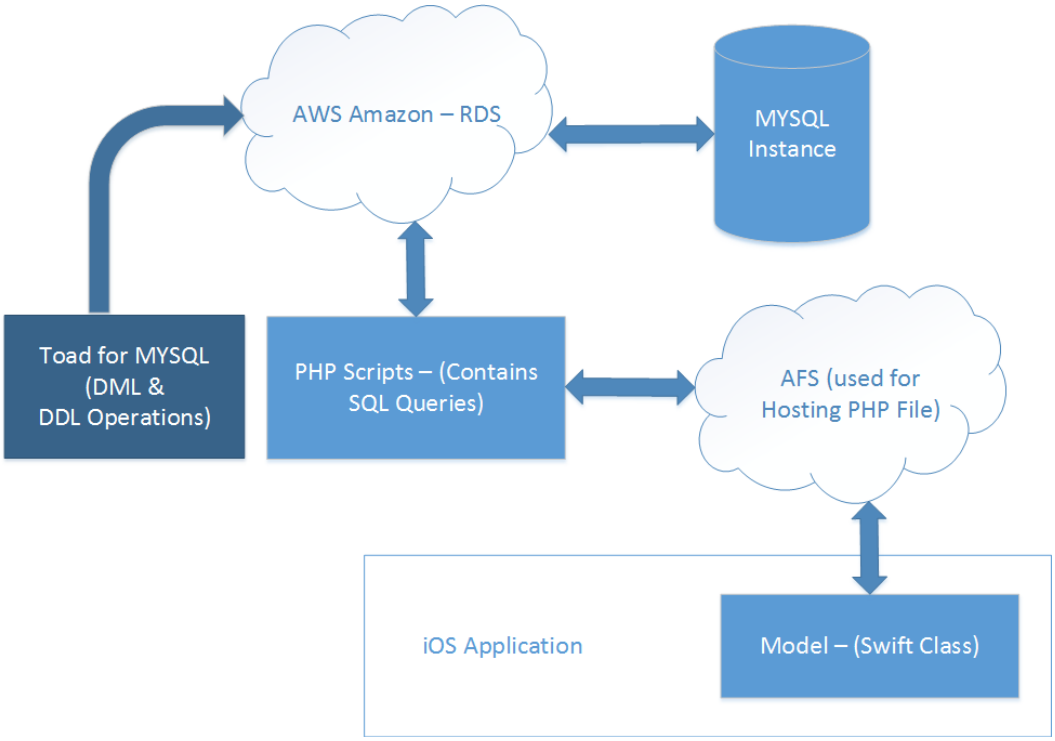


FIGURE 1 : SYSTEM LEVEL ARCHITECTURE.

Figure 1 exemplifies the model block of the aforementioned MVC model in detail. The model block of every module (that requires data from DB) in the app consists of a function which access a PHP file hosted in a AFS domain. The PHP script comprises of pre-coded SQL queries which fetches data from DB in the cloud and returns it to the model block as serialized data in JSON format. This JSON response is then parsed and manipulated by the model block to fulfill the functional needs and then passed on to the view block.

CRUD functionalities in the database are done using Toad, a relational database management tool. Toad is used to execute DML and DDL queries in the relational database hosted in Amazon's cloud.

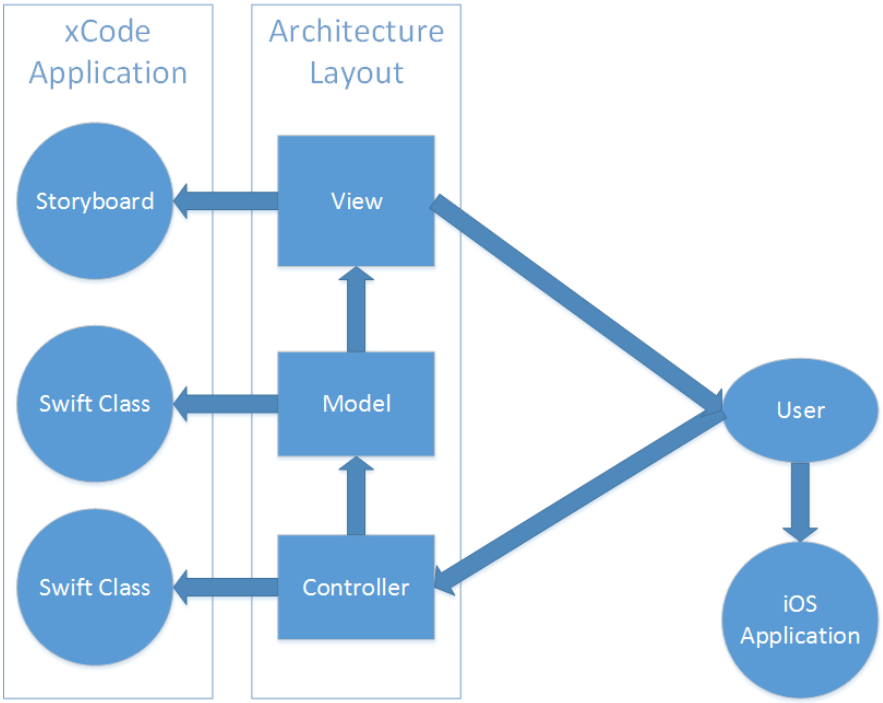
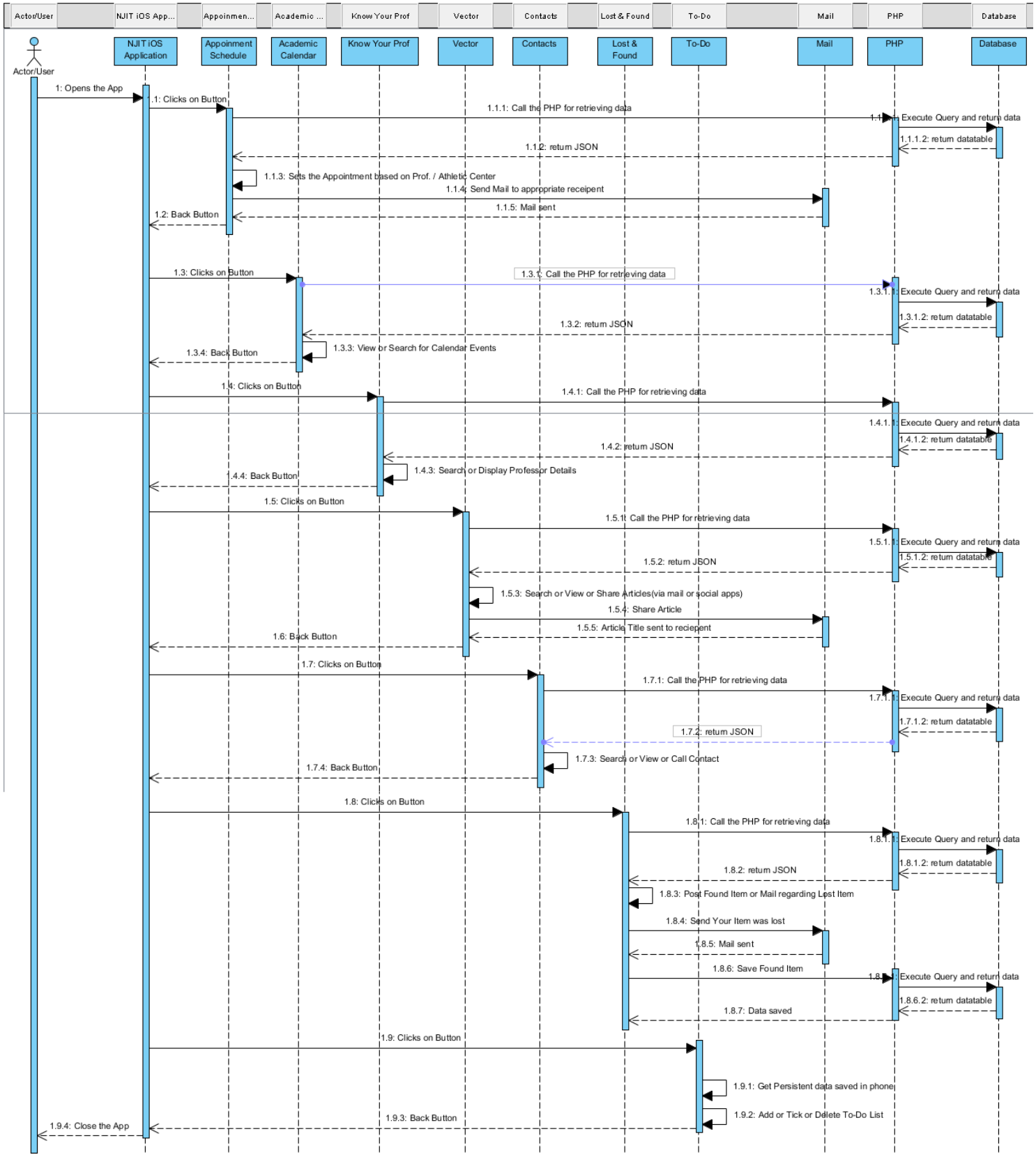


FIGURE 2 : SYSTEM ARCHITECTURE - MVC MODEL

5.0 Overall Sequence Diagram :



6.0 Network Architecture Diagram :

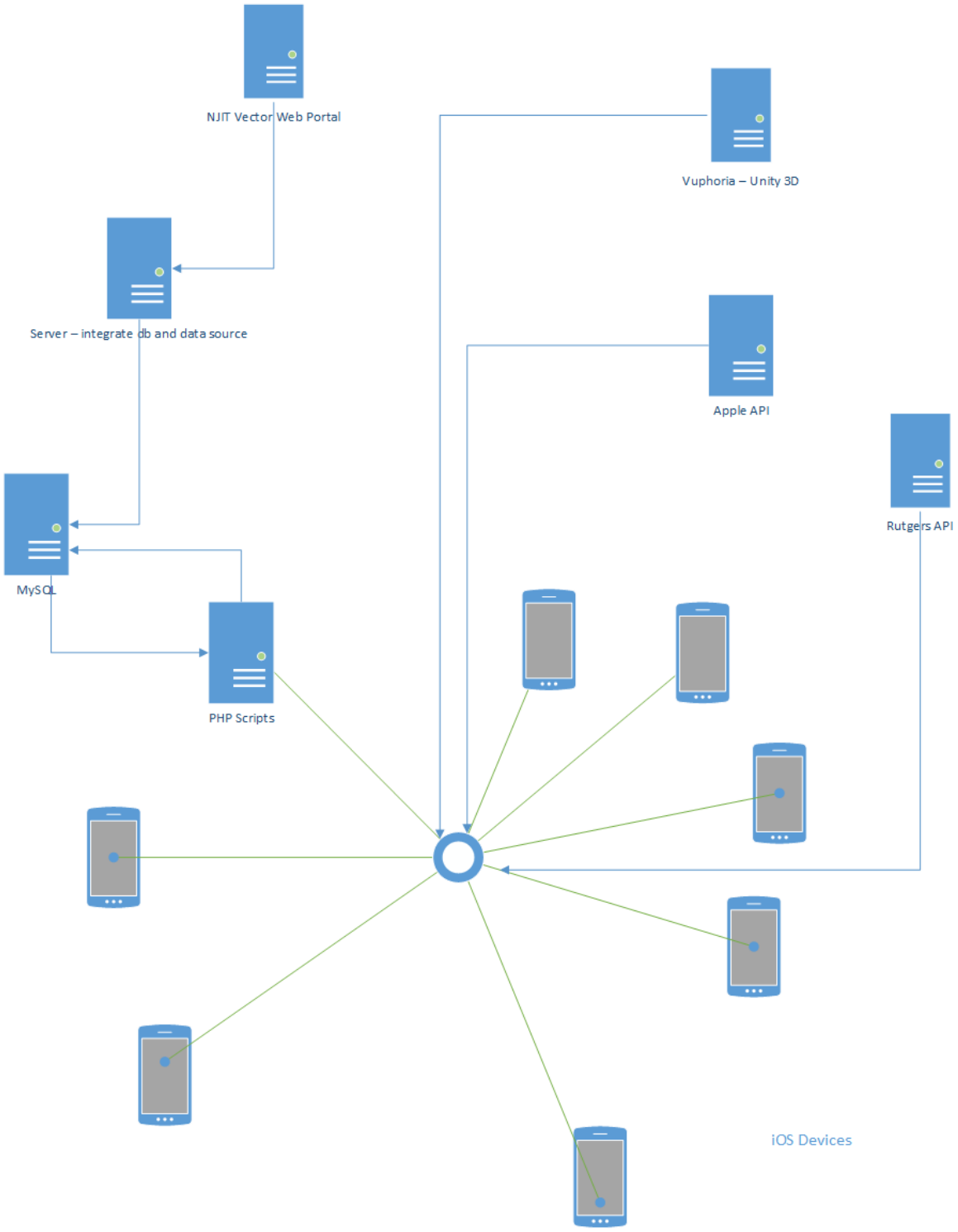


Figure 3 : Brief Network Architecture Diagram

6.0 User Interaction Flow :

FIGURE 4 : HOME PAGE

Figure 4 illustrates the home page of the app. From this page the users can access all the modules specified in the requirements specification.

6.1 Appointment Schedule Module :

In this module the user is presented with a scene where, depending upon the type of appointment he wants to make, either Athletic or Professors appointment scheduler is chosen. Selecting the athletic appointment scheduler presents the user with various playing courts followed by a data and time chooser, upon selecting those, a custom email compose panel is presented with the body of the mail containing the date and time of the appointment requested which can be sent to the supervisor's mail for approval. The process flow is exemplified in Figure 5 and the procedure works in the similar fashion for the professor appointment scheduler too.

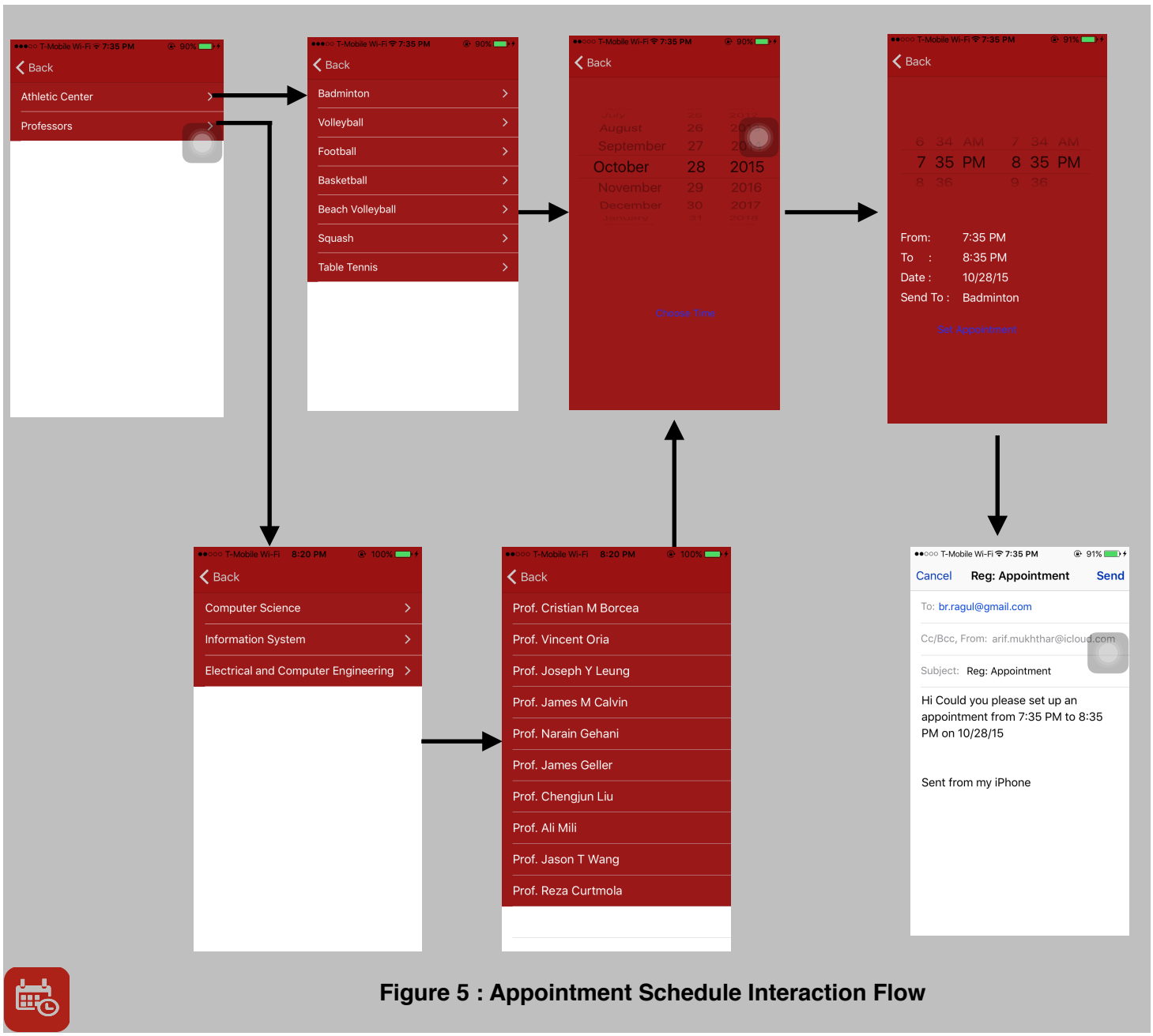


Figure 5 : Appointment Schedule Interaction Flow

6.2 Know Your Professor Module:

Professor's contact can be viewed in this module. The user can click through the interface to view a professor's details. Also the user can search for a professor by his name or view professor's as a set based on the department they work with. This process flow is as exhibited in Figure 6.

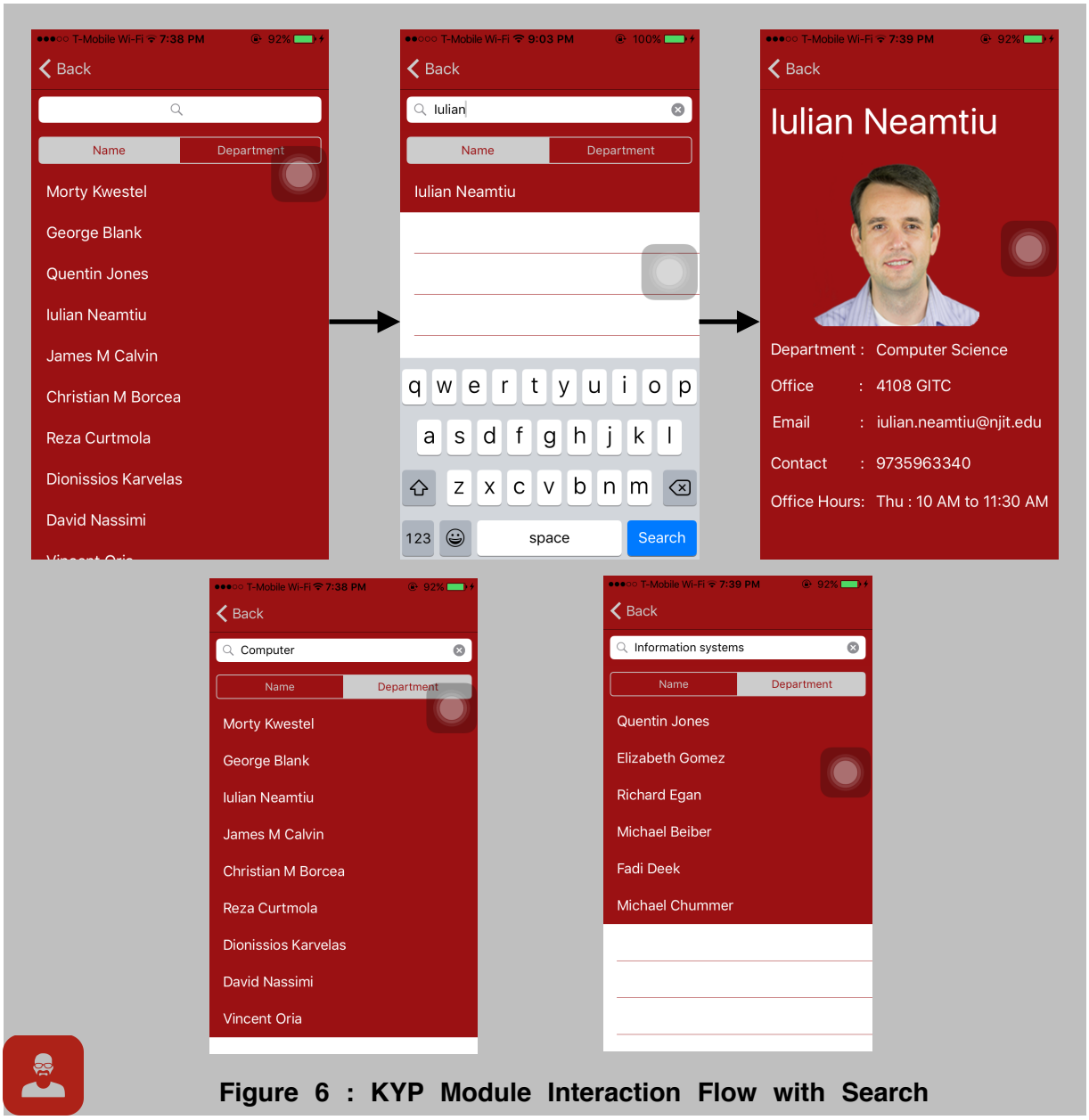


Figure 6 : KYP Module Interaction Flow with Search

6.3 Academic Calendar Module :

The academic calendar module comprises of important dates in the current academic term. The first scene in the module shows the user a set of academic terms for which the important dates are stored. Selecting a term presents the user with the important dates in the semester. Further, user can make custom searches like ‘last day to drop class’ or ‘pay fee’ to view the last date to withdraw from a course or fee payment deadline. Figure 8 shows this process flow.

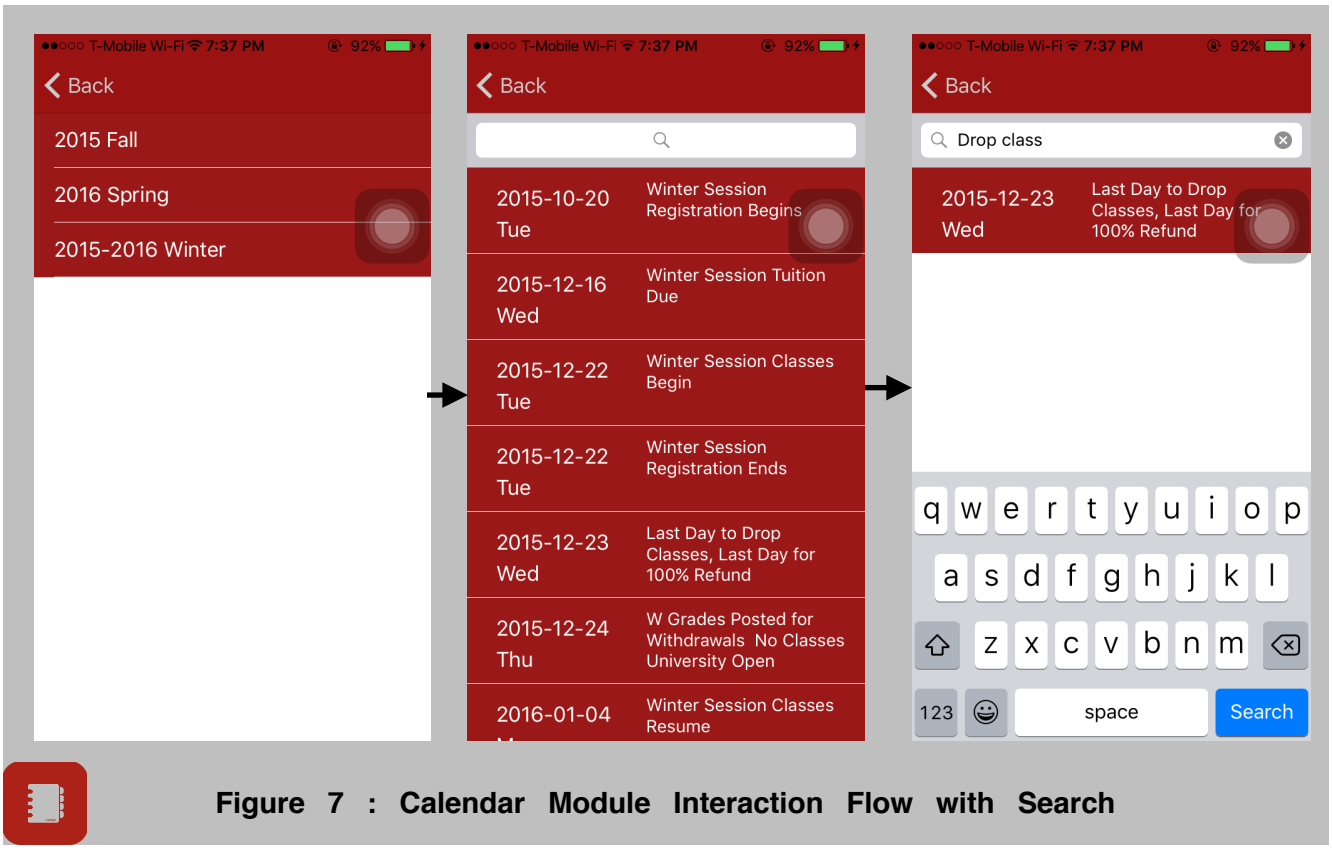


Figure 7 : Calendar Module Interaction Flow with Search

6.4 Vector Module :

This module displays articles published by the NJIT newspaper, Vector. The users can view the article's title along with the date it was published in the first scene and clicking on the article moves the control to the second scene where the user can read the complete article presented in a scrollable view. In this scene the user can also share the article they are reading via message, WhatsApp, Facebook, et.al by clicking on the share button. Vector module also boasts a search functionality which enables users to search for an article by title.

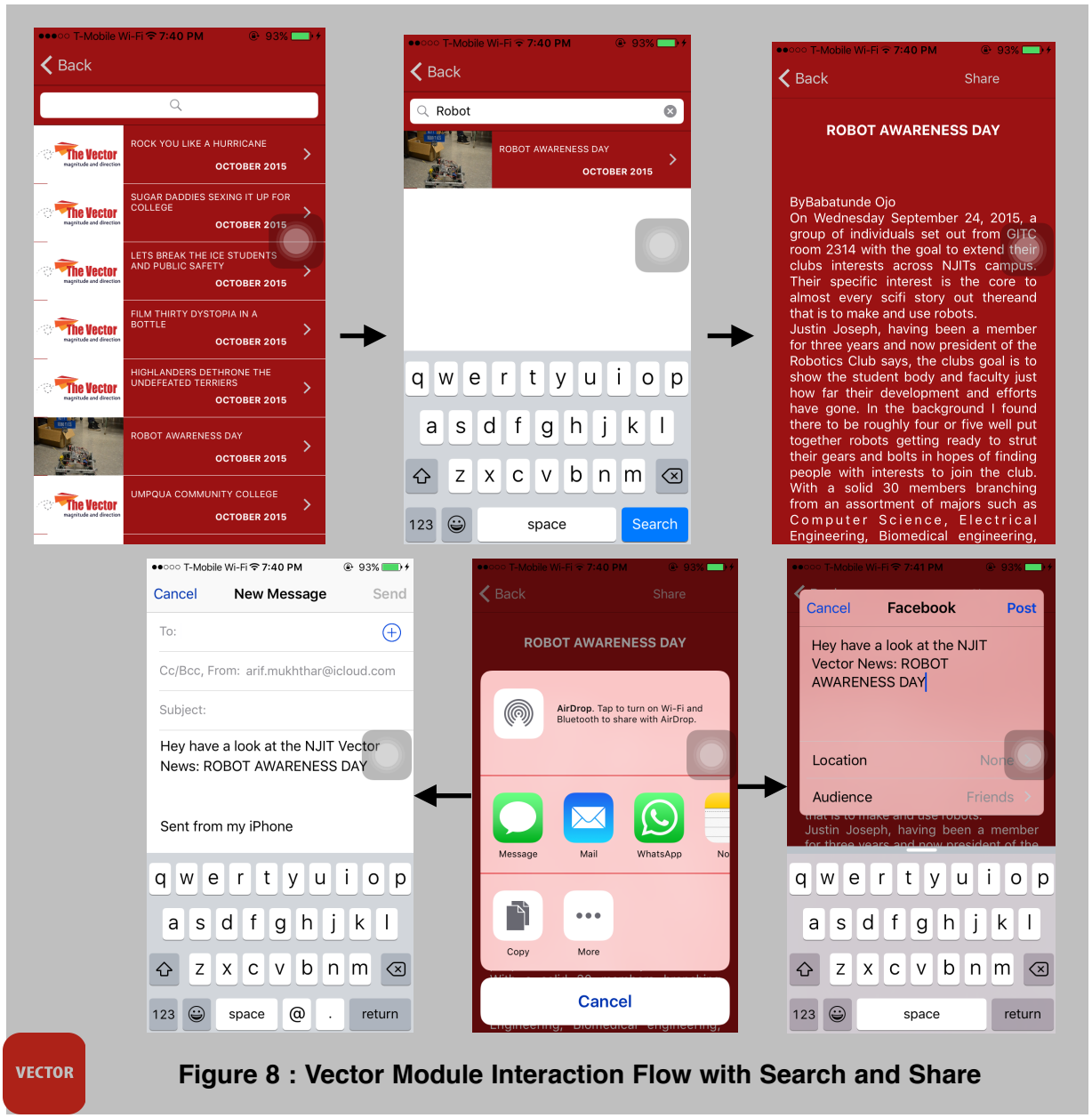


Figure 8 : Vector Module Interaction Flow with Search and Share

6.5 Contacts Module :

Contacts module contains contact details of various departments in NJIT. This is a single scene module. Search functionality has been incorporated in to this module too. Clicking on a particular contact would place a call using the stock phone app. The descriptive flow is as exemplified in Figure 9a and Figure 9b.

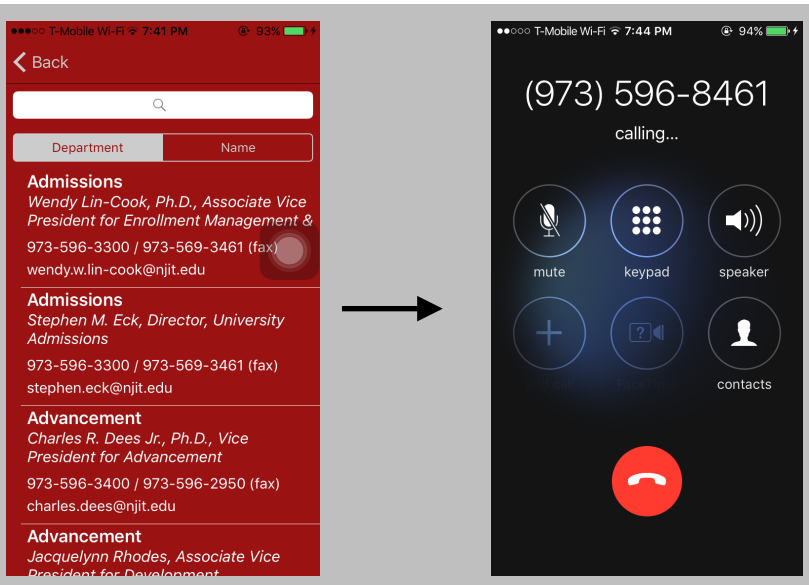


Figure 9a : Contacts Module Interaction Flow with Click to call Functionality

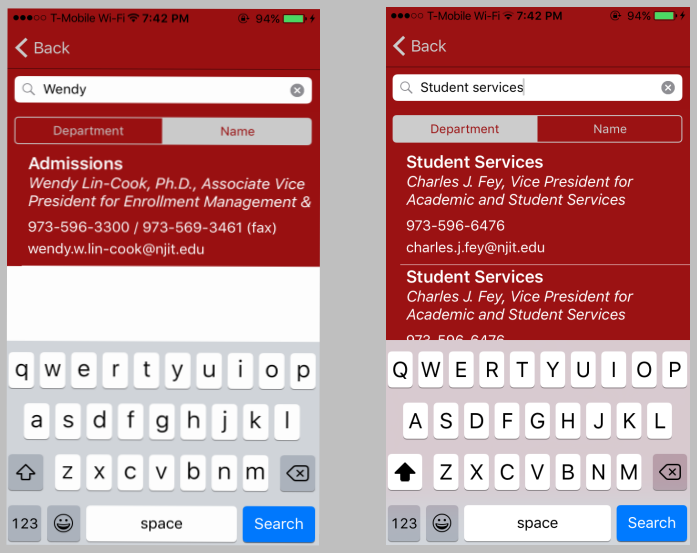


Figure 9b : Contacts Module Interaction Flow - search by name & department

6.6 Lost & Found Module :

Lost and found module helps the user to post a detail about an misplaced item he/she stumbles upon. Another user who uses the app and has misplaced the item can look up the lost & found module and if he finds an entry of an item that matches his misplaced item's description then he can press a button, this enables him to get in touch with the user who has found the item through email. Figure 7 below shows the descriptive flow.

Figure 7 : Lost & Found Module Interaction Flow

6.7 ToDo Module :

This module consists of a single scene. It enables the users to create a check list by entering the tasks and clicking a button. The view provides the check list items created in order. Click on a task will display a tick mark against the task. The user can also un-tick a task and also completely delete it from the view. The figure below demonstrates the module's functionality.

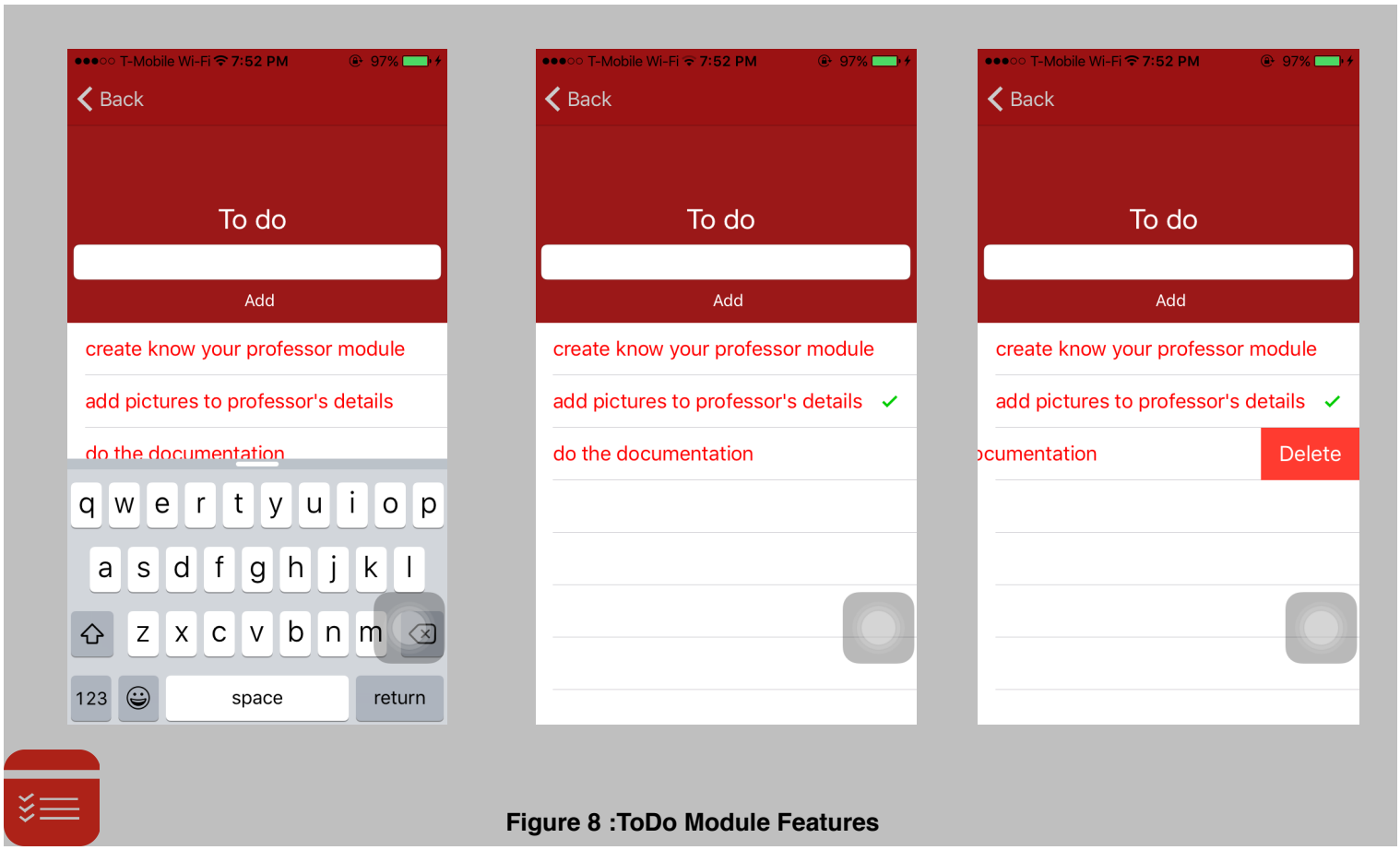


Figure 8 :ToDo Module Features

6.8 Broadcast :

Broadcast module comprises of the rsvp details of various events that are scheduled to take place on campus. There are two scenes in this view controller. The first scene consists of a list of upcoming events along with their dates. Clicking on a event will take the user to another scene which provides detailed information of the event, which also has a button to add the event to the user's calendar.

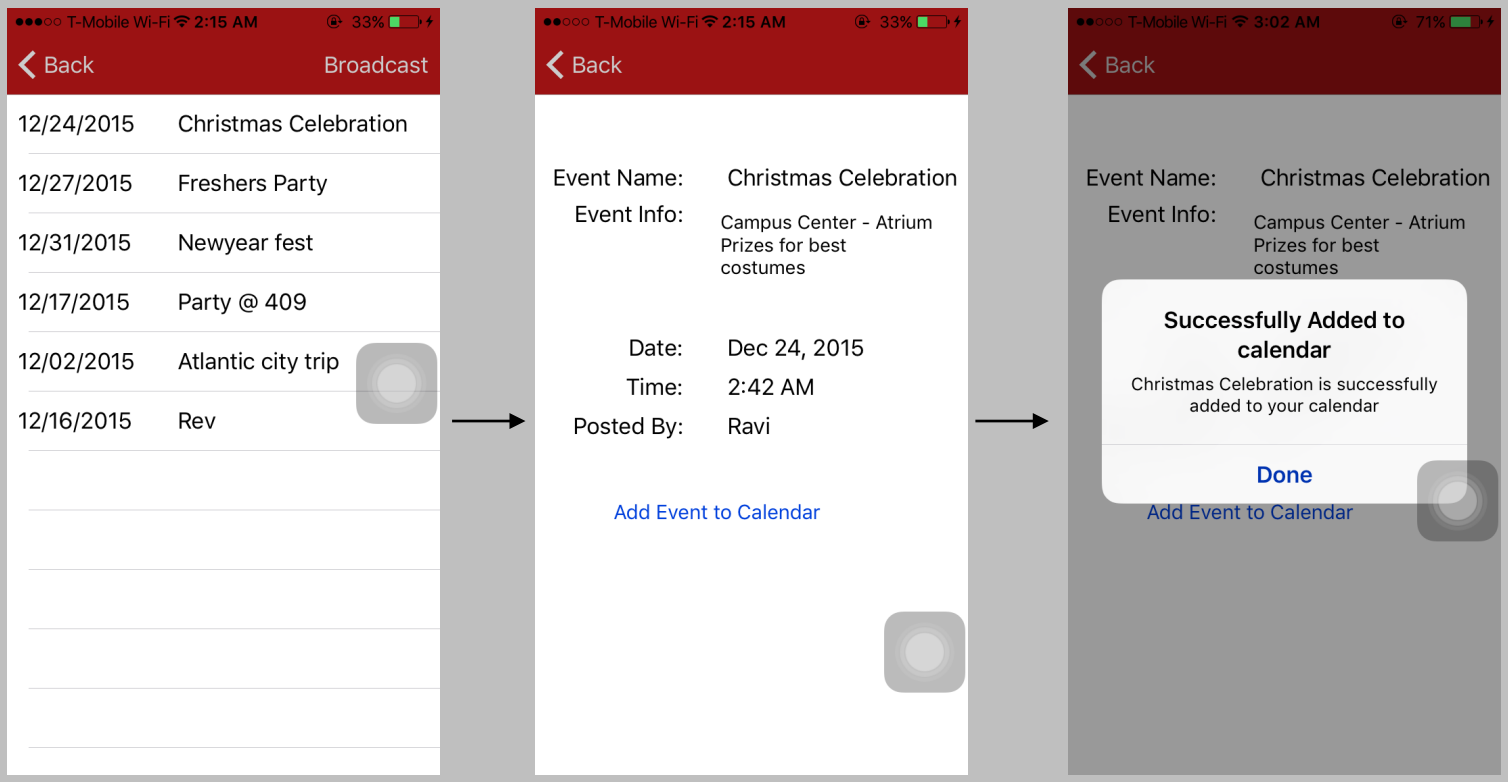


FIGURE 9 : BROADCAST MODULE INTERACTION FLOW

6.9 What's that building?

The what's that building module helps in user to identify a building by using the phone's camera. The user after entering the module can point the camera at a building and the app will display the building's name.

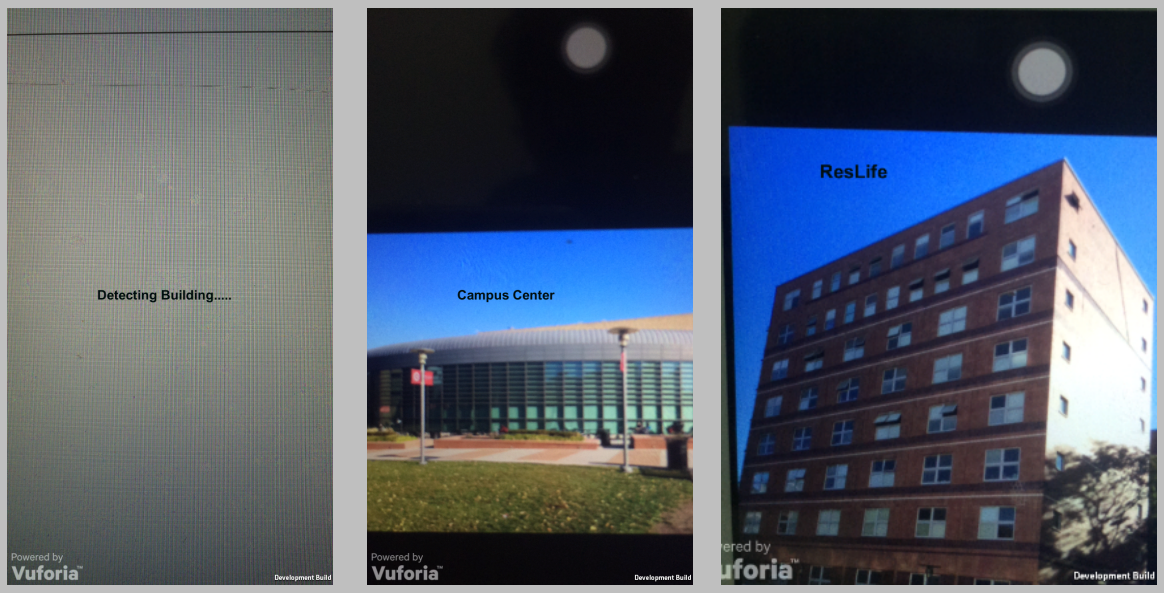


FIGURE 10 : WHAT'S THAT BUILDING MODULE INTERACTION

6.10 Shuttle :

This module provides the user with various shuttle routes and also bus arrival predictions for each stop. Users can also use the get direction feature to find the route to the nearest shuttle stop.

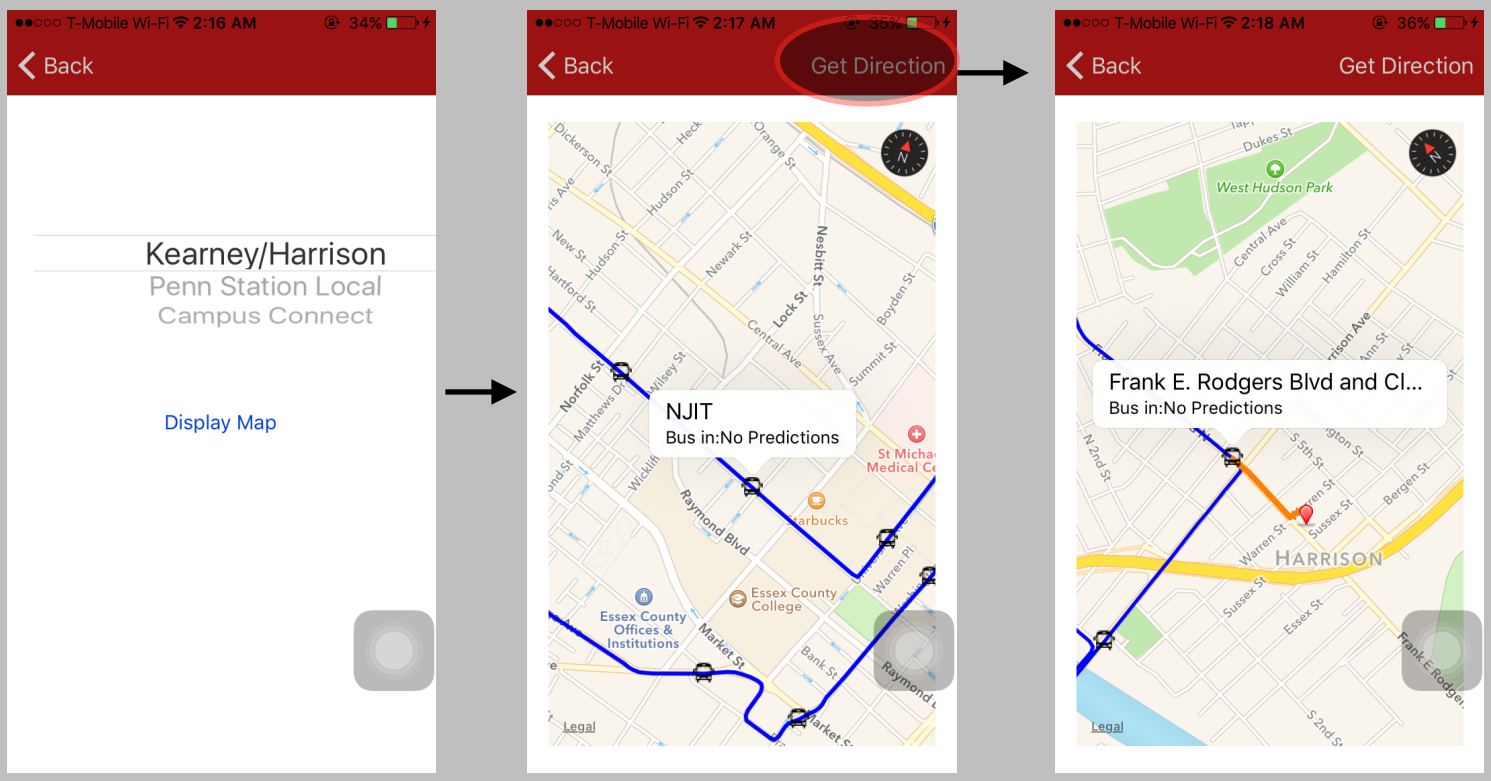


FIGURE 11 : SHUTTLE MODULE INTERACTION FLOW

6.11 Buy and Sell books:

Buy and sell module consists of two scenes. One scene displays a list of books along with their details and the other scene displays a form through which an user can post a book to be sold. There is also a feature implemented, which enables the buyer to contact the seller via email.

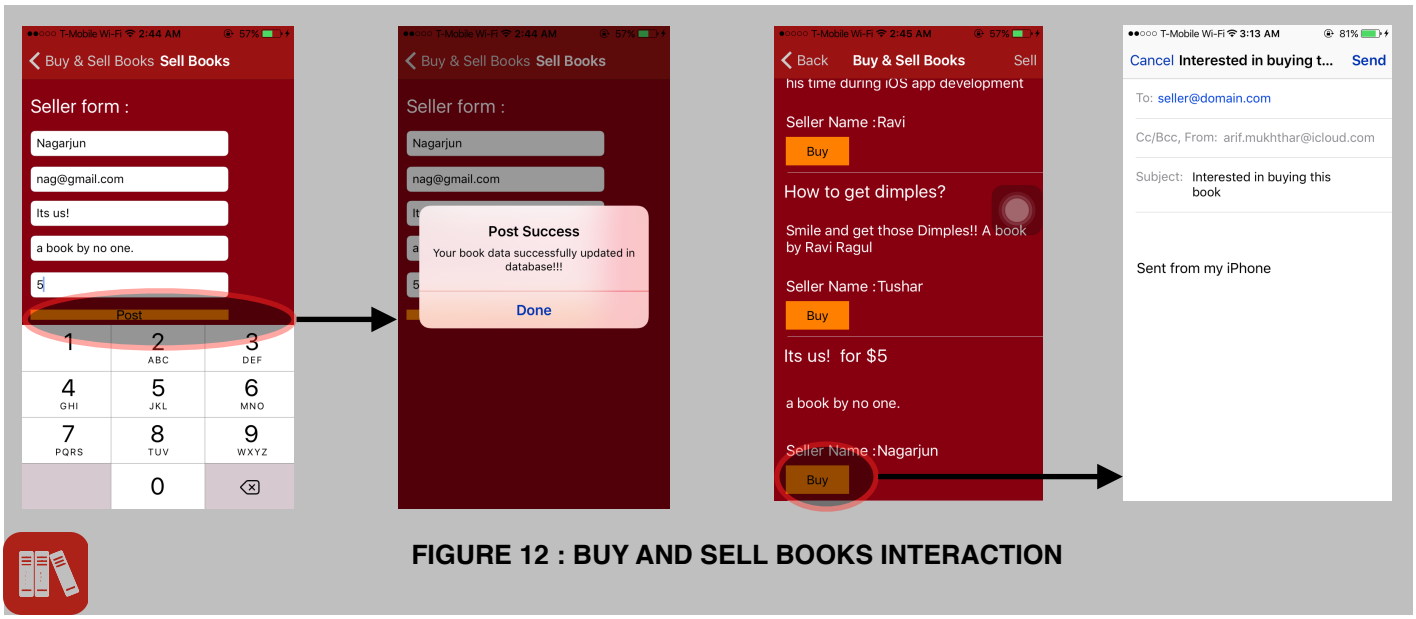


FIGURE 12 : BUY AND SELL BOOKS INTERACTION

6.12 NJIT MAPS

To view the internal map of NJIT, users can use this module. NJIT MAP module not only has a static map image, but also helps the user find routes to a particular building of his choice. This is achieved by providing two scenes for each of the aforementioned functionalities.

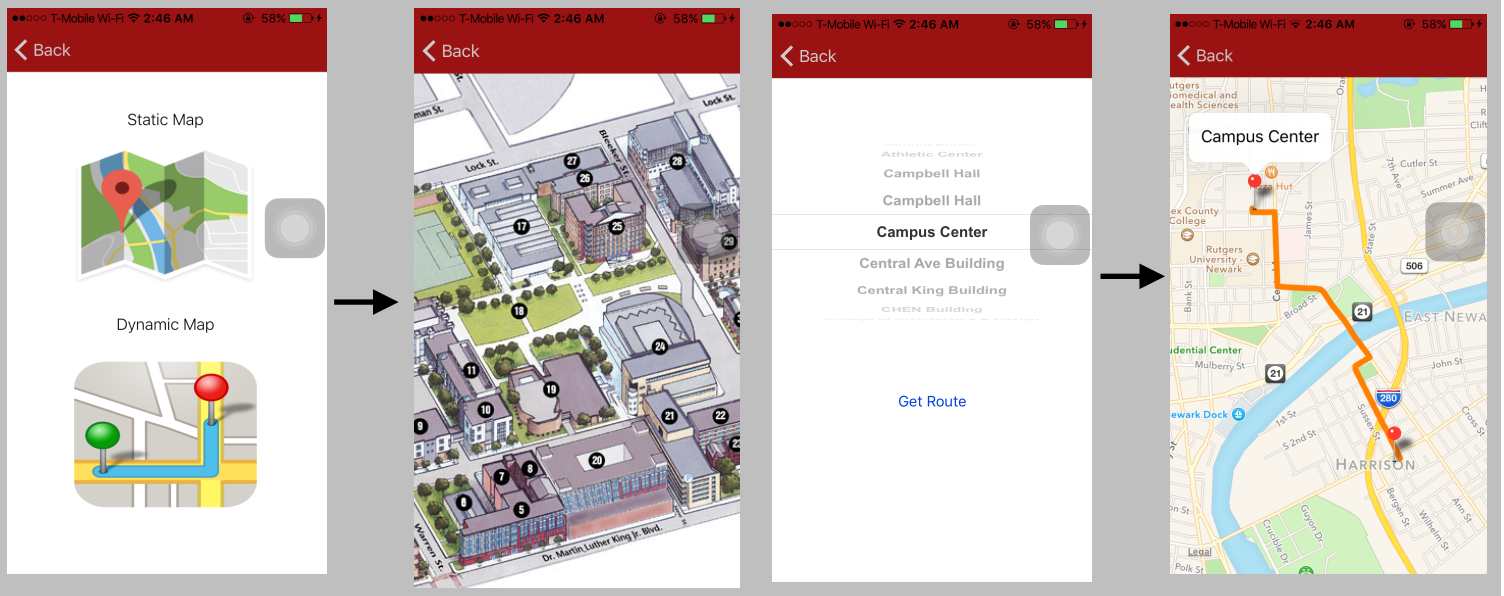


FIGURE 13 : NJIT MAPS INTERACTION FLOW

7.0 Data Flow Diagrams :

7.1 Appointment Schedule Module :

In the appointment schedule module the user places a request to either fetch data from Athletic center details table or Professor details from the database in the cloud. This is achieved with the help of a PHP script written exclusively to fetch data from the DB mentioned above and parse it into a serialized array and sent back to the application. After fetching the details, the user can select the entity with which an appointment has to be set up and use data and time picker to select a time and send it as an email to the concerned party for approval. Figure 9 charts out the data flow between various sub modules.

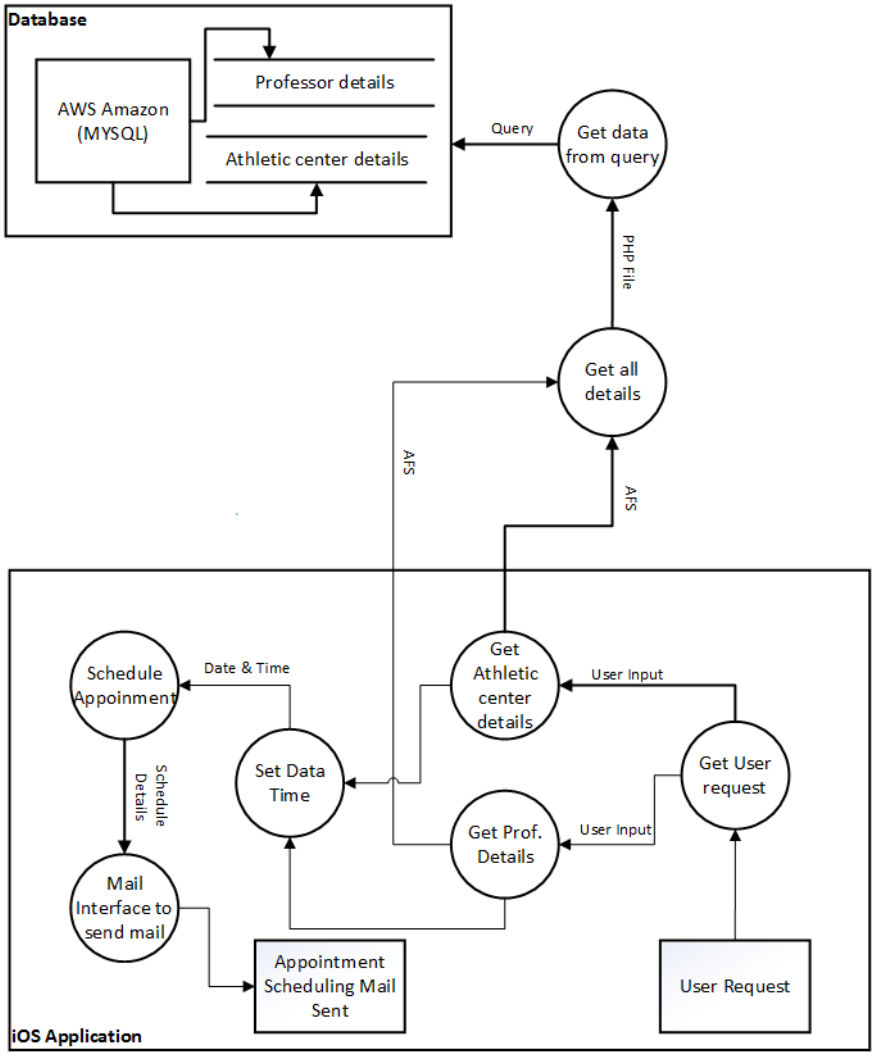


Figure 14 : Data Flow Diagram - Appointment Schedule Module

7.2 Academic Calendar Module :

In case of the academic calendar module, the user can select a term for which he wants to view calendar for. Upon selecting the term, the module uses a customized PHP script which fetches data from an amazon RDS DB and sends it back in JSON format. This data is then parsed and manipulated to be used in the view block. Search functionality helps the user search for a particular day or event without using SQL queries.

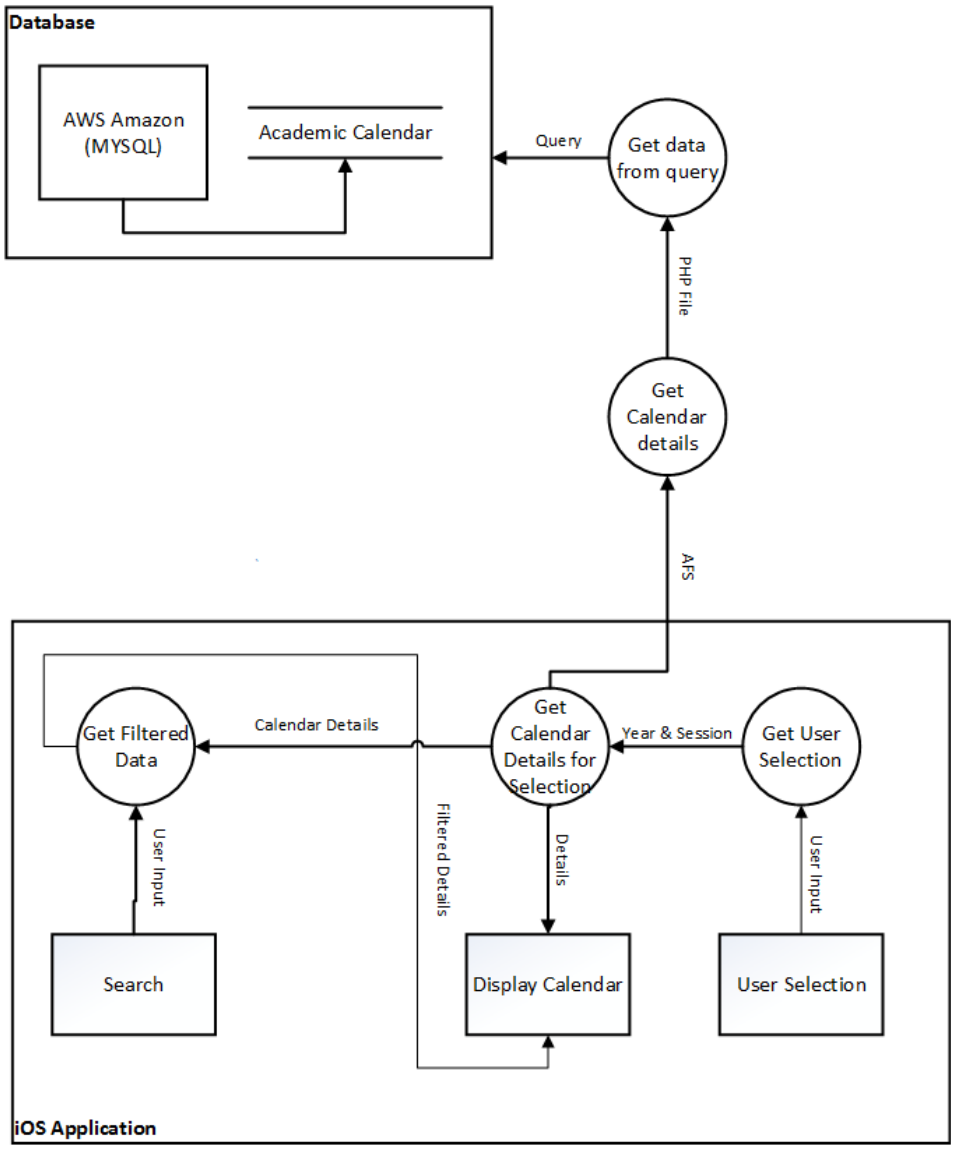


Figure 15 :Data Flow Diagram - Academic Calendar Module

7.3 Know Your Professor Module :

The Know Your Professor module also uses a backend DB connection to store professor's detail. When searched for a professor, PHP script fetches data from the backend DB and converts it into JSON serialized array in a key- value pair fashion and is sent to the application. The professor's details are then displayed in a separate view. To view the image of a professor, the image URL stored in the database is fetched and is set to an image object which can fetch the image of the professor and set it to the image frame during runtime.

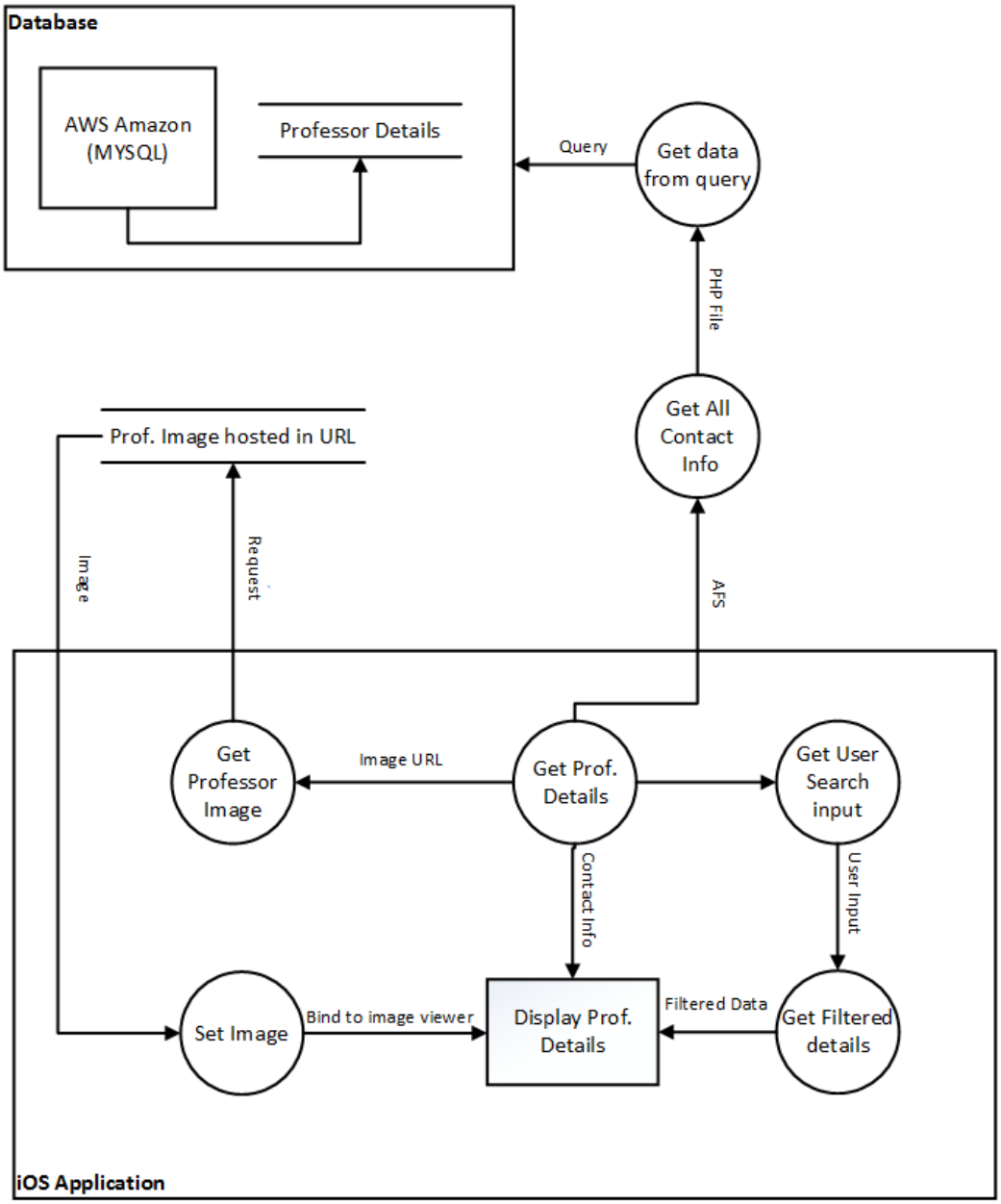


Figure 16 :Data Flow Diagram - Know Your Professor Module

7.4 Vector Module :

In the vector module, a JAR file comprising of Java classes which employs a custom algorithm that can filter out and segregate text from an HTML file. In this system, the articles from the vector magazine's website is filtered out in a text format and based upon the tags present in the html files, the title and the articles are segregated and inserted into a MySQL databases hosted in the amazon cloud.

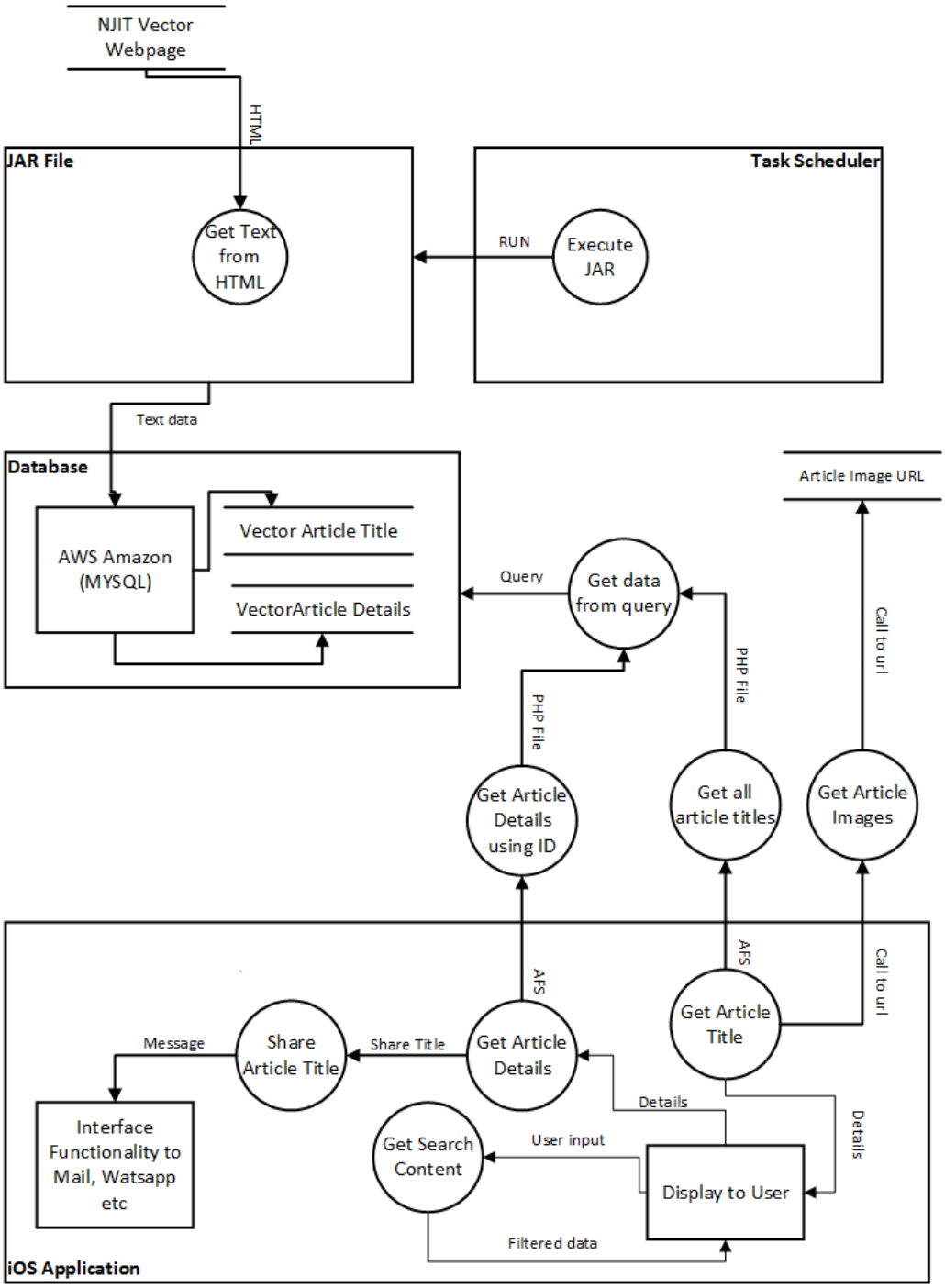


Figure 17 : Data Flow Diagram - Vector Module

When a user loads the vector scene 1 in the application, by using a PHP script hosted in an AFS domain, the data from the vector database is fetched and fed to the application in JSON format. Upon manipulating the JSON format, articles are displayed in the scene.

Clicking on an article takes the user to second scene where he/she can view the article in detail.

Search functionality has been incorporated into the vector module. This enables the user to make use of an in-built search controller which can return the article the user is looking for as he keys in the article's title in the search bar.

Another feature implemented in the vector module is the ability to share an article via messages, email, whatsapp, facebook, et.al. This is done by using the activity view controller built into swift.

Taking into consideration the space it would take to store an articles image in the DB, a work around has been devised to fetch images directly from the vector's webpage and using them alongside the article's title in first scene.

So as to keep the database up to date as new articles are published in vector's website, a scheduled task has been setup in operating system to run the HMTL to text converting JAR file periodically.

7.5 Contacts Module :

The contacts module displays the contact details of various departments in NJIT campus. Figure 18 exemplifies the data flow in the module.

This module contains a single scene only. A PHP script file hosted in an AFS domain is tailored to fetch contact details from a backend database, stored in the amazon cloud, and return it back to the application in JSON format. This data relayed by the PHP script is manipulated and presented to the user in a TableViewController in the scene. Clicking on a contact name would call a function that would pass the telephone number of the contact beyond the app's sandbox into the iPhone's phone application, which places a phone call.

Search functionality has been added to this module. After the data is populated into the scene, the search controller is used to search for a contact and present the user with search results in the same scene itself.

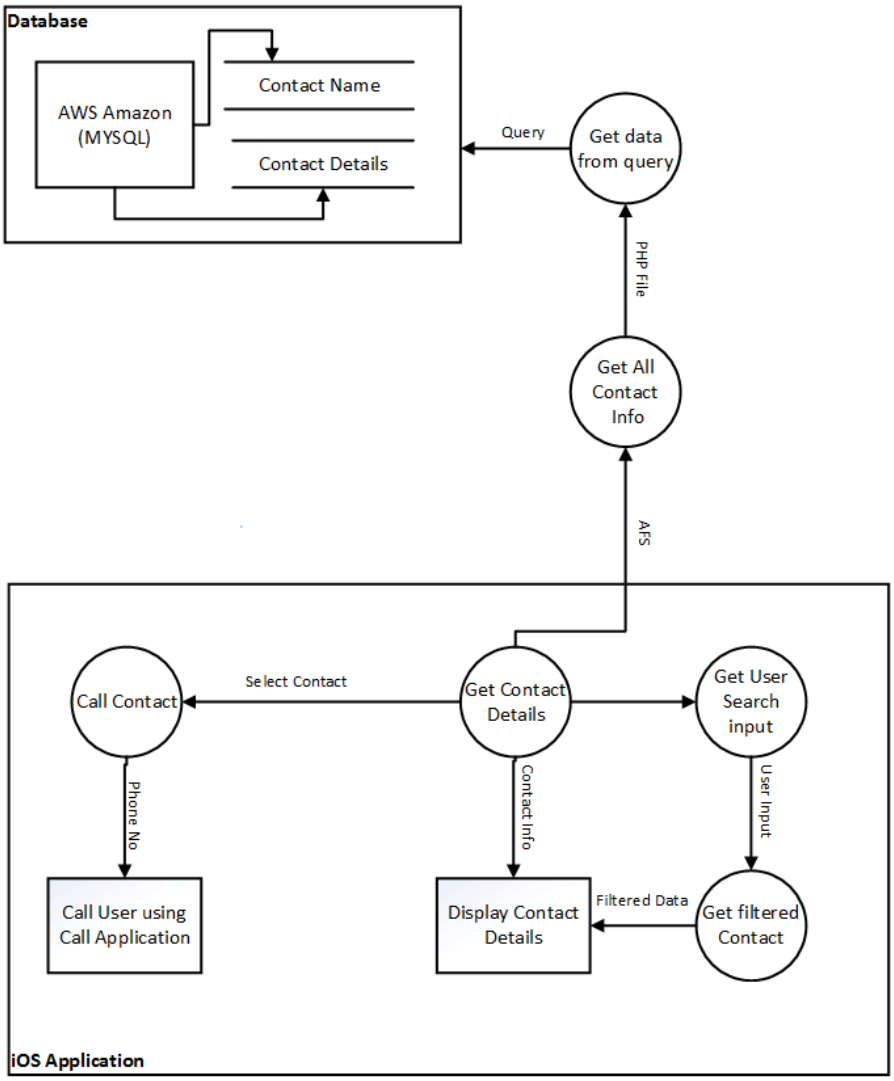


FIGURE 18 : DATA FLOW DIAGRAM - CONTACTS MODULE

7.6 Lost and Found Module :

Lost and Found module has a two way interaction with the user.

The module has two scenes and one which has the details of the lost items which are fetched from the database in amazon's cloud. A PHP script file acts as a middleware to fetch data from the DB and return it as JSON to the Lost and found module. If a user recognizes an item he has missed, he can click on the 'I lost it' button, which takes the user to a compose email screen with the email id of the person who had found the item into the field.

The module's second scene is navigated to by clicking on the POST button in the scene one. This takes an user to a form, where he/she can fill up the details of an item lost by another person. The details from the form is converted to JSON format and sent to another PHP file which is coded to insert the details into a database. This is the database that is used to populate the lost and found module. The comprehensive data flow diagram is given below,

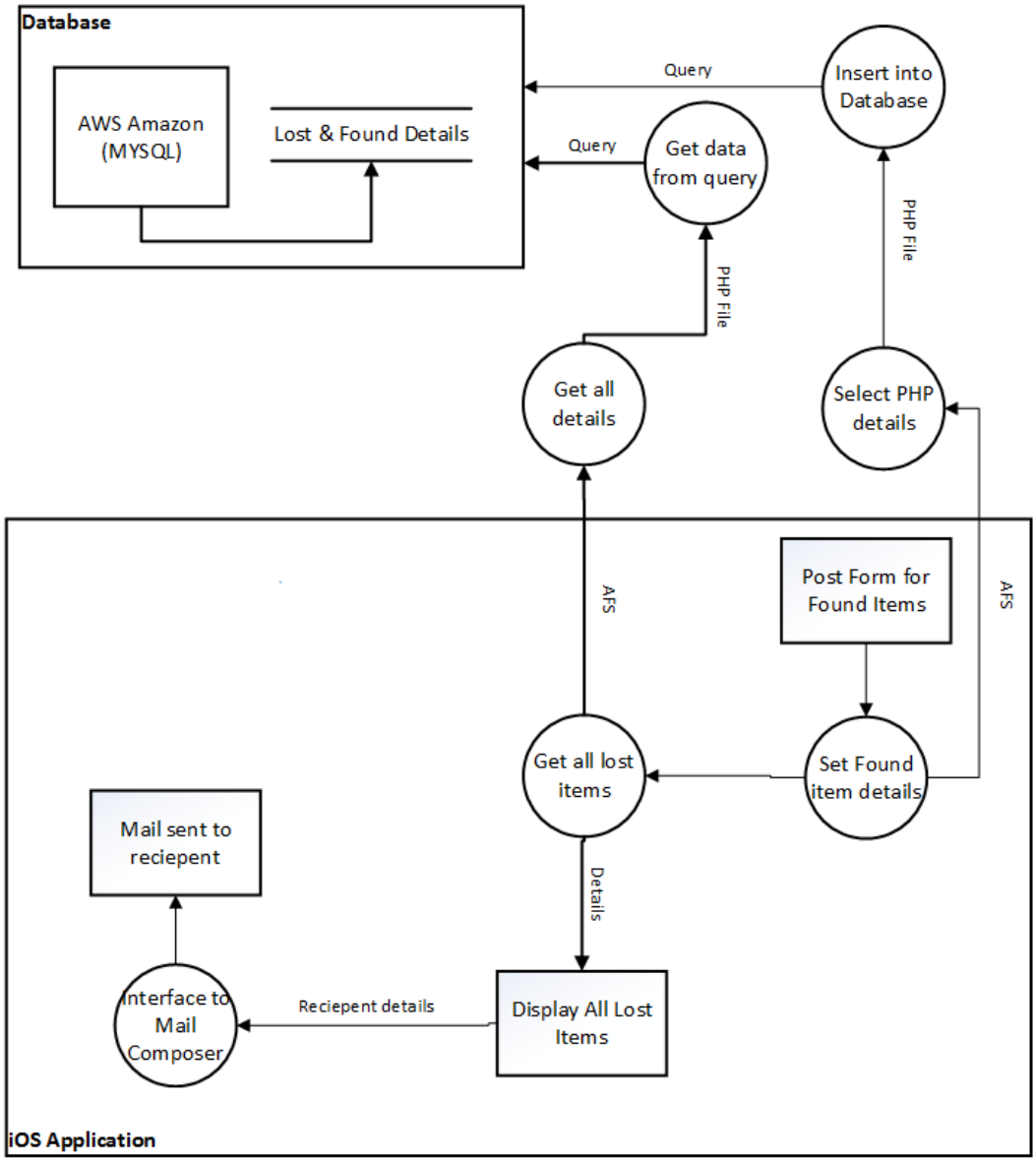


Figure 19 :Data Flow Diagram - Lost and Found Module

7.7 Broadcast Module :

A webpage has been developed for event admins to log in and post event details. These details are sent through a PHP script and inserted into a database stored in Amazon AWS.

In the client side, the module has two scenes. The first screen fetches event details data like event name and date from the database stored in amazon AWS and populates the table view controller, PHP script hosted in an AFS server acts as an middleware here. Clicking on an event takes the user to a second screen, where details like the event's rsvp are fetched from the DB and displayed.

The module makes use of the Reminder functionality, which is a built-in functionality in iOS, to add the event details to the phone's calendar application.

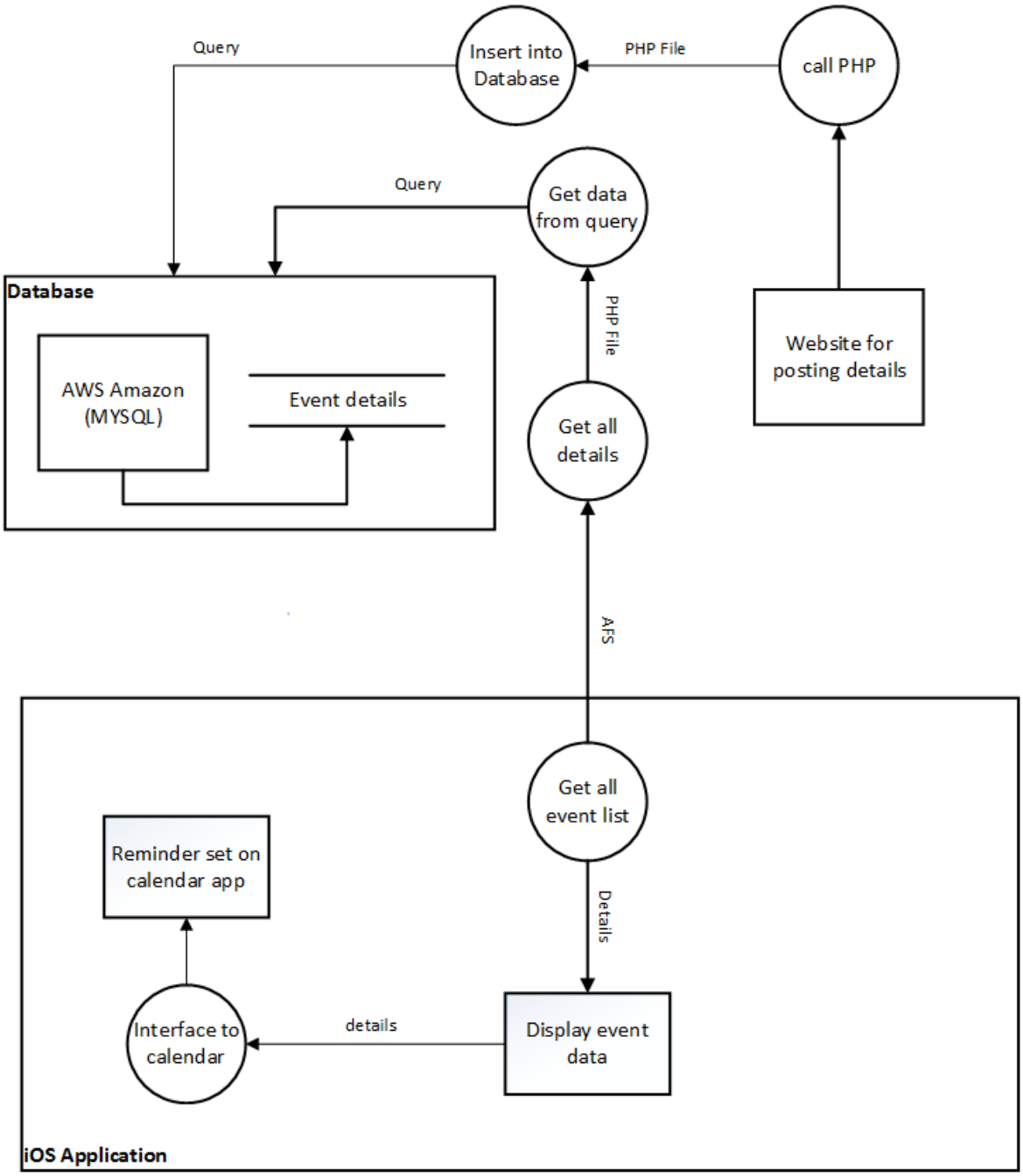


Figure 20 :Data Flow Diagram - Broadcast Module

7.8 NJIT MAPS Module :

This module comprises of two sub modules. A static map image stored in the local database is displayed in the static map image view controller.

For the dynamic maps module, a database stored in amazon AWS database is use to store the latitude and longitude values of all the buildings in NJIT campus. In the client side, the user's current location (coordinates) and also the location of the building the user chooses to go to is sent to the to the DB and this data are passed as parameters to Apple Maps and the routes are plotted dynamically.

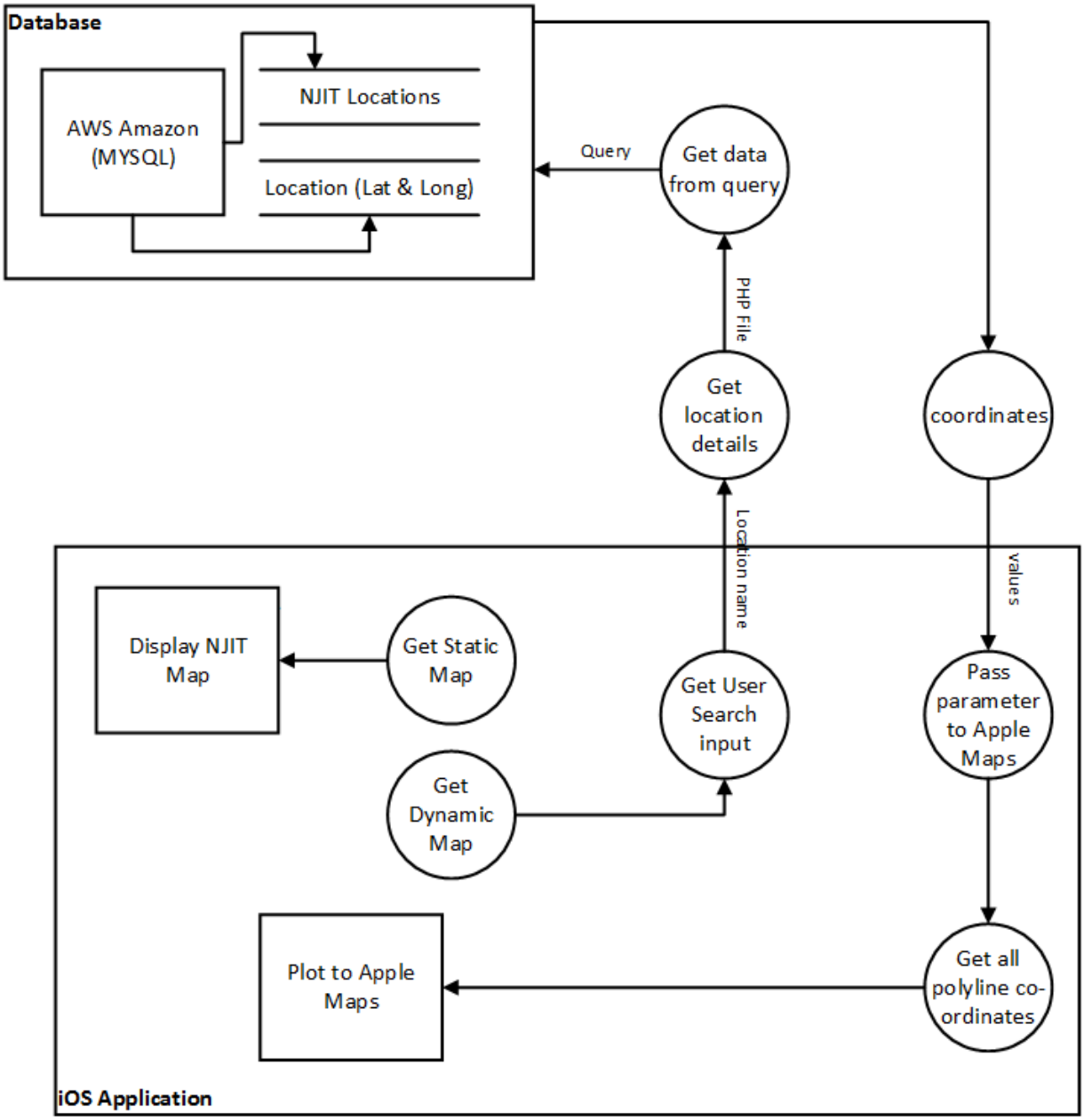


Figure 20 :Data Flow Diagram - NJIT Maps Module

7.9 Buy and Sell books Module :

Buy and Sell books module has a two way interaction with the user.

The module has two scenes and one which has the details of the books put up for sale, which are fetched from the database in amazon's cloud. A PHP script file acts as a middleware to fetch data from the DB and return it as JSON to the Buy and Sell books module. If a user recognizes finds a book of interest, he can click on the 'Buy' button, which takes the user to a compose email screen with the email id of the person who had put up the book for sale.

The module's second scene is navigated to by clicking on the POST button in the scene one. This takes an user to a form, where he/she can fill up the details of a book to be sold. The details from the form is converted to JSON format and sent to another PHP file which is coded to insert the details into a database. This is the database that is used to populate Buy and Sell books table view controller. The comprehensive data flow diagram is given below,

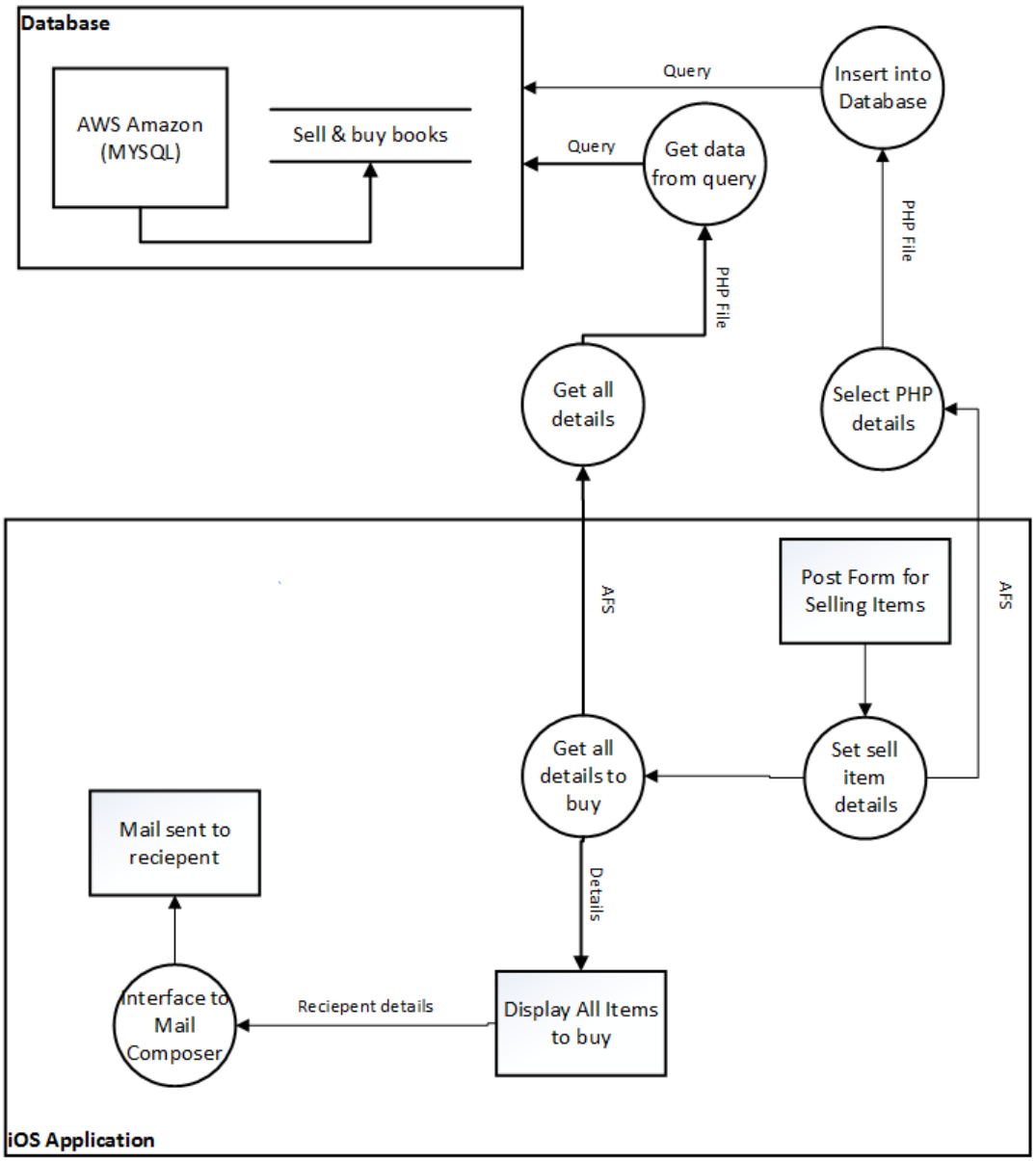


Figure 20 :Data Flow Diagram - Buy and Sell books Module

7.10 Shuttle Module :

Shuttle module makes use of the Rutgers API to get coordinates of shuttle stops and also live coordinates of shuttles. The coordinates are then passed as parameters to Apple Maps and the plotted with annotation pins. Clicking on a shuttle stop pin would load the name of the stop and bus predictions data. The coordinates of route, stop and bus location is obtained from Rutgers API website. The bus location is fetched for every five seconds and updated in the apple map.

We can also get the directions for the stop. The user location is also loaded in to the map. By clicking on bus stop and selecting get direction button the route is obtained from apple api and plotted in the map.

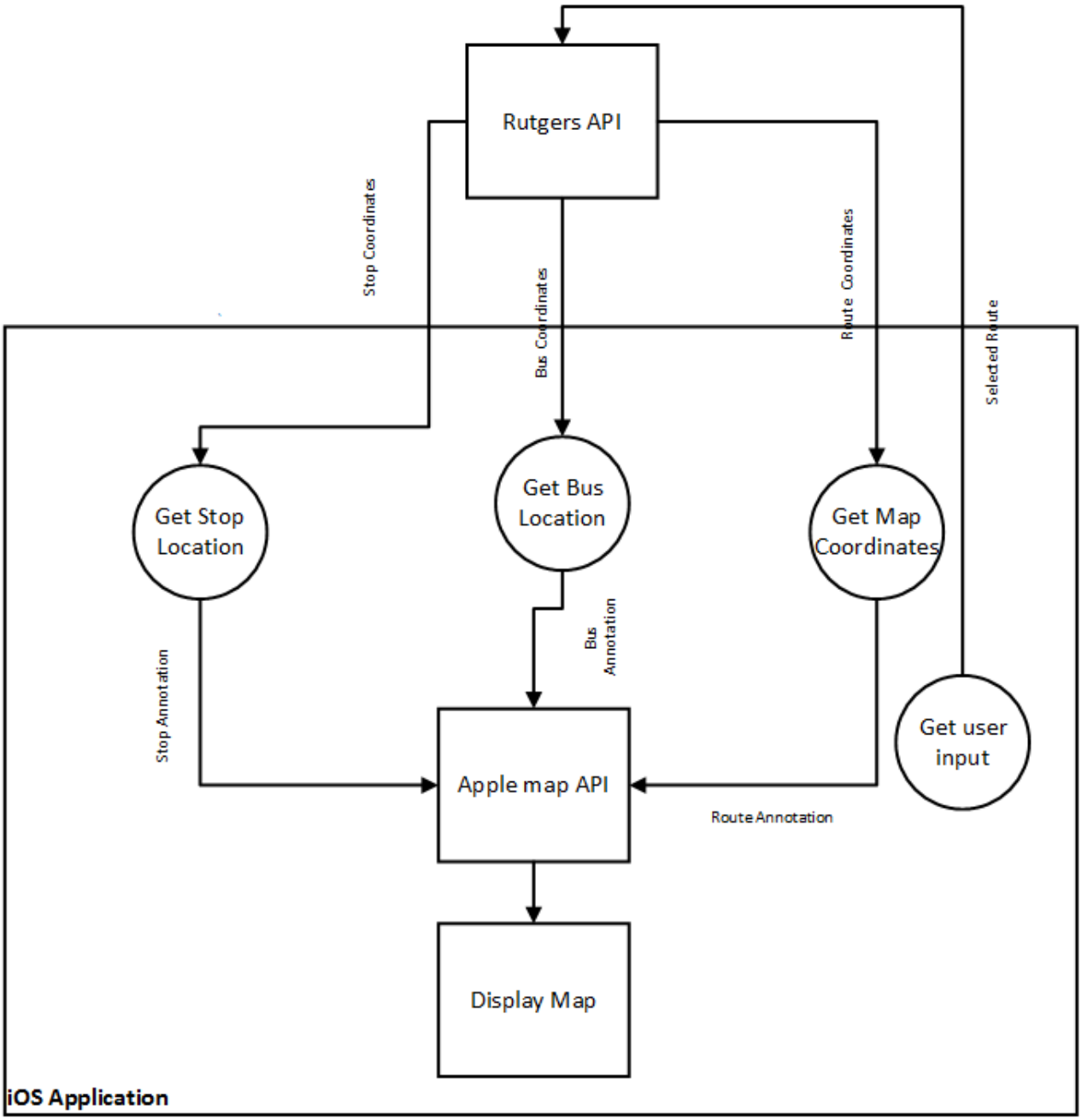


Figure 21 :Data Flow Diagram - Shuttle Maps Module

7.11 What's that Building :

The ARCamera component acts as the device camera at the device level. The 3D models of the building are built and made datasets in the Vuforia developer library which is downloaded and deployed in the app. The data from the camera that is the frames of images from the device input is pushed through the output as metadata objects which is taken by the Vuforia QCAR object model. This metadata datasets is continuously compared with the existing dataset in the object model library every time the frame is modified and once a match is found, a trigger is set which calls the TrackingEventHandler of that specific dataset and if the tracking is found it sets the text in the UI to the corresponding building's name of the dataset.

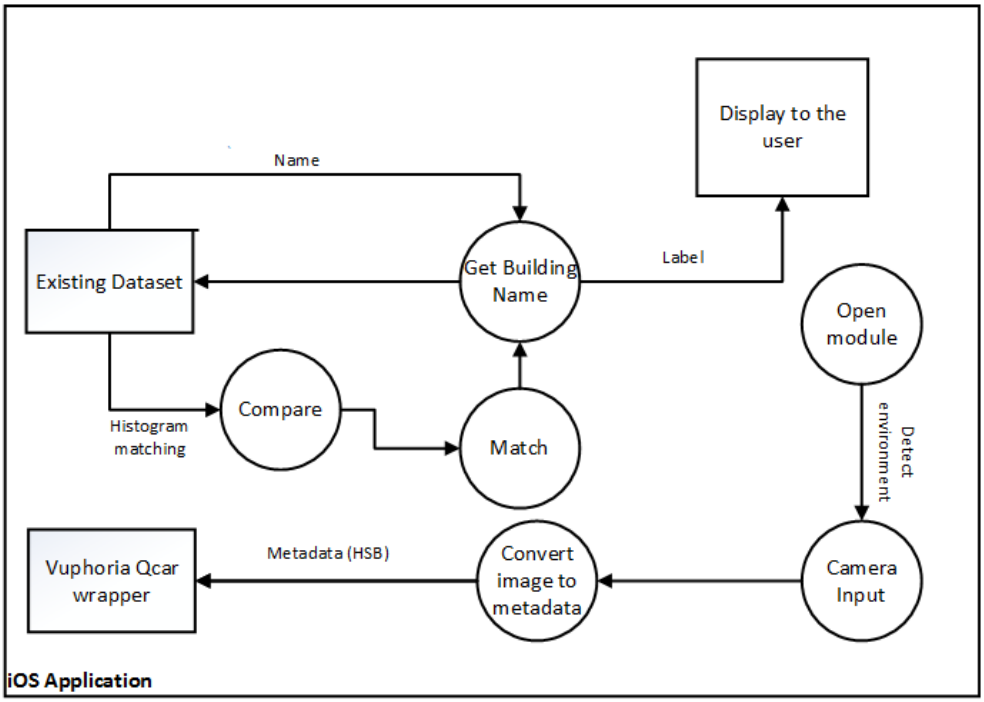


Figure 22 : Data Flow Diagram - What's that building

8.0 Test Case:

TEST CASE 1.1

Description: **Verify application compatibility with different Apple iPhone version 4S, 5, 5C, 6, 6S and iPad. Verify that application can be installed successfully.**

Expected Result: **Application is installed successfully.**

Pass/Fail Criteria: **If the content does not appear obviously installed on the device then the test is a fail.**

Test Steps

1. **Install the application being tested. The application must install without error.**
2. **Verify the application has been successfully installed on the iPhone and iPad by navigation to the install location. The application should be represented as an icon in the location**

Actual Result: **Application successfully installed in all devices**

Pass/Fail: **Pass**

TEST CASE 1.2

Description: **Verify all buttons in the application is working correctly and calling the respective view controllers**

Modules Applicable: **All modules**

Expected Result: **All buttons should call the respective view controllers correctly.**

Pass/Fail Criteria: **If the button is calling the wrong view controller or not working then the test is a fail**

Test Steps

1. **Install the application being tested. The application must install without error.**
2. **Verify that all buttons when clicked, goes to correct pages and no error should occur**

Actual Result: **All buttons working correctly and calling the respective modules**

Pass/Fail: **Pass**

TEST CASE 2.1

Description: **Verify whether the user cannot set date in date picker less than current date (today) in appointment scheduling module**

Expected Result: **User should not be able to set less than today**

Pass/Fail Criteria: **If the date can be set to less than today's date then the test is a fail**

Test Steps

1. **Open the application and click on appointment scheduling button**
2. **Navigate to Professors->Click any Department->Click on the Professors Name, a date picker should appear in next page**

3. **Verify whether you can set the date less than today's date**

Actual Result: **Cannot set date less than today's date**

Pass/Fail: **Pass**

TEST CASE 2.2

Description: **Verify whether the user cannot set from time greater than to time in appointment scheduling module**

Expected Result: **User should not be able to set from time greater than to time**

Pass/Fail Criteria: **If the from time can be set to greater than to time then the test is a fail**

Test Steps

1. **Open the application and click on appointment scheduling button**
2. **Navigate to Professors->Click any Department->Click on the Professors Name->select date->Set from time and to time click set appointment**
3. **Verify whether you can set the from time greater than to time**

Actual Result: **Cannot set from time greater than to time. A 'Time Conflict' warning is displayed**

Pass/Fail: **Pass**

TEST CASE 3.1

Description: **Check if using the search function in all modules returns relevant data from the database.**

Expected Result: **Display the relevant data which matches the search alphabets and presents it in the table view.**

Pass/Fail Criteria: **If the search doesn't fetch data relevant to the search query then the test is a fail.**

Test Steps

1. **Open any module with the search functionality and click on the search bar.**
2. **Key in the search field the data that needs to be searched.**

Actual Result: **returns the relevant data in the table view.**

Pass/Fail: **Pass**

Modules using Search functionality: KnowYourProfessor, Vector, Academic Calendar, Contacts Information

TEST CASE 4.1

Description: **Verify the data loaded in table view controller is valid in all modules**

Modules Applicable: **Lost and Found, Broadcast, Vector, Appointment module, Academic calendar, Know Your Professor, Buy and Sell books,**

Expected Result: **Data is displayed correctly**

Pass/Fail Criteria: **If the content does not appear, the test is a fail.**

Test Steps

1. **Open the application and click on the respective module**
2. **Verify every record from the database is fed to the table view. The contents of the text field match with contents of the database records and objects**

Actual Result: **Data is displayed successfully with match conditions**

Pass/Fail: **Pass**

TEST CASE 4.2

Description: **Verify if an empty dataset, invalid email address is given to the Post Form it throws an alert to the user.**

Modules Applicable: **Lost and Found Module**

Expected Result: **An alert box is thrown to the user with message "Enter all fields" or "Enter the \(specific) field"**

Pass/Fail Criteria: **if an empty dataset from the form is saved, test is a fail.**

Test Steps

A.

1. **Leave blank for th text fields of User Name, User Email, Item Name, Item Description and click Save.**
2. **Verify the alert view being called and shows message that the text fields are empty.**

B.

1. **Leave any of the text fields blank and click save.**
2. **Verify the alert view being called and show message the specific text field is empty.**

C.

1. **Enter an invalid email address and other valid fields and click save.**
2. **Verify the alert view being called and shows message that it is an invalid email address.**

Actual Result:

A. Threw an alert stating that text fields are empty.
B. Threw an alert stating the specific field empty.
C. Threw an alert stating invalid email address.

Pass/Fail: **Pass**

TEST CASE 4.3

Description: **Verify whether mail composer exceptions are handled in all modules.**

Modules Applicable: **Lost and Found, Appointment Schedule Module, Vector**

Expected Result: **User should be able to send request to retrieve the item through in built mail composer.**

Pass/Fail Criteria: **if the mail composer does not open up or if the 'to:' field not set in the mail composer, test is a fail.**

Test Steps

A.

1. **Open the application and click on lost and found button**
2. **Item cell → I lost that (button)**
3. **Verify whether mail composer opens up and sets the 'to:' field to the respective user if mail composer set up in the device**

B.

4. **Open the application and click on lost and found button**
5. **Item cell → I lost that (button)**
6. **Verify whether alert is thrown if mail composer not set up in the device**

Actual Result: **A. Mail composer opens up and 'to:' field is set.**
B. Alert is thrown stating mail composer is not set up for the device.

Pass/Fail: **Pass**

TEST CASE 5.0

Description: **Verify user login in Broadcast website to post an event.**

Expected Result: **User must be able to post only after he/she enters correct username and password.**

Pass/Fail Criteria: **If the event is posted without correct username and password then the test is a fail.**

Test Steps

1. **Open the application and click on Broadcast**
2. **Click on Post button on Navigation bar**
3. **Enter wrong user name and password, then click login**
4. **Verify whether you are able to login**

Actual Result: **An error is displayed to enter valid user name and password**

Pass/Fail: **Pass**

TEST CASE 6.0

Description: **Verify whether the direction is correctly plotted in NJIT map and Shuttle map modules**

Expected Result: **The direction should be plotted correctly in the map according to specified coordinates**

Pass/Fail Criteria: **If the direction is wrongly plotted then the test is a fail**

Test Steps

1. **Open the application and click on NJIT map or Shuttle map module**
2. **Click on the get direction or show map button**
3. **The direction from the user location to selected building of shuttle stop must be correctly plotted**

Actual Result: The direction in the map is displayed correctly

Pass/Fail: **Pass**

TEST CASE 7.0

Description: **Verify whether the building captured in what's that building module is identified correctly**

Expected Result: **The building names of images captured should be identified correctly**

Pass/Fail Criteria: **If the building name is not displayed or a wrongly identified then the test is a fail**

Test Steps

1. **Open the application and click What's that Building module**
2. **Capture the image of the building in camera opened in the app**
3. **Check whether the name of building is displayed correctly**

Actual Result: The name of different buildings is displayed correctly in the app.

Pass/Fail: **Pass**

9.0 Reference :

- https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID309
- https://www.youtube.com/channel/UCGdtWQJp0g6QnhAhmB7PnQw
- http://documents.software.dell.com/toad-for-MySQL/12.7/user-guide
- https://aws.amazon.com/documentation/rds/

10. Contribution Log:

Milestone 1 Contribution Log :

We as a Team have worked on various activities like Requirement Analysis, Feasibility Study, Coding, Debugging, Integration Testing, Peer Testing, Feedback analysis etc.

Requirement Analysis

Framed the Software requirement specifications – gathering inputs, ideas and suggestions from the group.

Feasibility Study

Check the complexity of the Requirement and determine if they can be developed in real time scenario with the available resources and time.

Coding

Forming regular sessions so that any feature which is generic is added to others as well. Took up coding sessions to facilitate the development stage.

Debugging

Helping peers to overcome issues and errors whenever there is a bottle neck situation. Continuous brainstorming for finding the solution.

Integration Testing

Individual Modules are taken up by each person but integrated to a common storyboard. Integration testing done in groups so that the work of one individual is not hindered by the other.

Peer Testing

There is always a room for improvement and we made sure that by peer testing and provided the knowledge about the module at the same time to our peers. This avoids dependency of a person to any module at any situation.

Feedback Analysis

After rigorous development the application was put forth among students for getting their views on the application and the feedbacks where noted and necessary changes were made according to the inputs provided.

Documentation

As a group we are working on the document template and including the parts which were managed by each individual.

As an INDIVIDUAL

Module Split-up

	Percentage of Contribution
Tushar Swaminathan NJIT Maps Broadcast (Upcoming Events) – Website creation for event entry Documentation - Data Flow Diagrams & Architecture Pattern Diagrams	**25%**
Srinivasan Sundaramoorthy Whats that building UI & UX designing (For the entire application) Testing & Debugging	**25%**
Ravi Ragul Balakrishnan NJIT Rutgers Shuttle Broadcast (Upcoming Events) – iOS frontend creation and DB activities	**25%**
Arif Mukhthar Umarkayam Shakilabanu Buy and Sell books Documentation – Template preparation & Documenting Generic features	**25%**

Tushar Swaminathan

UCID: ts336
Team: Team 4

Modules Worked

NJIT Map
Broadcast (Upcoming events)

Tools Used

xCode	:	iOS development
Toad for MySQL	:	DDL & DML activities
Brackets	:	Webpage development
AngularJS & jquery	:	Webpage designing
Notepad++	:	PHP scripting
GIT	:	Version Control

Cloud Sources

AFS	:	PHP script hosting
AWS AMAZON – RDS	:	MYSQL instance hosting

NJIT Map

Functionalities

Dynamic Map - Live GPS location & directions
Static NJIT Map

Work Done

The NJIT Map module consists of two types of maps – Static and Dynamic map. The Co-ordinates for various locations in NJIT is stored into the database and depending upon the user request the co-ordinate is fetched and displayed. Static map is an added feature which displays the overall NJIT map as a **static scrollable image**. The dynamic map, on user request fetches the coordinates from MYSQL database through a PHP script. These co-ordinates along with the users current location coordinates are then sent to the Apple Maps API to get the direction list. The directions are then plotted using a **polyline** concept. Place markers are also added for denoting the users coordinate and the destination coordinate.

Broadcast – Upcoming events

Functionalities

Login feature for pushing upcoming events
Webpage for pushing upcoming events

Work Done

The Broadcast module allows the users to receive notifications from various clubs and organizations of NJIT, but the user who is posting the event may or may not have an iOS device. In order to overcome this issue a standalone web application is built made up of **AngularJS, jquery, PHP and HTML**. The user is provided with a LOGIN from the administrator through he / she can access the web application. A login

validation is also provided for better privacy and to avoid unwanted posting. Once logged in the user can enter the details of the events in the provided fields and save the post. A message will be displayed to the user indicating if the post is uploaded or not. The PHP is used here to take the value from the webpage and insert it to the MYSQL database hosted in Amazon RDS. The HTML file along with the AngularJS and dependent files are hosted into the AFS environment through which the website is running. AngularJS is used for a user UI of the application like alert messages and calendar, time display.

Srinivasan Sundaramoorthy

Team-4 NJIT Official Guide for iOS

Modules Worked

What's that building module

Tools Used

XCode	:	iOS development
TOAD for MySQL	:	DDL & DML activities
Unity	:	Augmented Reality
Sublime Text	:	PHP scripting
Adobe Photoshop		Design
GIT	:	Version Control

Cloud Sources

AFS	:	PHP script
AWS AMAZON – RDS	:	MYSQL instance

What's That Building module:

Functionalities

- User can identify the buildings at NJIT
- User can use his/her device's camera to identify the building.
- User can point at a building to probe information about the building.

The module was developed in Unity3D and built for iOS. Vuforia libraries and prefabs was used for the building recognition methods. An AR camera prefab was imported into the project which later acts as the device camera when deployed to an actual device. The Image target prefab was used to set the target images. The image target prefabs was placed in the view of the AR camera. Used four image targets with 2 different buildings images at different angles. A device database in the Vuforia developer library is initially created in which the image target files are processed. The JPEG files of the buildings is uploaded to the developer library in which is processed and converted to image target datasets for Unity3D. All the datasets are downloaded and used in the Unity3D. An App key is generated in the Vuforia developer portal which is copied into the app key field in Unity3D. The image targets are placed side by side in the view of the AR camera. Custom trackable event handlers are written individually for each image target which displays text on tracking found displaying the building name. The custom trackable event C# scripts are wired to each other. The build is exported for iOS and the Objective-C scripts and libraries are imported into the existing native iOS project.

RaviRagul Balakrishnan

Team-4 NJIT Official Guide for iOS

Modules Worked

Shuttle Stop Module

Broadcast Module – iOS

Tools Used

xCode	:	iOS development
TOAD for MySQL	:	DDL & DML activities
Notepad++	:	PHP scripting
GIT	:	Version Control
Rutgers API	:	For Shuttle Map coordinates, Bus and Shuttle Stop location

Cloud Sources

AFS	:	PHP script hosting
AWS AMAZON – RDS	:	MYSQL instance hosting

Shuttle Stop Module:

Functionalities

- Drawing the shuttle route
- Plotting bus stops
- Updating location of bus and predictions
- Direction from user location to bus stop

In the shuttle stop module, I have used the Rutgers API which is open source for plotting the map. Initially I created a view controller with a UIPickerView for selecting different shuttle routes. After selecting the route the viewcontroller is segued to a map kit which contains the map of the selected route. The map is drawn using the list of latitudes and longitudes obtained from Rutgers API website as xml page. Then using NSXMLParser delegate the xml attributes are parsed and the latitude and longitude belonging to the selected route is stored in a double array. Then the annotation of each coordinate is create using MKAnnotation delegate by passing array of objects of Station class(used to initialize the lat, long and titles of annotation). Then the annotation is passed to renderForOverlay method of MKPolyLine delegate. Here the map line between the coordinates is drawn using MKPolyLineRenderer object. After the route is drawn, I plotted the shuttle routes by getting the route data from Rutgers website as xml and using the same method to parse and create annotation using Station class.

I have plotted the shuttle stop by passing the annotation to viewForAnnotation method which plots the annotation and adds the title (the information that is shown when the annotation pin is clicked in the map). The bus location is updated as the bus moves in the Rutgers website. So to continuously update the map I created a NSTimer object to constantly call an onTick function which updates the location of the bus in the map. The onTick function is called for every five seconds. Here the annotation of bus should be deleted and reprinted to show the bus moving. The prediction of the each bus stop also gets updated in the website as the bus moves. So the getPrediction class gets the prediction of each stop and create an NSDictionary of the stops and predictions in minutes. The predictions is passed as subtitle while creating the annotation of the bus stop. So when we click on the bus stop pin the title of the stop displays and the prediction in minutes is shown below the title. The current location of user is plotted in the map using CLLocationManager delegate. There is a get direction button in the navigation bar, when we click on the button the controller gets the location of user and the selected stop and plots the direction using MKPolyLine.

Broadcast Module - iOS

Functionalities

- List the Broadcasted event information
- Add events to calendar

In the Broadcast module i have listed the events by pulling the information from Amazon AWS database using php script in NJIT AFS web server. The information is passed as json and using NSJSONSerealizaion to copy the json data to array. Using the array list I populated the UITableViewController to list the events. When clicking on an event it is segued to a view controller which shows all the information about the event like the venue, time and who posted the event. To add the event to calendar I have used the EKEventStore delegate. I took the time, date and title information from database to specify the date, time and title to be stored as calendar event respectively. To post and event we can click on Broadcast button which opens a link in web browser and we can input the event information and post the data.

Arif Mukhthar Umarkayam Shakilabanu

UCID: au56
Team: Team-4

Module Worked on

Buy and Sell books – NJIT Professors details

Tools Used

xCode	:	iOS development
TOAD for MySQL	:	DDL & DML activities
Notepad++	:	PHP scripting
GIT	:	Version Control

Cloud Sources

AFS	:	PHP script hosting
AWS AMAZON – RDS		:

NJIT Buy and Sell books

Functionalities

View books that are put up for sale by other users
Contact a seller via email
Post details about a book that is to be sold

Work Done

The Buy and Sell books Module is based on a table structure. This module helps the students to post details about the books they want to sell and also allows them to view books that are put up for sale by other users and also get in touch with them by email.

To post a book's detail to the buy books page the professor's details I've created two views,

1. A table structure to list the books details and selecting a book would take the user to a view which enables them to send an email to the user expressing his interest to buy the book.
2. The second view is a seller form. This is where has the users who wish to sell their books can post details about their book.

I have created a table in the database with the book's details that are posted by the sellers. I have written two PHP scripts, out of one which fetches the details from the DB and returns it as JSON to the Buy and Sell books Module. The JSON data is then parsed and the array is spilt into separate details and then is populated in the buy books view controller.

I have also written another view controller which is a form from where the sellers can post details about their books. There a another PHP script written here which parses the details in this view controller and inserts the data into buy and sell books database.

M0 Contribution Log:

As a TEAM

We as a Team have worked on various activities like Requirement Analysis, Feasibility Study, Coding, Debugging, Integration Testing, Peer Testing, Feedback analysis etc.

Requirement Analysis

Framed the Software requirement specifications – gathering inputs, ideas and suggestions from the group.

Feasibility Study

Check the complexity of the Requirement and determine if they can be developed in real time scenario with the available resources and time.

Coding

Forming regular sessions so that any feature which is generic is added to others as well. Took up coding sessions to facilitate the development stage.

Debugging

Helping peers to overcome issues and errors whenever there is a bottle neck situation. Continuous brainstorming for finding the solution.

Integration Testing

Individual Modules are taken up by each person but integrated to a common storyboard. Integration testing done in groups so that the work of one individual is not hindered by the other.

Peer Testing

There is always a room for improvement and we made sure that by peer testing and provided the knowledge about the module at the same time to our peers. This avoids dependency of a person to any module at any situation.

Feedback Analysis

After rigorous development the application was put forth among students for getting their views on the application and the feedbacks where noted and necessary changes were made according to the inputs provided.

Documentation

As a group we are working on the document template and including the parts which were managed by each individual.

As an INDIVIDUAL

Module Split-up

	Percentage of Contribution
Tushar Swaminathan NJIT Vector Newspaper Article Feed NJIT Contact Information Documentation - Data Flow Diagrams & Architecture Pattern Diagrams	**25%**
Srinivasan Sundaramoorthy Lost and Found UI & UX designing (For the entire application) Testing & Debugging	**25%**
Ravi Ragul Balakrishnan NJIT Appointment Scheduling NJIT Academic Calendar Documentation – Template preparation & Documenting Generic features	**25%**
Arif Mukhthar Umarkayam Shakilabanu Know Your Professor To-Do (Under Development – Extra Feature) Initial Software setup in Apple environment	**25%**

Tushar Swaminathan

UCID: ts336
Team: Team 4

Modules Worked

NJIT Vector Newspaper – LIVE Article Feed
Contact Information of NJIT

Tools Used

xCode	:	iOS development
Toad for MySQL	:	DDL & DML activities
Eclipse	:	Article Extraction from HTML Tags
Task Scheduler	:	Executing the JAR file (to provide LIVE feed of data)
Notepad++	:	PHP scripting
GIT	:	Version Control

Cloud Sources

AFS	:	PHP script hosting
AWS AMAZON – RDS	:	MYSQL instance hosting

NJIT Vector Newspaper

Functionalities

LIVE (Real-time) data feed
Display Article Images
Search Functionality
Share Article feature through email, watsapp etc…

Work Done

The Vector module is a basic table view structure. The Source of the information is present in the form of website, to retrieve the information I created a java file which reads the URL information (includes HTML tags). After I get the URL structure I parsed the content with JSOUP and HTML Editor Kit. I retrieved the Article Title, Article URL and Date of Publishing etc. After extraction I created a connection to the database and dumped the data into the respective tables. For real time feed I converted the java code into a Runnable JAR file and then created a Task (using the Task Scheduler), which runs the JAR file at specified time interval. Thus whenever there is a new update it is automatically inserted into our database and a LIVE feed is provided.

To view the vector information, I created 2 views one containing the Article Title and other the Article information. The Article title are fetched using a call to the PHP script which interacts with the DB instance and returns JSON. I then converted the JSON into an Array format and merged it with the view, which displays the data.

In the second view, I created another PHP script which fetches the Article information based on the selected article title. I passed the article id to the PHP script which in turn interacts with the DB and returns the JSON. I then convert the JSON into Array format and merged it with the view, which displays the data. The Article Images are fetched the same way as the other data. To display the image I had created an image view which displays the image based on the image URL.

Search Functionality is a feature where a User can search for the articles he wants to view. I have created an internal application filtering method, so data is processed at a faster rate and details appear as user enters new characters. Since there is no database call for each search it is efficient.

Share Feature is an extra feature (which was not mentioned in the requirements). Here the user can share an article which he / she likes via e-mail or any other social media he is logged into like watsapp, facebook, twitter etc. I have created a variable which passes the required data into the Share functionality and then invoke the required apps respectively.

NJIT Contact Information

Functionalities

Search Functionality
Scope based search – Department and Person
Click to Call

Work Done

The Contact module is used to display the contact information of all the administration and useful contacts present in NJIT. I have saved the contact information into the database via Toad. Later in the view I had created a table structure to display the data. In the model I had created a call to the PHP script containing the MYSQL script. After execution of the PHP the data is returned in the form of JSON. I then convert the JSON into Array and bind the data into the table structure.

Search Functionality is an added feature where a User can search for the articles he wants to view. I have created an internal application filtering method, so data is processed at a faster rate and details appear as user enters new characters. Since there is no database call for each search it is efficient.

Scope functionality is added for the user to switch between the searches criteria. The Scope is added to the search bar and depending upon the value of the selected scope I have filtered the sequence.

Call Functionality allows the user to call a specific number from the contact list provided by just clicking on the cell. I have created a function call to the applications call app which passes the parameter of the number which is currently selected by the user.

Srinivasan Sundaramoorthy

UCID: ss2773
Team: Team 4

Team-4 NJIT Official Guide for iOS

Modules Worked
Lost and Found Module

UX/UI Design(aesthetics & functionality)

Tools Used

XCode	:	iOS development
TOAD for MySQL	:	DDL & DML activities
Sublime Text	:	PHP scripting
Adobe Photoshop	:	Design
GIT	:	Version Control

Cloud Sources

AFS	:	PHP script
AWS AMAZON – RDS	:	MYSQL instance

Lost and Found Module:
Functionalities

User can post the items they found in college

People who lost items can check for the list of found items and retrieve it from the user who has posted.

A news feed like structure of lost items at NJIT.

A table view controller was set up to display the lost items. A UIImageView for displaying the image. A title label to show the name of the item lost. A text View for displaying a short description of the lost item. A table was set up in the MySQL instance for this module. Columns for User Name, User Email, Item Title and Item Description. The Lost item table loads data from this table at run time. 'I Lost That Button' was put on each cell of the table. Clicking on that button brings up a Mail Composer if set up and 'to:' field in the Mail composer are set to the

User who posted the Lost item. A Post Button is at the top right end of the table. On Clicking the Post button takes to the View Controller which handles the input form for the Lost Item. This form consists of User Name, User Email, Item Name and Item description fields. A save button to save the details to the database.

A PHP script was written to handle the form data from the Lost and found Module and hosted in my AFS directory. The Post form from the application parses the data in the body of the HTTP request. The PHP script takes the parameters from the body of the request using the

necessary keys. Connection is set up between the amazon rds and the php engine. On connection success, using mysqli_query the query is executed to insert into the table. Likewise a PHP script to handle the incoming GET requests executes the fetch query from the database and sends the result as JSON to the module.

User Interface Design:

Studied numerous applications in the App store for design inspiration. The native red color from the NJIT logo was retained. Designed a rounded rectangular in-app icons for every module in Adobe Photoshop CS6. No stock images or stock vectors were used for the design of the in app icons. Every in-app icon was made from scratch as a custom vector designed in all 1x, 2x, 3x resolutions. In Total twelve in-app icons were designed for the app. Minimalism was the prime focus. These icons were exported to XCode and added to the assets library. The placement of various buttons and navigation items was made intuitive and not make the user search for a specific button. The text colour for every module was made white and the background for every module was made red and all the modules were made to be coherent in look and feel.

RaviRagul Balakrishnan

UCID: rb454
Team: Team-4

Modules Worked

Appointment Scheduling module(With Professors and Athletic Center)
Academic Calendar module

Tools Used

xCode	:	iOS development
TOAD for MySQL	:	DDL & DML activities
Notepad++	:	PHP scripting
GIT	:	Version Control

Cloud Sources

AFS	:	PHP script hosting
AWS AMAZON – RDS	:	MYSQL instance hosting

Appointment Scheduling module:

Functionalities

Set appointment with Professor or Athletic Center View Professors by department

In the appointment scheduling module I created a list containing Athletic Center and Professor. By clicking Athletic Center the list of games is fetched from the Amazon AWS web server database (db) using PHP script, which is hosted in NJIT AFS web server. When clicking on Professor a list of department is fetched from db. After clicking any of the department, using the httpPOST request to PHP script the selected department is send to the db and the list of professors in that department is fetched and shown in the table view. After selecting a professor's name or game name a date selection page is displayed, where user can select the date using datepicker (user cannot set a date less than today's date) and move to selecting time for setting the appointment. Here there are two timepicker views where user can select from time and to time. By default, from time is set to current time and to time is set to current time plus one hour. The date, from/to time and Professor/Game which the user selected is displayed. By clicking the set appointment button mail app is opened with new mail page. The 'To address', subject and body of the mail is already typed. The user can edit the mail or directly click the send button to send the appointment scheduling mail. For the mail to work the user should have a mail app installed and logged into that mail app. I have used the Toad for MySQL for inserting and modifying the data in the Amazon AWS db.

Academic Calendar module:

Functionalities

Search Functionality

In the Academic Calendar module a list of semesters is displayed in a table viewcontoller. By clicking on a semester, the academic calendar of the selected semester is fetched from database using the httpPOST request to PHP script and displayed in a table view. The list includes the date, day and the events of that day. I have also added the search functionality for the calendar where user can search for the information in the selected semester.

Arif Mukhthar Umarkayam Shakilabanu

UCID: au56
Team: Team-4

Modules Worked on

Know Your Professor – NJIT Professors details
To Do – Check list creation tool (Extra Feature Under Development)

Tools Used

xCode	:	iOS development
TOAD for MySQL	:	DDL & DML activities
Notepad++	:	PHP scripting
GIT	:	Version Control

Cloud Sources

AFS	:	PHP script hosting
AWS AMAZON – RDS	:	Database Hosting (MYSQL)

Know Your Professor

Functionalities

View Professor's details and contact
Search Functionality

Work Done

The Know Your Professor Module is based on a table structure. This module helps the student view a professor's details. I have also implemented a search functionality in the module which helps the user to search for a professor by his/her name.

To view the professor's details I've created two views,

1. A table structure to list the professor's name and click on a name would take the user to the second view.
2. The second view has the details of the professors along with the picture.

I have created a table in the database with the professor's details. I have written a PHP script which fetches the details from the DB and returns it as JSON to the "Know Your Professor" Module. The JSON data is then parsed and the array is spilt into separate details and then is populated in the second view controller.

I have also written a search function using Swift's search display controller class. The search function filters the data populated in the first view controller (without) in real time and presents the user with the data they are looking for with speed. I've also written a swift function to retrieve professor's image from people.njit.edu by using the URL stored in the database and displaying it in the professor's details screen in the second view controller.

Also I have added a search functionality for the module, which enables the users to search for professors based on their name or department.

To Do List (Under Development – Extra Feature)

Functionalities

View & Enter To Do List
Check After Completion

Work Done

The To Do module uses a local sqlite database to store values. Enables the user to create and delete a task. Put a check mark against the task after it's done.