`

Study Buddy

CS 307 - Software Engineering

Design Document

Team 18: Bor Shiuan Chen, Abhijay Gupta, Joseph Miller, Nathan Scott, Zeb Wezensky

Table of Contents

Purpose

You missed class for some reason and the professor doesn't post the notes on blackboard. At this point you have no way of getting what the professor was talking about, and you know don't anyone in the class. Our application helps you connect with other students from the same class. Even beyond that, it can be a way for you to find others with similar hobbies. For example, when you need a spotter in the gym, or if you just want to play ping pong with someone.

For many students, meeting other classmates can be a challenge. Meeting someone who would like to study together can also be difficult. Our application aims to make it easier for students to connect with like-minded, class oriented peers at Purdue University. At this point in time, there are no similar applications that help students meet fellow students that we are aware of. While sites and applications exist for the purpose of meeting new people, there are none that place an emphasis on meeting someone to study with.

The following helps show what we hope to accomplish with our application:

- Create an app that Purdue students can use to meet fellow students in their classes for the purpose of studying together.
    - Using a matchmaking system by classes.
        - Users enter a list of their classes/interests, and the app matches them with others with the same classes or interests.
        - Users are identified by their Purdue email.
        - Users can choose to meet with a group, i.e. match with multiple people in the same class with the same availability.
    - Matches people in classes for the purpose of studying together.
        - Also able to match people who have similar interests, i.e. going to the gym, ping pong buddies, etc.
    - Includes a chat feature which allows students to talk to each other.

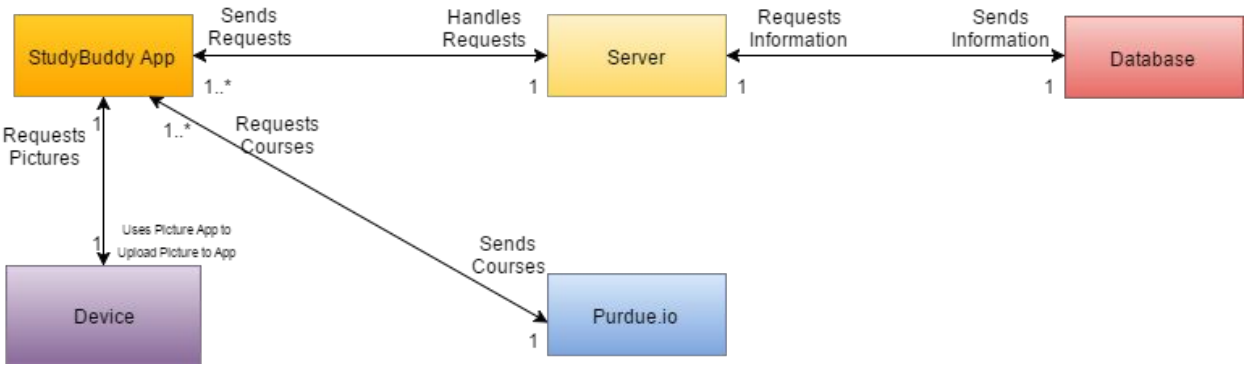
Design Outline

High Level Overview of System

The system is set up in such a way that there is one central **server** which each **app** sends requests to. The system is as light as possible on the app, attempting to do as much work as possible on the server in order to make the app lighter and faster.

The server uses a **database** of all users which it can send information to in order to update any user's profile, including their picture, courses, inputted bio, and contacts/chat list. The database also includes all interests that users input, which users of the app can add to their profile bio in the same way they add courses.

The **device** is used when a user wants to upload a picture of themselves to add to their profile bio. The user edits their picture and the default picture managing app (e.g. Gallery, Google Photos, etc.) is used to select the picture they want to upload. **Purdue.io** (an external API) acts like a seperate database of courses. When searching courses, the app sends out a request to Purdue.io to get all courses and filters through them to see which classes match the input the user has already started typing (e.g. if the user has typed "c", the app will filter the course list to only show courses that begin with "c", and if the user types "s" after "c" to make "cs", then the app refreshes the list to show only courses that start with "cs").



Design Issues

1. *Authentication*
    a. Users create a new account which we store in a database
    b. Users uses a pre-existing authentication method

Using a pre-existing not only improves security, but also reduces the amount of steps for the users. For our application, we decided to use the Purdue authentication method. Rather than using other methods of confirming whether users are from Purdue University, by integrating Purdue Web Authentication it confirms that as well.

2. *Server Implementation*
    a. Writing an entire server with custom protocols
    b. Using Django as a Web API

Due to time constraints, for us to put out a working server, with good security, we decided to use an existing framework available to us. The client will contact the Django server with an API. Then Django will call the python script for the API and return with an output in a readable format (ex. JSON)

3. *Messaging*
    a. User will be given the other user's contact information. Then they can use other applications to contact each other
    b. **Integrate a messaging system**

It would be really inconvenient for users to have to leave our application to contact someone. Integrating a messaging system directly into the chat system will make it easier for users to contact others. At the same time, we can reuse the system to make it into a news feed.

4. *Categories*
    a. Having default categories for each class and interest
    b. **Pulling class from Purdue.io and letting users to create other categories**

Classes are always changing every semester which means our categories will have to be dynamic. Purdue.io is a site that allows users to search up classes that are currently being offered and statistics about the class. At the same time, it is impossible for us to predict the user's interest. It will be easier if we allow the user to create their own interest page.

5. *Privacy*

a. User will be anonymous to others unless both added each other as a contact

b. A public profile picture will be accessible to others.

Rather than just having a name for the users, letting users customize their own profile will make them unique. This also allows users to have a preview of other people there were matched with. With this we have also given user the ability to block others if it is needed.

6. *Database*

a. GadFly
b. mSQL
c. PostgreSQL
d. Microsoft SQL Server 2000
e. Informix
f. Interbase
g. Oracle
h. Sybase
i. MySQL

Python offers a wide range of database methods but we chose to use MySQL. MySQL is easy to use and popular among other programmers.

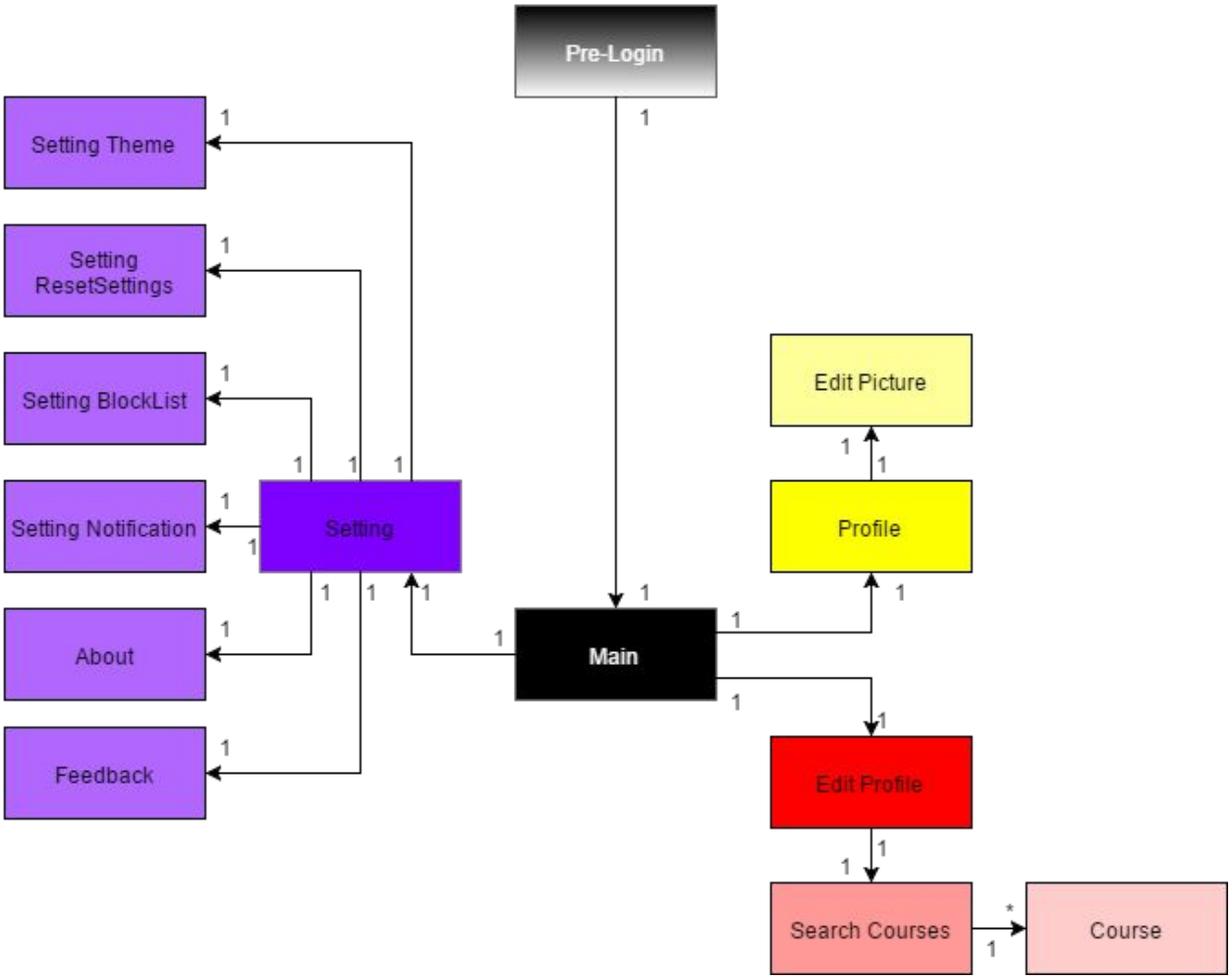
7. *User Groups*

a. Normal Users, Professors, and Administrators

b. All in one group

We feel that there shouldn’t be any distinctions between the users in the backend. Users shouldn’t have the ability to do any administrative controls. We will allow administrators to access our backend directly if any drastic measures are required. Users will also be able to block other users if they feel necessary.

Design Details



Class Descriptions

Activities - Each activity in our application will require its' own class. The following are descriptions of each.

Main	This is the main page the user will see when they open the application. All content is accessible from this page, either directly or indirectly. Main features will include a class news feed, sliders and buttons to quickly pull up menus and chat.
Profile	A user's profile page, visible to other users. Will include all user provided information: picture, basic user information, bio, courses, and interests. If it is the user's own profile page, buttons next to each section of information will allow the user to edit their information, via text boxes and the add courses or edit picture pages. For matchmaking purposes, the database will contain a list of user added interests.
Edit Picture	Accessible from the user's own profile page. Will allow the user to upload or take a picture, and then save the information to their profile.
Course	A course page serves as a main page for a given course. Accessible from various course links throughout the application. Will contain course news feed which users can add to, and the ability to quickly add this course if not already in the user's list of courses.
Search Courses	Will quickly let users find any given course. Accessible from the edit profile page. The page itself will contain only a search bar, which upon user entering text will automatically list a drop down menu of classes. Example: if the user types "CS", the drop down menu will begin displaying all CS courses. Will need to have access to a list of all classes from the Purdue.io.

Chat	Chat page that will allow users to communicate with other users. Like many chat functions, will contain the name of the other user at the top, and messages will display on either the left or right depending on who sent the message.
Contacts	Will contain a list of other users that the current user has either communicated with, bookmarked, or both.
Setting	Accessible from menu in main. Will contain basic settings the user can modify; see below.
Setting Theme	Accessible from settings page. Will let user change basic color themes that affect the appearance of the application.
Setting Notification	Accessible from settings page. Will let the user change phone notifications, from people, individual classes, or all of the above.
Setting BlockList	Accessible from settings page. A list of the current user's blocked users, with the ability to unblock any given user.
Setting ResetSettings	Accessible from settings page. Will allow the user to clear various types of application data, including chat history, current classes, or all of the above.
Feedback	Accessible from settings page. Will allow the user to give feedback on the application, which will be sent to the development team.
About	Accessible from settings page. Will contain basic information about the application, such as version number and creators.
Pre-Login	Only accessible from the first use of the application or if the user logs out at any point. A basic login screen where users can input their Purdue credentials to enter their account.

Client Database Interaction

All classes will be able to access and or modify application data. However, some will require interaction with a server.

Main	Will require server communication to access data pertaining to a user's course feed.
Profile	Will communicate with server to provide information about the user, and to receive information about other users.
Edit Picture	Will need to be able to modify the user's profile, updating server information.
Course	The application will need to receive information to keep the course news feed up to date, in addition to sending information if the user posts to the course news feed.
Search Courses	Will need to access a list of all current Purdue courses, and be able to access their course pages when necessary.
Chat	Will require server communication to send and receive messages to and from other users.
Contacts	Will contain a list of other users that the current user has either communicated with, bookmarked, or both.
Settings	Most settings will only require the modification of the application's data, but the block list and resetsettings will need to communicate with the server to change various data if necessary.
Pre-Login	Will need to access the Purdue Career Account services to authenticate user credentials.

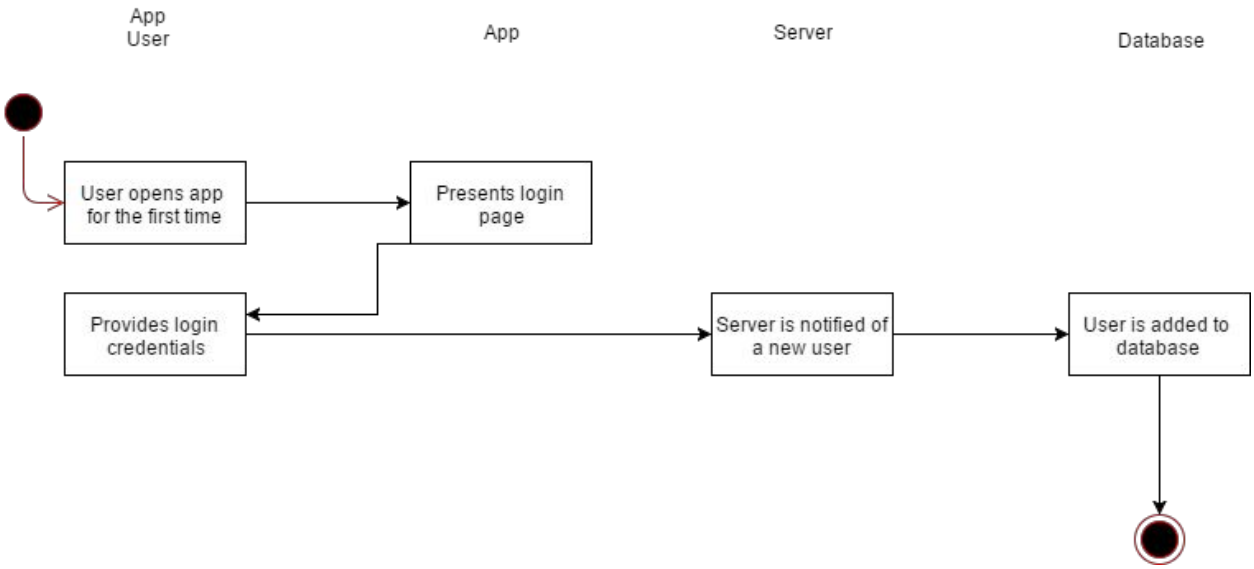
Server

The StudyBuddy server will utilize the open-source Django web framework. Django makes use of the model-view-controller architectural pattern, where the “model” component keeps the view up to date, the “view” component provides visual information to the user, and the “controller” component changes the model with information given from the users.

System Interactions

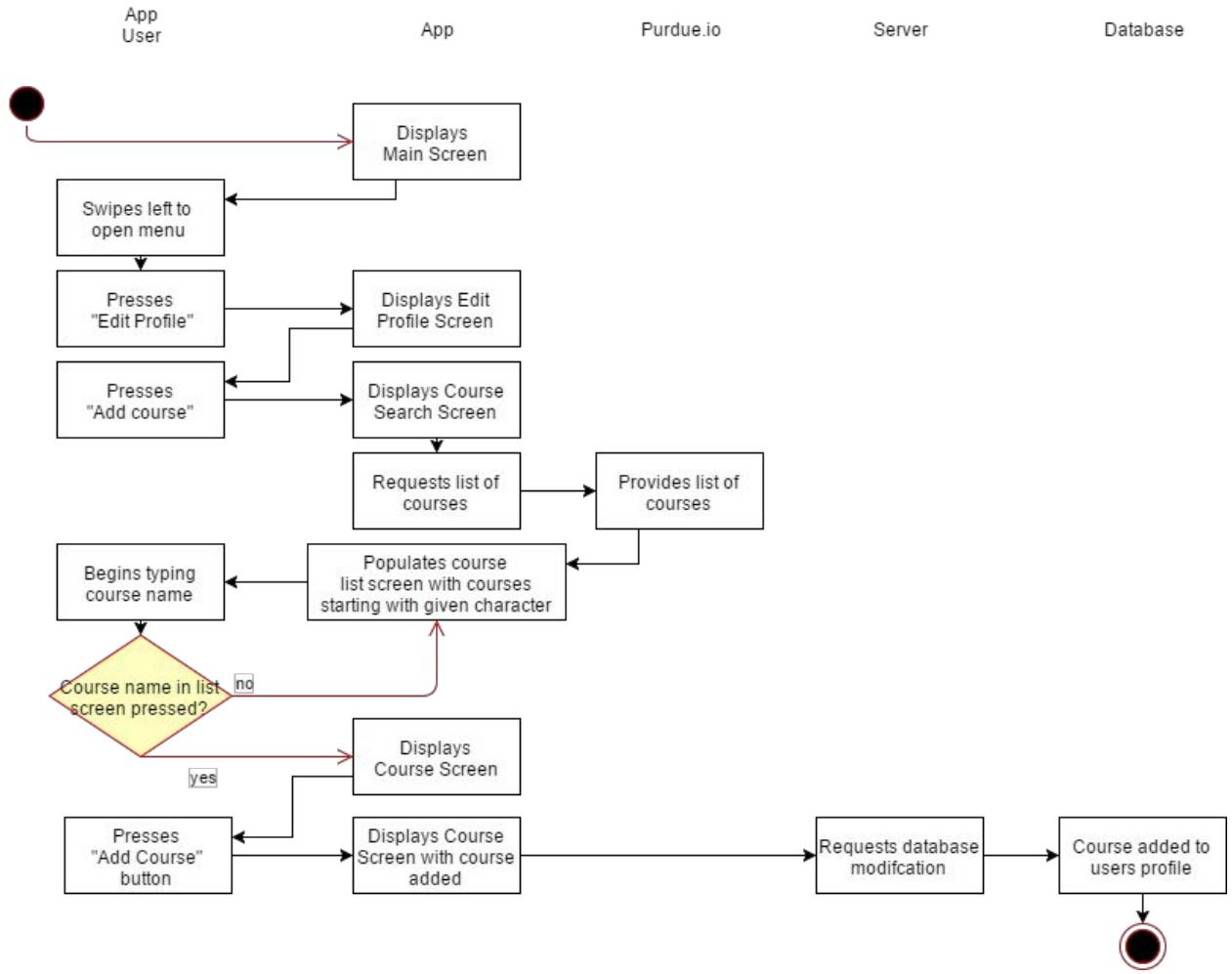
I. Initial App Setup

The app presents a login page, where the user presents his/her credentials, and the app presents the main page for this user, after adding the user to the database.



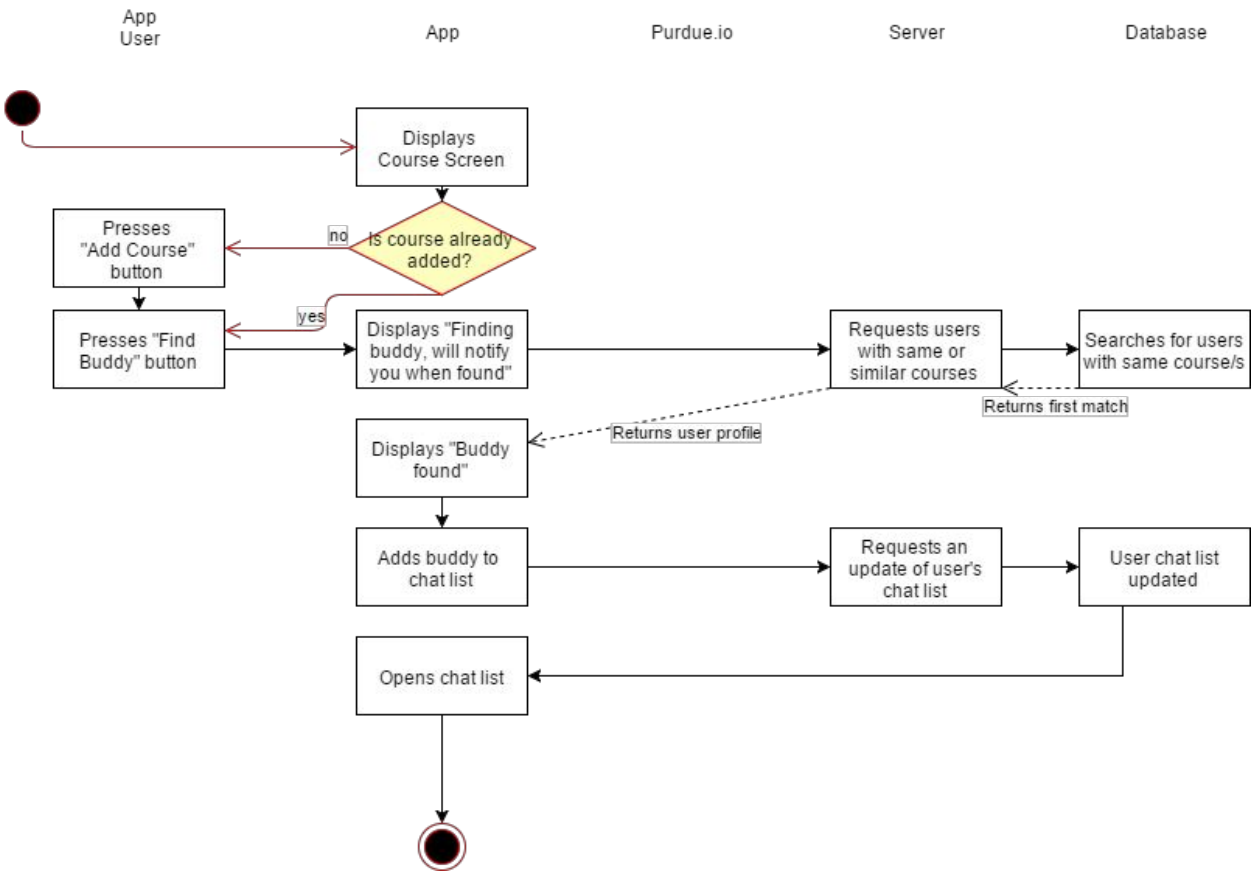
II. Add Courses

The user navigates from the main page to the edit profile page, then from the edit profile page the user clicks on the add course button, where it takes the user to the course search screen. The app then requests the list of courses from Purdue.io, where it populates the course search screen with the courses starting with what the user has inputted, listed in alphabetical order (if the user hasn't inputted anything, the entire course list is displayed). When the user selects a course from the list at any point, the app displays that course's screen, where there will be an add course button at the bottom. When the user presses that button, the app will send a request to the server to change the user's "courses" database entry.



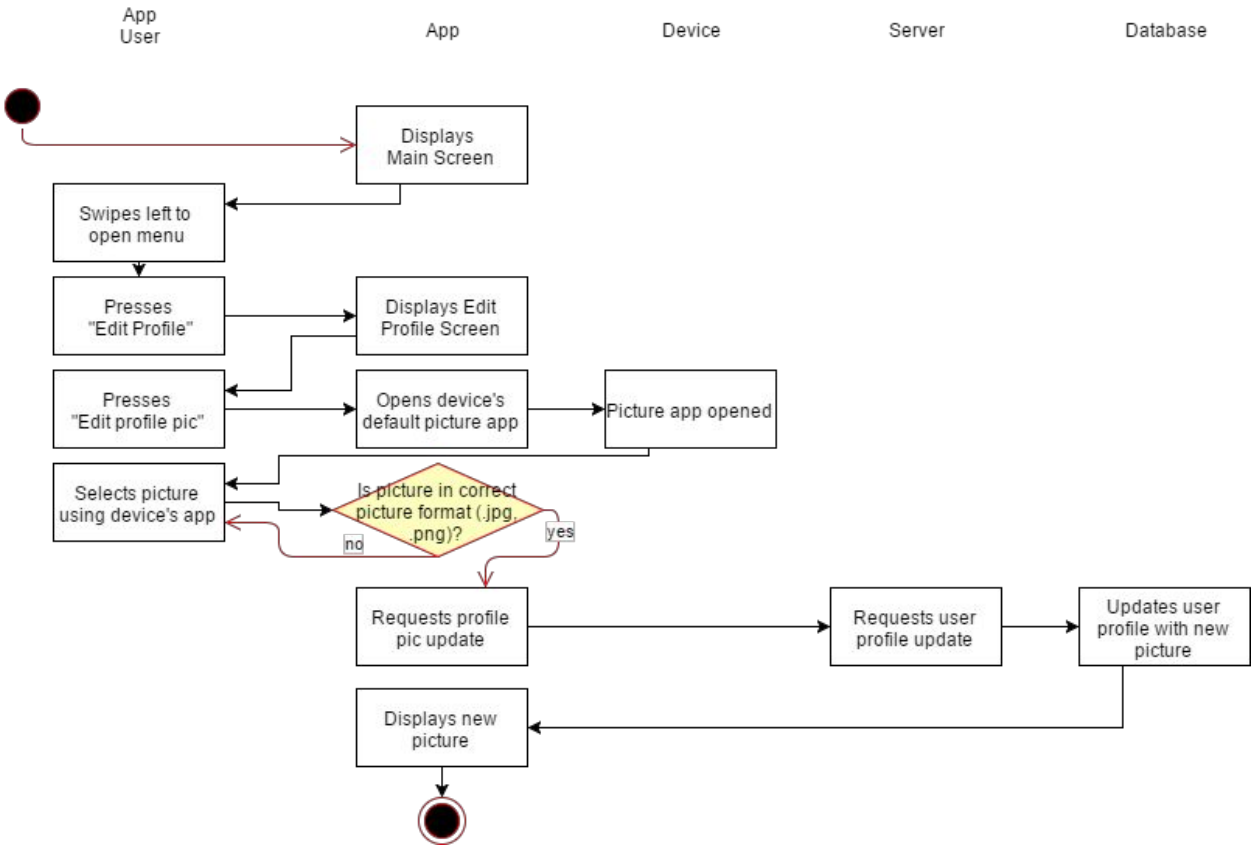
III. Find Buddies

On the course screen, the user will add the course if it's not already added. Once added, the Add Course button becomes the Find Buddy button. The system then displays the notification that it is finding you a buddy, and will notify you once one has been found. It then requests from the server a user with the same courses, which searches the database for users with the same courses and returns the first match. It returns the user profile to the app, which displays "Buddy Found" and adds the buddy to the user's chat list/contacts. The server then updates the user's contacts list in the database and opens the chat list with the new buddy added.



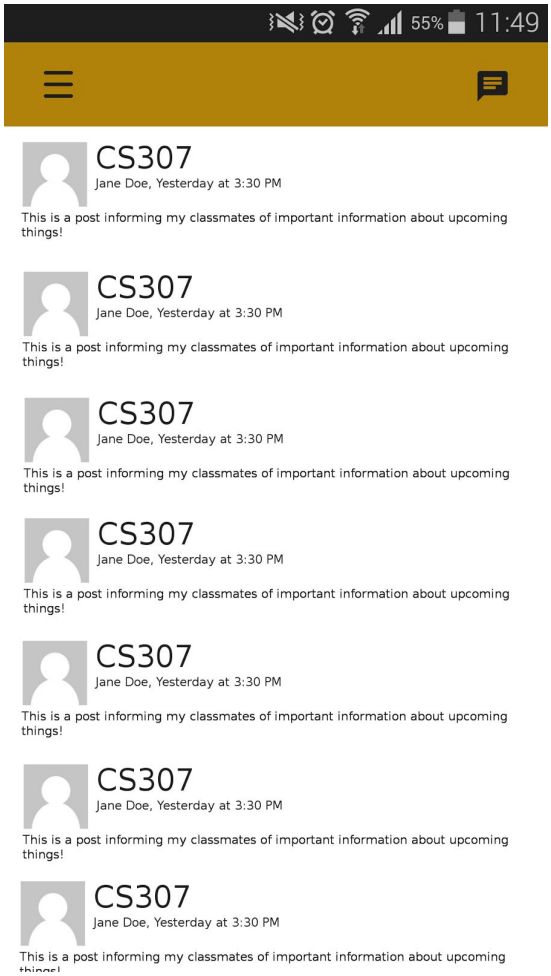
IV. Edit Profile Picture

The user navigates to the “Edit profile” page of the app, which allows the user to edit any aspect of their profile such as their bio, courses, and picture. Once the Edit Profile Picture button is pressed, the device then opens the device’s default picture app (or if there are multiple apps and none is set to default, the device performs normal Android behavior and asks which app to use), from which the user will select the picture. If the picture is in correct format, the app will send a request to the server and then to the database for an update of the user’s profile picture. After this is completed, the new picture is displayed in lieu of the former one.

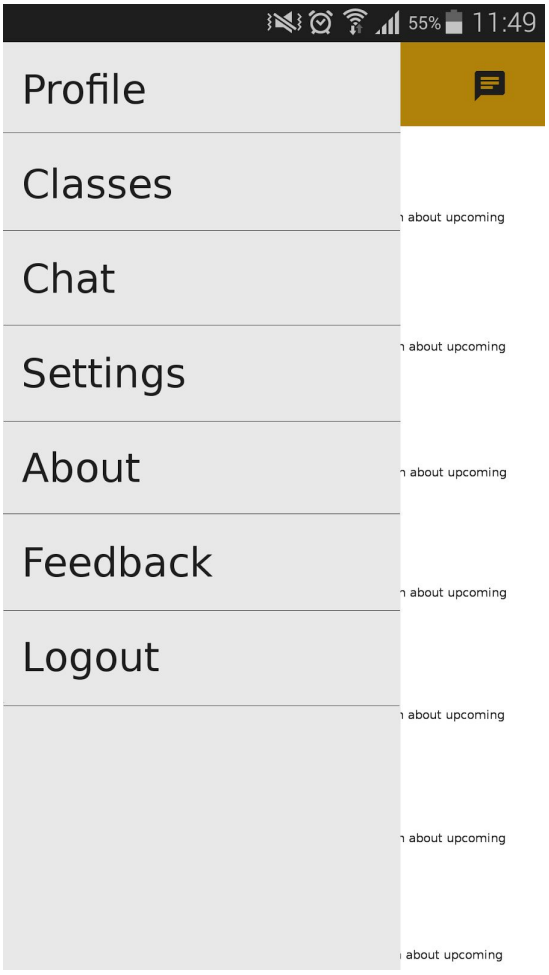


UI Mockup

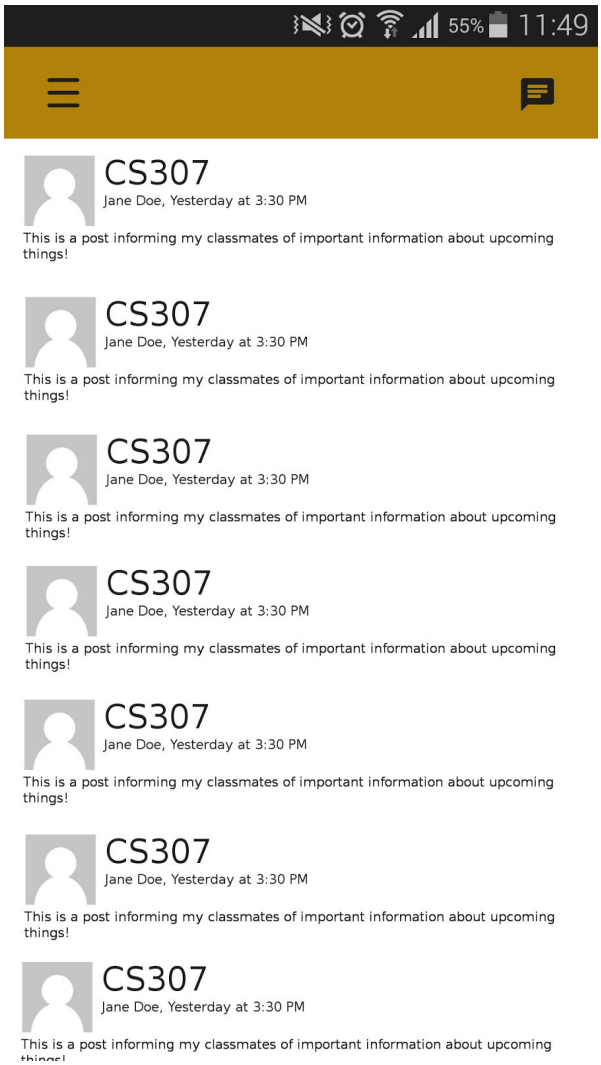
Main Menu



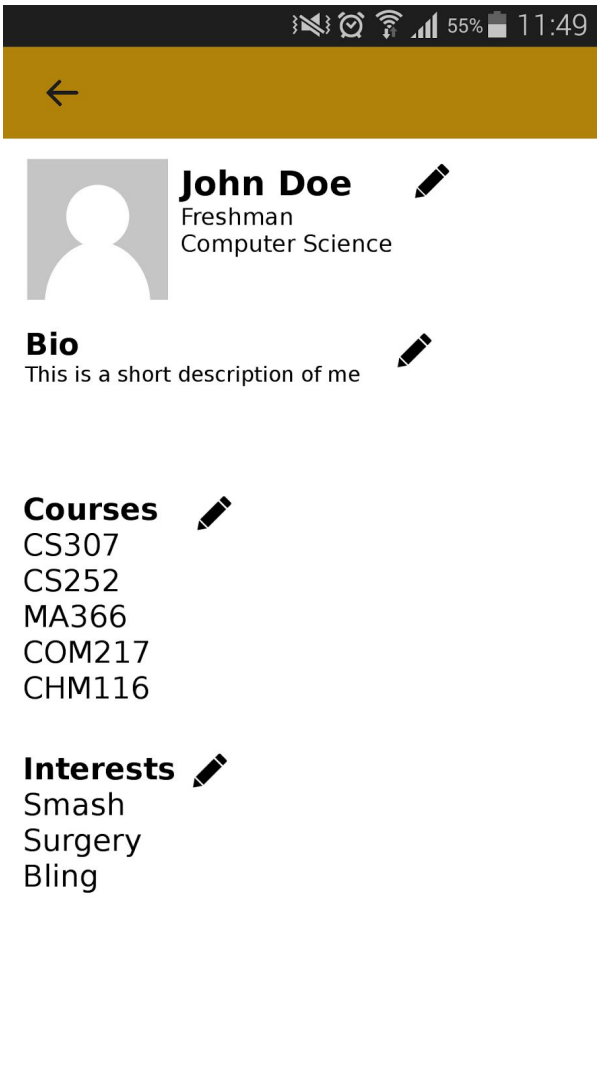
Main Menu Slide



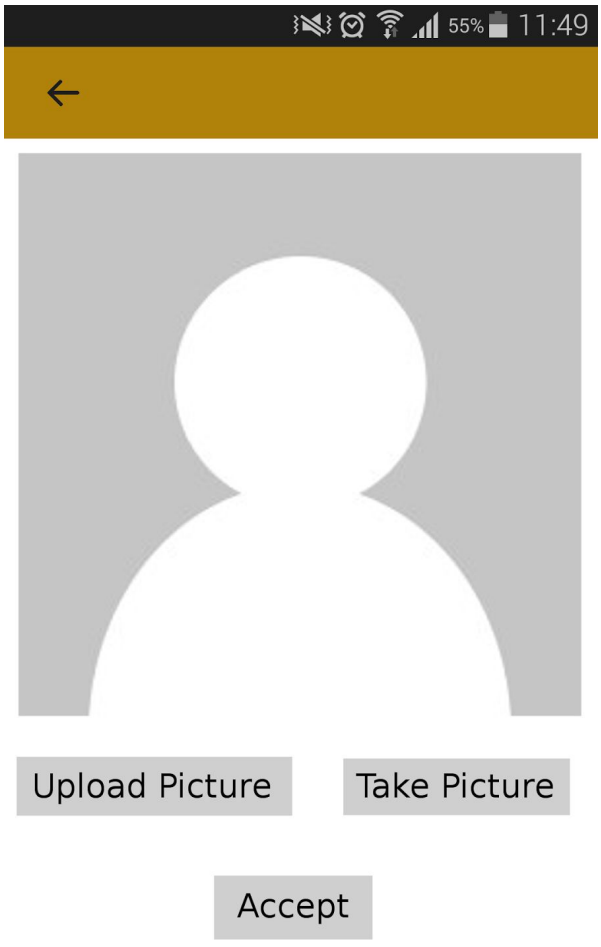
Main Menu Chat



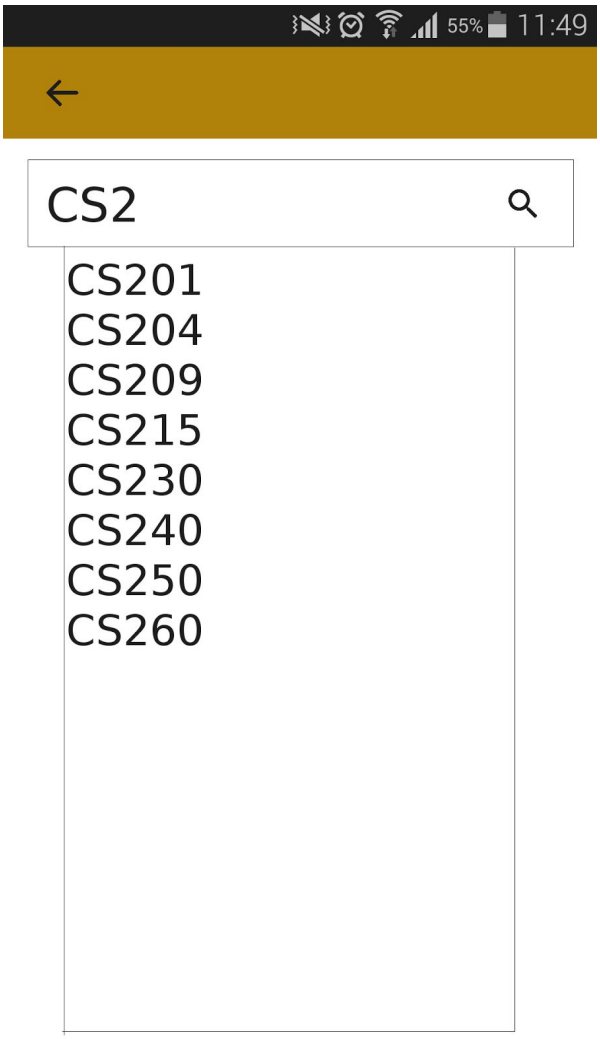
Profile



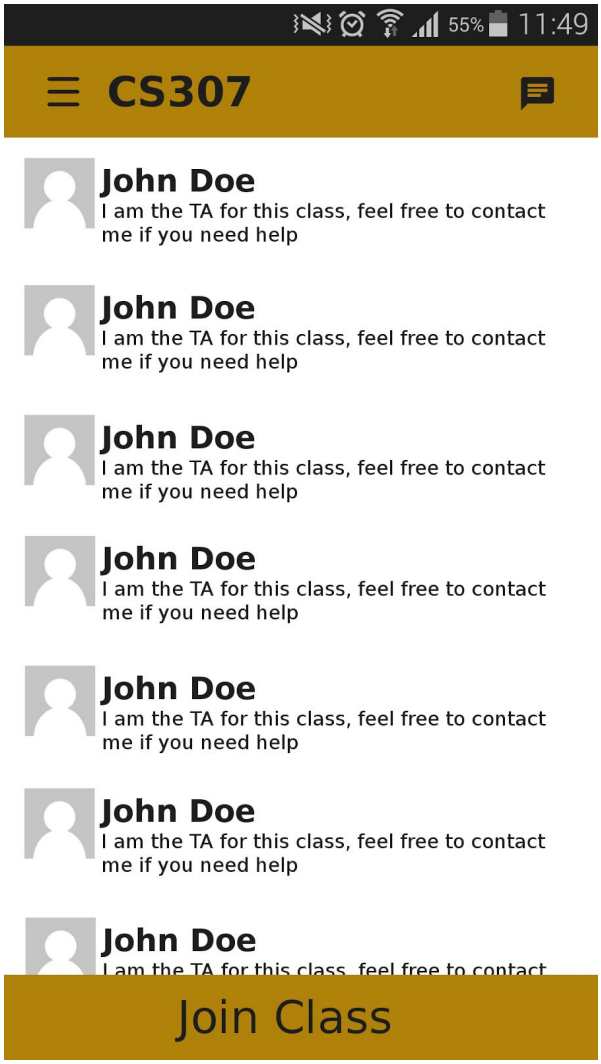
Edit Picture



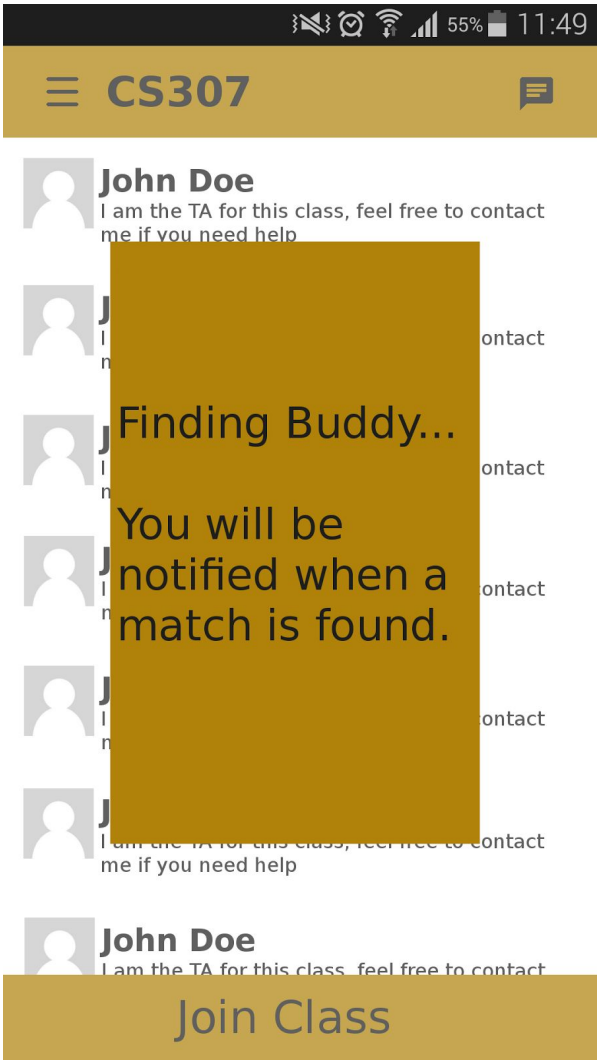
Course Search



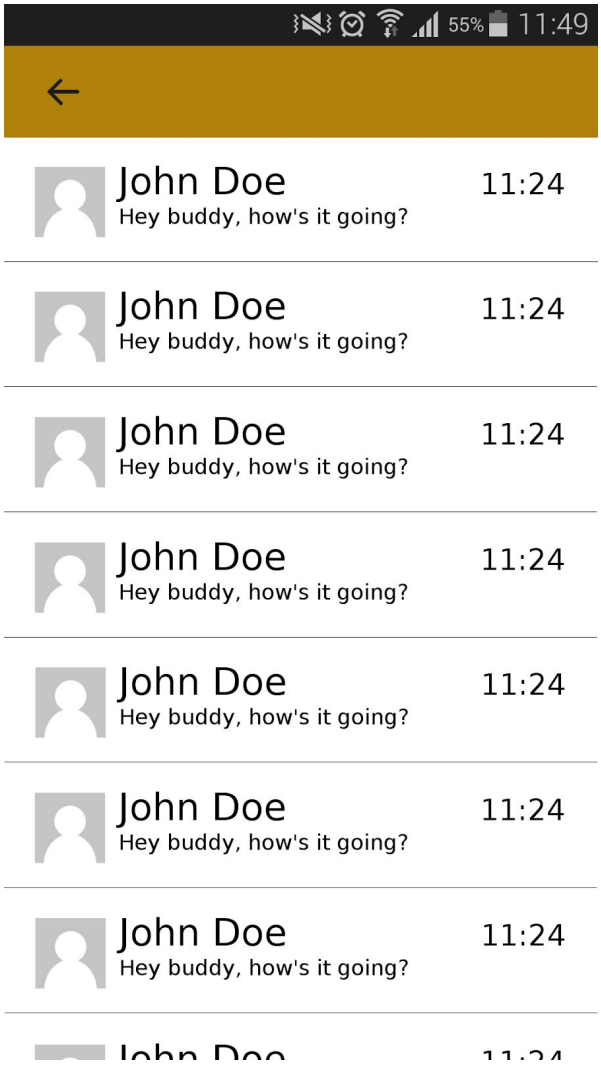
Class Page



Buddy Found



Chat Page



Chat Session