Moods – App for Champlain Students

By

Amir Osman

Étienne Bérubé

William Perron-Lafleur

Presented to

Mr. Amin Ranj Bar

420-204-RE, Integrative Project in Computer Science and Mathematics, section 00871

Wednesday, February 26th, 2017

Champlain College Saint-Lambert

Table of Contents

Introduction

As our final project we want to create a social network on android phones exclusively dedicated for Champlain students. Communication has always been something of paramount importance in the everyday life. During our college experience, we have noticed a lack of utilities to facilitate contacting the persons we care about. There is no way for students to get instant information about their college peers. Therefore, students who are looking to connect with other students are unable to do so to their liking. Considering students spend a lot of time at school, it is undoubted that such application would be extremely relevant and helpful in order for their life at school to be at its best state.

Additionally, we have also noticed that education is an unexploited domain in the programming world, even though education is essentially a domain everyone is involved at some point in their life. Hence, our goal is to make everyone’s experience at schools the best possible. We want people to have an optimal educative experience by providing them with a service that will allow them to facilitate their communication at school. Incidentally, what makes Moods special is that not only will it target the educative area, but also entertainment. Moods will offer students tools to help people study, but ultimately it will allow students to hang out together. Students wanting to eat for instance, will be able to connect with other students that are hungry. Students wanting to play outdoor soccer, will be able to organize an event. Students wanting to relax, will be able to reserve a room to lay back in. Moods has the opportunity to be successful because it provides simple solution to problems students meet at school. Our goal is not to provide something complex to students, as simplicity is complementary to efficacy. This is why each user is going to be able to reach out to other students within a click. We want students to be able to meet their friends and new people in the College more easily by using our application.

The lack of communication between students at school is essentially due to the fact that schools are too big. It is often difficult for people to know where their friends are, what they are doing, who they are with, what's their schedule, etc. Therefore, the simple idea of meeting up with a friend is difficult, since you absolutely need to have their phone number and to have them use it exactly when you attempt to communicate with them. On the other hand, Moods would solve that problem by allowing students to publish their location and status/mood so that people don't waste time anymore asking basic questions such as 'Where are you?' 'What are you doing?' to every single one of their friends. Instantly, the user will be able to see which students are available, what they are doing, and at what location they are (cafeteria, library, computer lab, etc.).

Design

Analyze, define, and understand the problem:

The major problem is that students' are having trouble connecting quickly with their friends within their college. The main reason why is that they have no way of consulting their peers schedule, thus it is extremely inconvenient as students don't know when they have common breaks with their friends. This prohibits the students from doing things they would prefer doing together, such as studying, eating, learning, relaxing, and the list goes on.

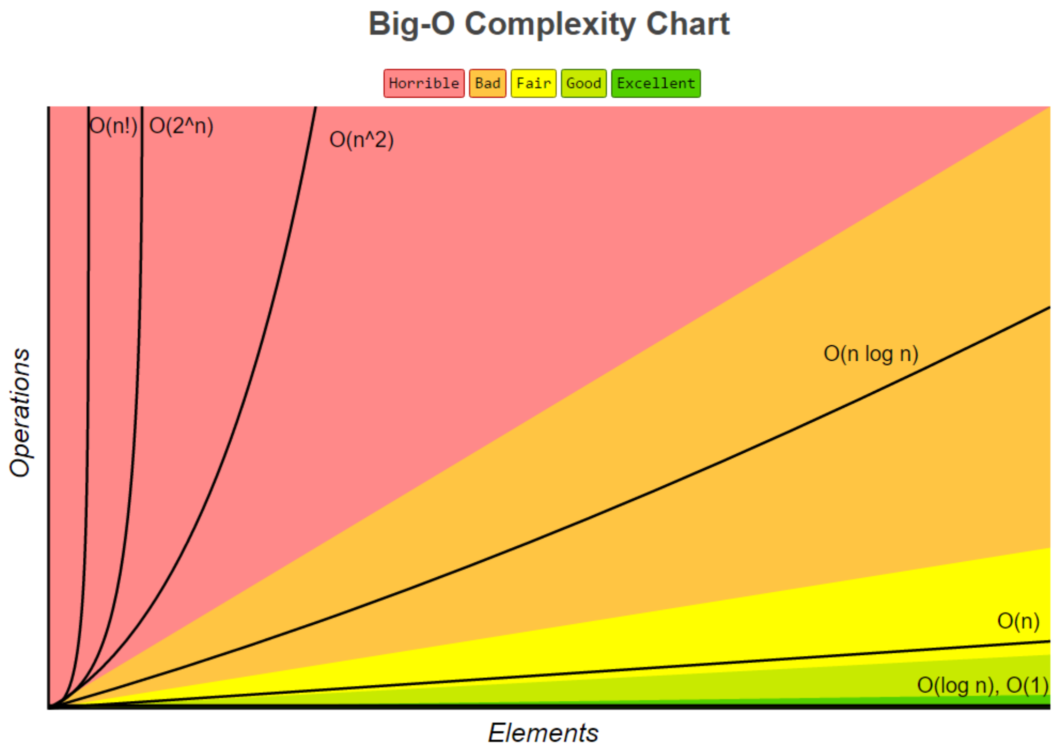
Our application would be the perfect solution to that problem as one of its features involves storing every student's schedule in database. Using an efficient mathematical algorithm, our tool will be able to perfectly match which students have common breaks. In other words, our users will be able to know who's available when you are to do anything you want, easily. In order to display what is that thing you want, you will be provided a tool to update your mood, thus your friends will know what you are up to.

Additionally, colleges are important in size, thus locating your best friends can most of the times be time-consuming. Therefore, our application would remedy to that problem by providing an option to share your location with your friends, so that they know exactly where you are when they need your presence, in the matter of seconds.

Again, there is an obvious need to simplify the students' social life at college by helping them connect together easily and quickly.

Algorithm:

The mathematic will be behind the scene. The application by itself is not based on any kind of science. The application is a social media, hence does not have a scientific purpose. However, a lot of mathematics is used since the program will have to deal with a lot of lists and arrays. For example, the search algorithm for finding profiles and lists is a binary search. This means that the average efficiency of the algorithm has a Big O notation of log(n). In the worst case scenario, the efficiency is O(n). This means that if the element that the computer is looking for is not in the array, there will be the same amount of comparison as there are elements in the array itself. As explained by the chart below, the time complexity for the binary search is in the excellent tier. Thus, for a big number of user profiles, it is the perfect algorithm for this given situation.



Moreover, this application will also require a lot of sorting of profiles in order to return an adequate list of friends. In general, it is difficult to find a good sorting algorithm. Moreover, our application will require sorting an array with non-primitive data types. Thus, the search algorithm is a normal insertion sort. The time complexity of this sorting method is of n^2. Yet, the arrays to sort do not have a considerable size, since one can assume that a user will not have a significant amount of friends, no matter which algorithm is used, the time difference will not be a big one.

Explaning Design:

For the application to properly work, it must connect to a web service to store and retrieve information from a database. The architecture of the system lies within 2 layers. The application layer and the web service layer. We will start by explaining how our application layer will work and then we will explain the internal structure of the web service.

Note : Every class will be assigned an Id based on the template c[xx], to facilitate user reading.

Classes that will be both used in the web server and on the mobile application.

C01 – User

Description of C01:

This class will be the main class for the application's user. Every user will have a user object related to him. It will contain his personal information.

Attributes of C01:

- name : String
- currentMoodId: int
- moodStatus: string
- contacts: ArrayList<User>
- username : String
- password : String
- photoUrl : String
- isOnCampus : Boolean

Methods of C01:

1. public boolean addContact(string: userId)

This method will be responsible of adding a contact to our contact list. It returns true, if the contact has been added and false if the contact is already in our contacts.

2. public boolean removeContact(string: userId)

This method will be responsible of removing a contact from our contact list. It returns true, if the contact has been removed and false if the contact isn't in our contacts.

3. public void setMood(int moodId, string message)

This method will set the user's mood so other people can see it.

4. public void setSchedule(Schedule schedule)

This method will save the given schedule by sending it to the schedule repository.

Link with other classes:

This class is linked to almost every class: Schedule, Mood, Poke

C02: Mood

Mood is a class of enum type. Basically, there are 7 different moods values possible: Hungry, Studying, Chilling, In need for help, Lift Home, Not Available, Custom

Attributes of C02:

- No attributes.

Methods of C02:

- No Methods

C03 – Poke

The poke class has the responsibility to help users connect together. A poke can be a conversation starter or simply a way to answer users' moods. A poke can be linked to a mood so the recipient of the poke understand that the poke is linked to his mood. When a user received the poke he can accept or refuse it.

Attributes of C03

- timeSent : Date
- timeAnswered: Date
- isAccepted : Boolean
- message : string
- moodId : int

Methods of C03

- Public void accept() :

This method is responsible to set the poke as accepted and specify the time it has been answered.

- Public void refuse()

This method is responsible to set the poke as refused and specify the time it has been answered.

C04 – UserActionManager

The User Action Manager class is the main façade of the application. This class is the one that will take the users actions and contact the proper classes to complete the actions.

Attributes of C04

- currentConnectedUser : User
  Explanation : When a user will connect, he will be stored in a property called currentConnectedUser so the application can keep track of who is currently connected.

Methods of C04

- Public boolean signIn(string username, string password) :

This method will try to connect the user with the given username and password. It will contact the UserRepository that will on its side check with the database if it’s the right username, password combination. This method will return true if everything when fine, and false if no match were found.

- Public void signOut()

This method will disconnect the user by simply removing the current user from the user manager.

- Public List<User> searchUser(string query)

This method will search a given string and return found users. Results will be filtered by users’ names.

- Public bool addUserAsFriend(string userId)

This method will add add a given user as friend. It will go through our users object and call the “add contact” method.

- Public void setLocation(boolean oncampus)

This method will set if we are on campus or not.

- Public Schedule getSchedule()

This method is used to load the users schedule by calling the getSchedule from the User class. If it’s the first time loading our schedule, it will create an object schedule with 7 empty days. After the creation, it will save the schedule by calling the setSchedule in the User Class.

- Public void setSchedule(Schedule schedule)

This method is to save the changes made to the schedule. It will send the schedule with the user’s id linked to it to the ScheduleRepository which will handle the saving.

- Public void sendPoke(int moodId, string message, int recipientId)

This method will be used to create a poke object and assign it to the proper user.

- Public void acceptReceivedPoke(Boolean : answer, string message)

When a user receives a poke, he can accept or refuse. He can also write down a message.

- Public void setMood(string message)

This method with call the users setMood method that will set the users mood.

- Public List<User> viewFriendList()

This method will contact the UserRepository to load and show our friends.

- Public void viewReceivedPokes()

This method will load and show our poke requests. It will contact the PokeRepository that is return the proper objects.

- Public void viewFriendsMoods()

This method will load and show our contact’s moods by contacting the UserRepositories and loading our friends moods.

C05 – Schedule

The schedule is assigned to a user. A user can only have one schedule. Basically, the schedule will group 7 days objects everyone representing one day of the week.

Attributes of C05

- userId : int
- days : ArrayList<day>

Methods of C05

Public void saveSchedule()

This method will save the schedule state by sending it to the ScheduleRepository class of the web service by doing an Http Request. (More details are going to be covered in the Web Service architecture explanation).

C06 – Day

The day class is what linked by aggregation to the schedule class. Like said before, the schedule class will have 7 day objects. Each day will be labeled with a the enum DayOfWeek. Ranging from 1 for Monday to 7 for Sunday.

Attributes of C06

- Int scheduleId : schedule
- dayOfWeek : DayOfWeek
- breakTimes : ArrayList<BreakTimes>

Methods of C06

- addBreakTimes(BreakTime breaktime)

Since a day will be constituated of breaktimes, this method will help the use manage the list of breaktimes inside the day.

- removeBreakTimes(BreakTime breaktime)

This method is used to remove a break time from the breakTimes list.

C07 – BreakTimes

The day class is what linked by aggregation to the schedule class. Like said before, the schedule class will have 7 day objects. Each day will be labeled with an enum DayOfWeek. Ranging from 1 for Monday to 7 for Sunday.

Attributes of C07

- startTime : Date
- endTime : Date

Methods of C07

- setStartTime(Date : startTime)
- setEndTime(Date : startTime)

<u>How will the web server work?</u>

We are going to create a simple web service that will be java based. This web service will run on a tomcat 7 server. Since tomcat can run on any Linux based Server, we are going to have our own Linux Server and run the Tomcat server on it.

Now for the web service itself, it will have the role to listen on a specific port for http requests. In fact, the mobile application will send http request asking for information and the web service will properly answer based on what the request is asking.

For example, if the user tries to login, the application will send a post http request to http://ourserverpath.com/user/signin. If additional information is required with the request, we can add it inside the request's parameters.

Once the server will receive the request it will use the proper method linked to the request and return the information that has been found as JSON objects.

Here are the classes needed for our web service

Note: The id of classes inside web services will be SC-[xx]

SC-01 – UserRepository

Description of SC-01

The UserRepository class is going to manage the connection between the User database Table and the Application.

Methods of SC-01:

- Public void create(User : user)

This method will contact the database and save the user.

- Public User signIn(string username, string password)

This method will try to find the related user in the database based on the username and password. It will return null if the password and username are wrong, otherwise it will return the object.

- Public List<User> findMyFriends()

This method will contact the database and get our friends.

SC-02 – PokeRepository

The PokeRepository class is going to manage the connection between the Poke database Table and the Applicaton.

- Public void sendPoke(Poke : poke)
- Public void viewFriendsPokes()
- Public void answerPoke(Boolean : answer)

SC-03 – ScheduleRepository

The ScheduleRepository class is going to manage the connection between the Schedule database Table and the Applicaton. This repository is linked to the DayRepository since when creating a schedule, it will create 7 days objects by calling the DayRepository.

- Public void create (Schedule : schedule)

Will create the user's schedule. Before creating the users schedule it is going to validate if the user doesn't already have a schedule associated to himself.

- Public Schedule get(int UserId)

Will load the schedule of a given user from the database. Returns null if he doesn't have one.

- Public void update(Schedule schedule)

Will update the given schedule in the database.

SC-04 – DayRepository

The DayRepository class is going to manage the connection between the Schedule database Table and the Applicaton.

- Public void create (Day: day)

This method will create a day object inside the database. It will usually be called from the Schedule Repository, since it is the one that is going to manage Day savings.

- Public void update(Day: day)

  This method will update a day object inside the database. It will usually be called from the Schedule Repository, since it is the one that is going to manage Day updates.

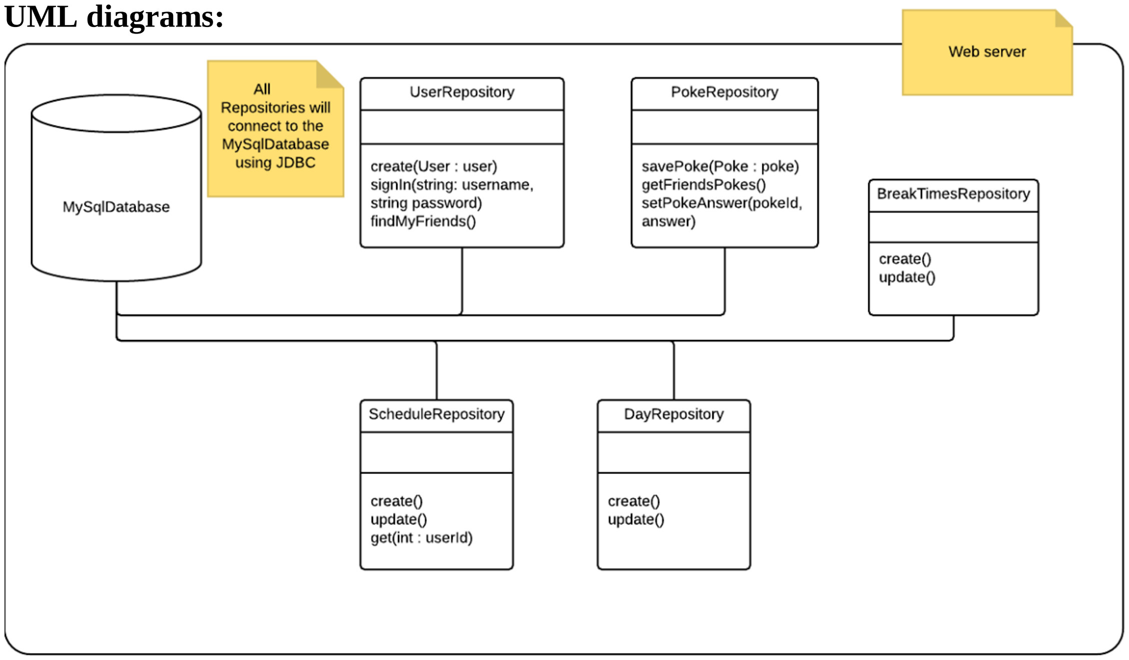
SC-05 – BreakTimesRepository

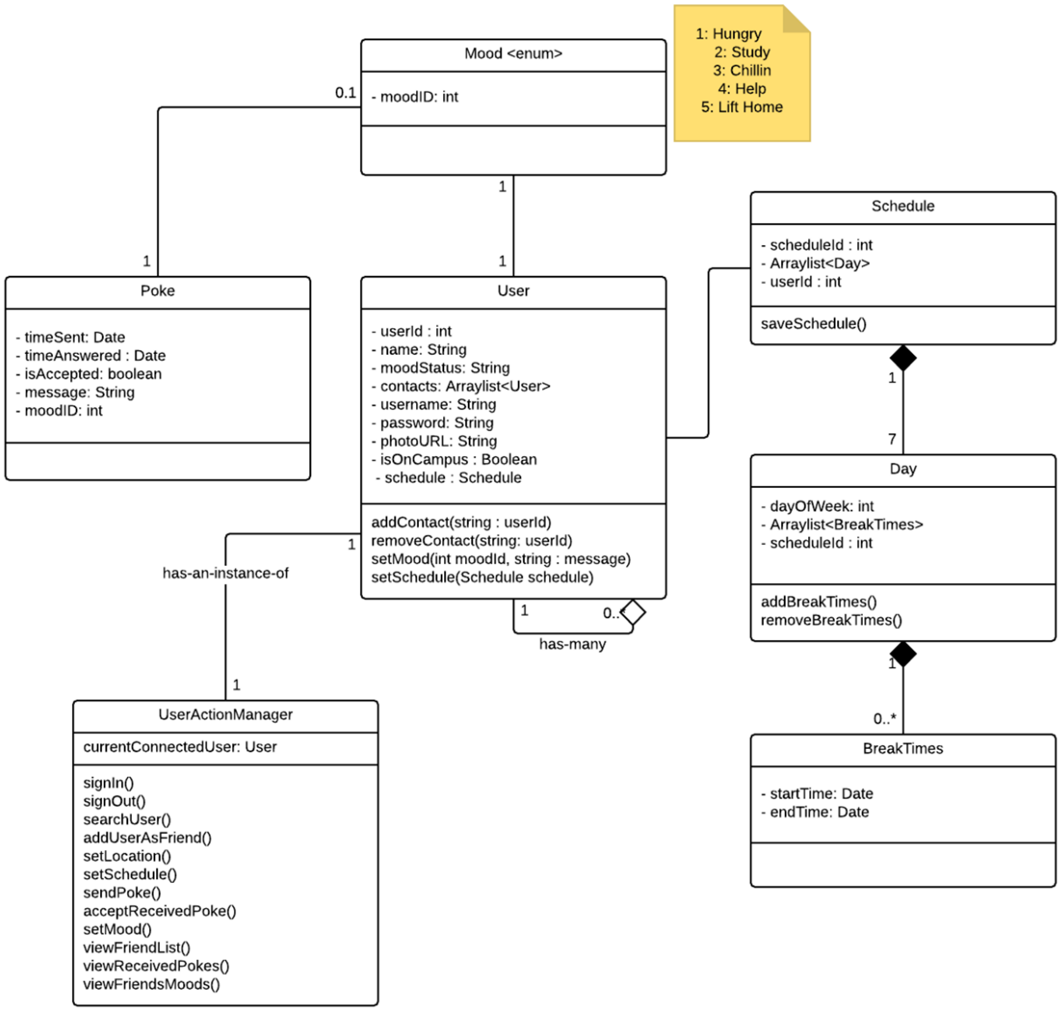
The BreakTimesRepository class is going to manage the connection between the Schedule database Table and the Applicaton.

- Public void create(BreakTimes : breakTimes)

This method will handle BreakTimes creations inside the database table related to BreakTimes.

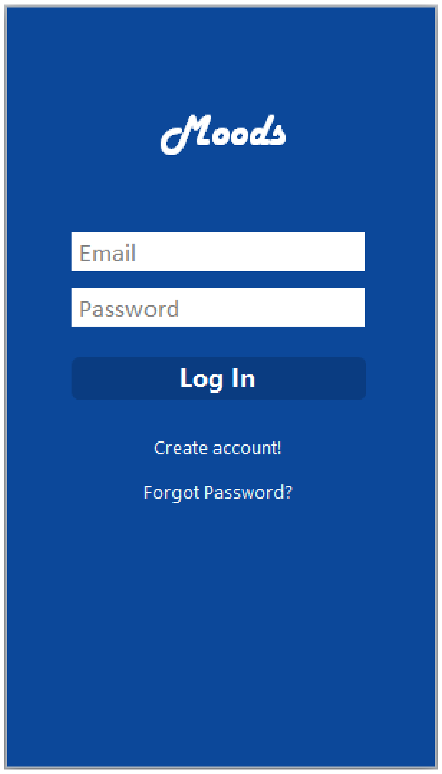
UML diagrams:



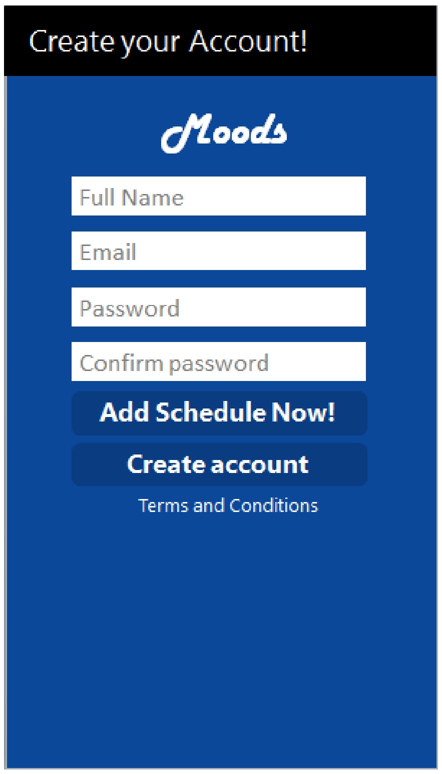


Graphic User Interface:

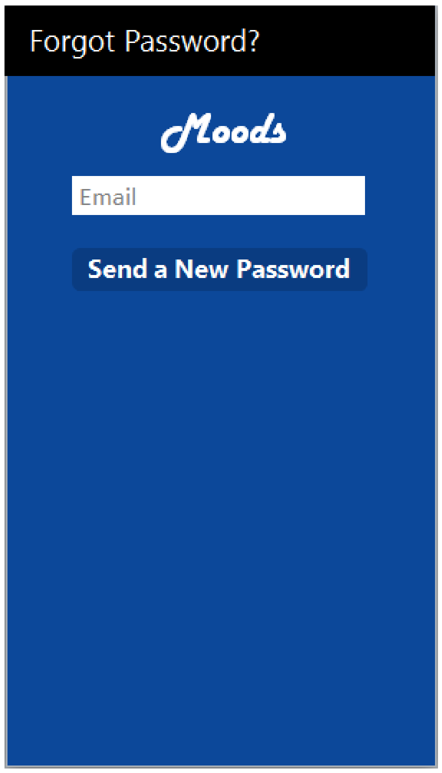
Login Page



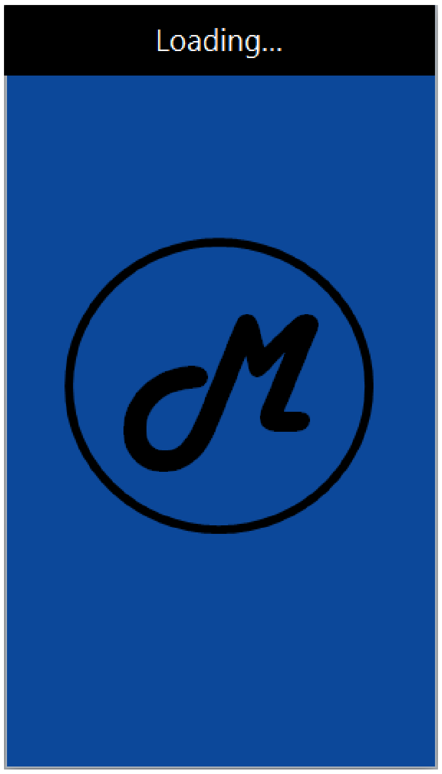
Create Account Page



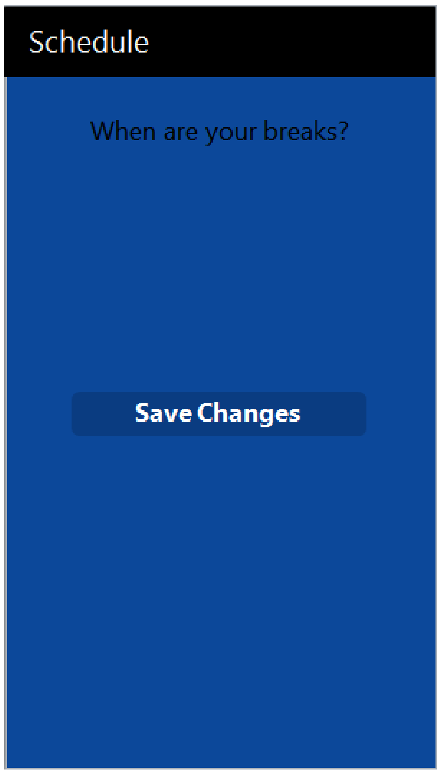
Forgot Password Page



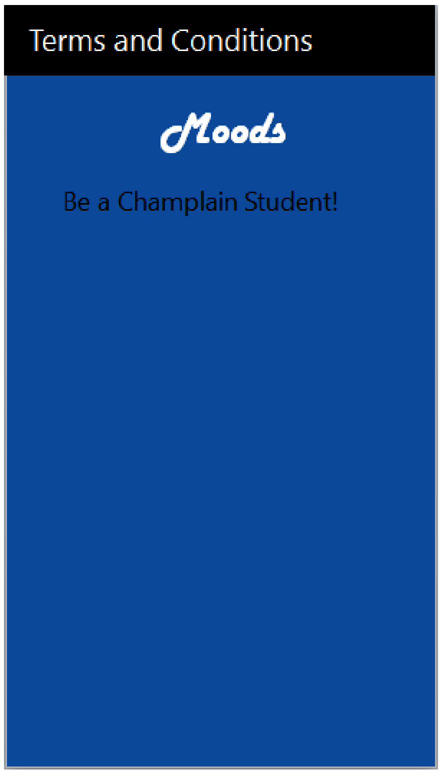
Loading Page



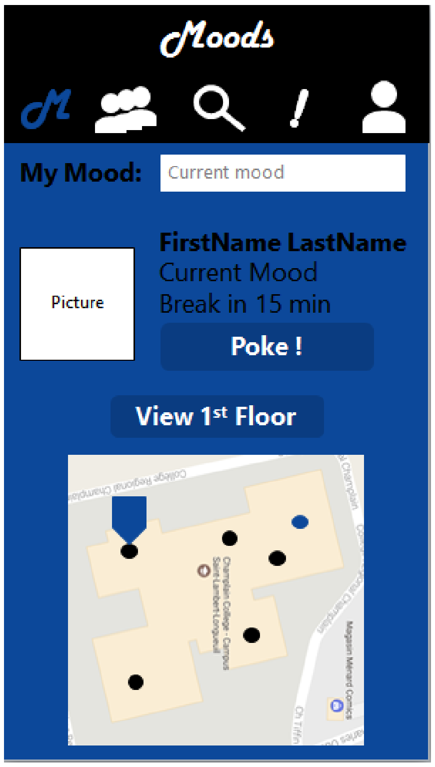
Schedule Page



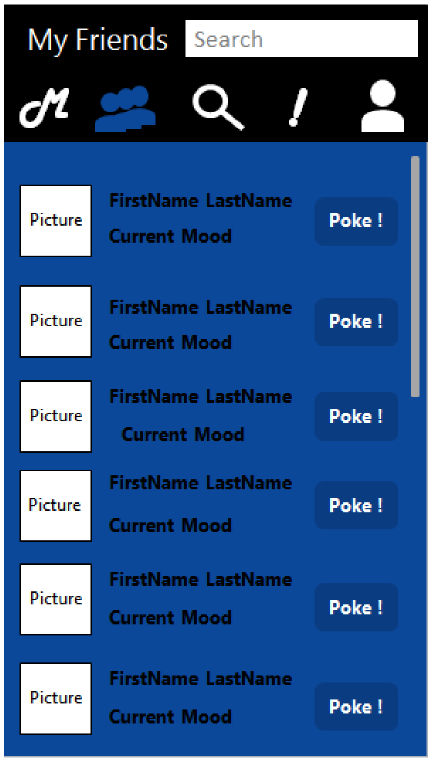
Terms and Conditions Page



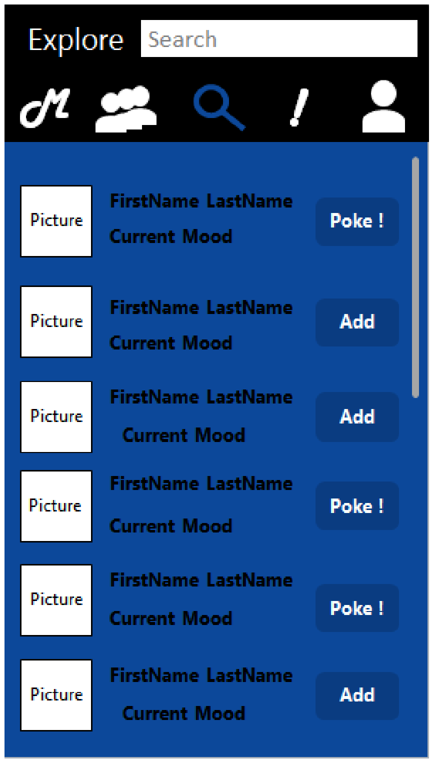
Home Page



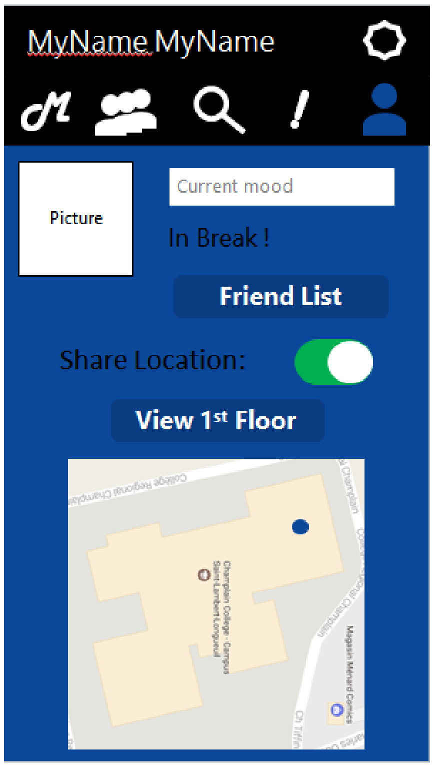
My Friends Page



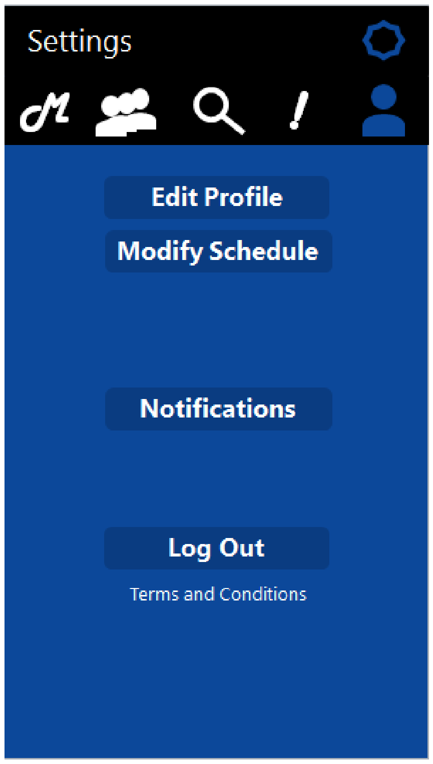
Explore Page



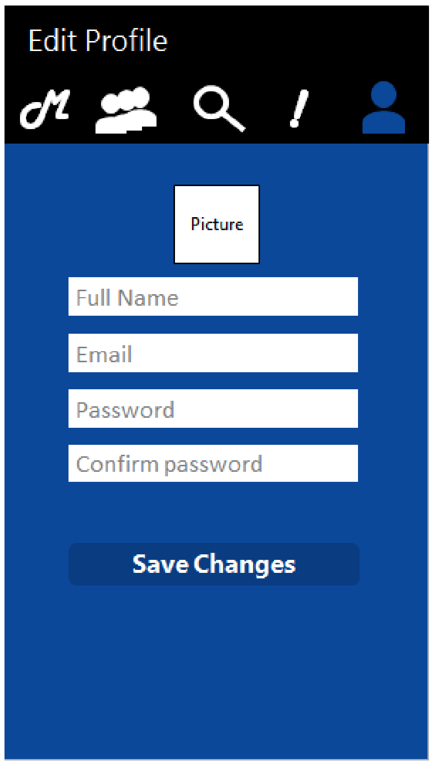
My Profile Page



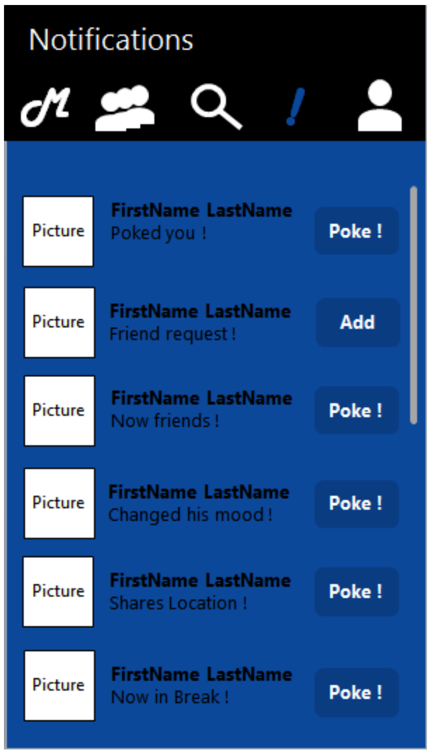
Settings Page



Edit Profile Page



Notifications Page


Notifications

Picture | FirstName LastName Poked you ! | Poke !

Picture | FirstName LastName Friend request ! | Add

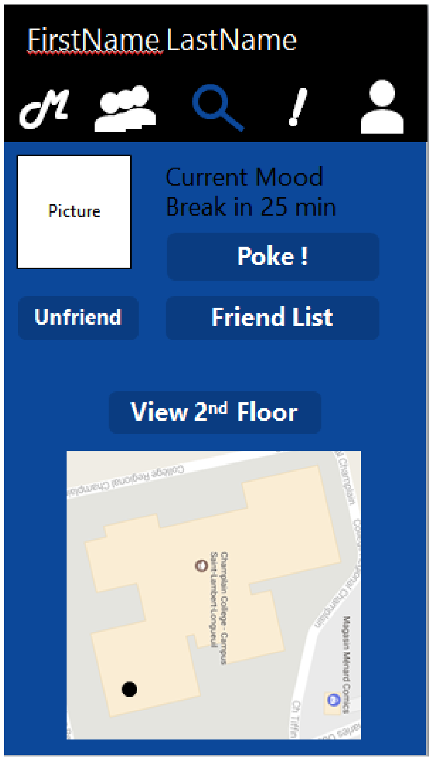
Picture | FirstName LastName Now friends ! | Poke !

Picture | FirstName LastName Changed his mood ! | Poke !

Picture | FirstName LastName Shares Location ! | Poke !

Picture | FirstName LastName Now in Break ! | Poke !

Friend Profile Page


FirstName LastName

Picture

Current Mood

Break in 25 min

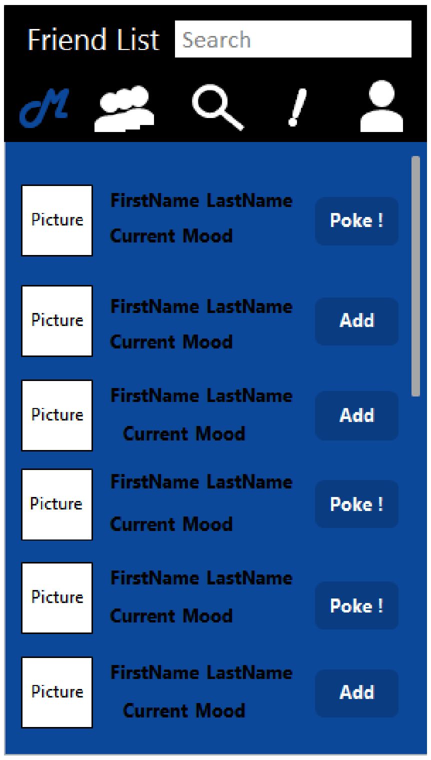
Poke !

Unfriend

Friend List

View 2nd Floor

Friend's Friend List Page


Friend List | Search

Picture | FirstName LastName Current Mood | Poke !

Picture | FirstName LastName Current Mood | Add

Picture | FirstName LastName Current Mood | Add

Picture | FirstName LastName Current Mood | Poke !

Picture | FirstName LastName Current Mood | Poke !

Picture | FirstName LastName Current Mood | Add

Moods uses a toolbar to navigate between the Home page, the My Friends page, the Explore page, the Notifications page and the My Profile page. When on the Home page, My Mood is easily editable and clicking on a black point on the map shows your friend's profile. The My Friends page makes it easy to find your own friends. When on the Explore page, it is easy to find your own friends as well as other students. The Notifications page shows all your notifications from friend requests to poke notifications. When on the My Profile page, you can choose to share your location as well as go to your settings. From the Settings page, the user can choose to edit My Profile, access notifications, modify schedule, or log out. When clicking on a student's name, the app takes you to the student's profile where you can access the student's Friend List.

Timeline:

Moods Project Timeline

Task	Plan Date	Assigned Person	Notes
Introduction	19-Jan	Whole team	Team choosed, discuss interests and ideas
Discuss Project	24-Jan	Whole team	Discuss the choice of project and the possibility of completion
Choose Project	26-Jan	Whole team	Moods Android App
Project Proposal	31-Jan	Whole team	Project's features and used softwares
Install Softwares	02-Feb	Whole team	Install and start getting familiar with Android Studio and GitHub
Familiarize with Softwares	07-Feb	Whole team	Familiarize withb GitHub and Android Studio
Familiarize with Softwares	09-Feb	Whole team	Do the Android Studio Online Lessons
Design	14-Feb	Will	App Layout
UML diagrams & SQLite Database	16-Feb	Amir	Understanding of SQLite and do the UML diagrams
Features & Algorithm	21-Feb	Étienne	Define the final features and the math algorithm
Finish Proposal	23-Feb	Whole team	Finish all other requirements for the proposal
Prepare presentation	Reading Week	Whole team	Using the final proposal, prepare an orl presentation
Oral Presentation	07-Mar	Whole team	Summary of written report
Start app layout	09-Mar	Étienne & Will	Design on Android Studio
Start database	14-Mar	Amir	Database and Class implementation at the same time
Finish 7 of the 15 windows	16-Mar	Étienne & Will	Finish half of layout on Android Studio
Continue working	21-Mar	Whole team	Database and layout
Finish layout	23-Mar	Étienne & Will	Finish all 15 windows on Android Studio
Start basic implementation	28-Mar	Will	On Netbeans (use UML diagrams)
Start networking	30-Mar	Étienne	Use computer at home
Finish database	04-Apr	Amir	Working database
Start implement	06-Apr	Étienne & Will	On Android Studio with the finished windows
Continue networking	11-Apr	Amir & Étienne	Using Étienne's computer
Continue working	13-Apr	Whole team	Will: Implementation
Continue working	18-Apr	Whole team	Amir & Étienne: Networking
Finish implementation	20-Apr	Étienne & Will	Finished
Finish networking	25-Apr	Amir & Étienne	Working network
Finalize app	27-Apr	Whole team	Combine all
Testing	02-Apr	Whole team	Debug
Testing	04-Apr	Whole team	Final testing
Final Version	09-Apr	Whole team	Finished Final version of app
Final Presentation	11-Apr	Whole team	Oral presentation of completed app

Softwares:

Platform**:** Android Java

The application is expected to run on Android Devices. All devices using Android Lollipop and above will have an access to the application.

Menu: User profile

The user will interact with the application throughout multiple pages and menus. There will be a home page for the user to change his mood and status, yet the rest of the settings will be accessed through the settings page.

Accounts: Login Authentication

The users will be able to log from a database. A basic encryption scheme will be used to protect the user's privacy. Using this database, the application will be able to retrieve the user's information.

Code sharing platform: GitHub

The GitHub platform is used in order to efficiently work on this project. Using this service, one may share his codes and resources in real time with his coworkers.

Code IDE: Netbeans & Android Studio

Netbeans: This program is used to program the Database and the networking since the coding environment is better for this kind of programming. Android Studio is not made for such programming.

Android Studio: Android studio is used in order to manage and program everything that is in relationship with the activities. This program is optimized for a visual environment in order to code XML and Java files efficiently.

Features

- Location of users (sends to selected contacts)
  - Uses GPS location from device (API integrated in Android Studio)
- Mood & Statuses (Looking for someone to Eat, Study, Tutor, Lift/Go Home, Chill)
  - Helps you communicate with friends
  - Stored in database (resets every 24h)
- Map of Champlain
  - Drawn by us (JPG format)
    - Ask school for plans
  - May use Google Map SDK
- Notifications to cellphones
  - Send if the friends is near, accepted a “poke request”, friend request, etc.
- Access to vibrations or other phone components
  - Access to GPS
  - Access to Vibrations
- Networking
  - Use SQL databases (SQLite implemented in Android Studio)
  - Store user information on server
    - User name
    - Email
    - Pictures (Optional)
    - Schedule
  - Store user information on phone
    - ID
    - Temporary mood

Conclusion

In conclusion, as explained in the Design, for the application to properly work, it must connect to a web service to store and retrieve information from a database. The architecture of the system lies within 2 layers. The application layer and the web service layer. The design has allowed us to see how we will exactly proceed in order to build the application. The most challenging part was to understand how the web server will work, as it is something that we have not studied in class. Nonetheless, after hours of research, we have now a fair understanding of how our server will interact with the application layer. The design was overall far more complex than what we thought, as we have to handle every single interaction between the user and other users. Therefore, we have an overwhelming amount of methods that we will need to treat properly in order for the application to run seamlessly. Considering we have less than two months to implement the coding, it will require a lot of work and discipline, as we have to be the most efficient possible in order to achieve what we initially planned.

Finally, we are a team of 3 devoted college students who want to change the way Champlain students interact with each other. Our goal is to design an efficient application called Moods who will aim on the wellbeing of the student's relationships. This program will allow Champlain students to connect more easily together during breaks. Each user will be able to share his current mood (a small sentence that describes his current status i.e. hungry, need help is [Course name], bored, etc.). Our project uses a SQL database to store each of the user's course schedule, friends and mood. Most of the program's interactions will be between the database, the user and the user's friends. There will be a certain amount of Android pages (activities), but most of the work is in the networking.

This application will have a scientific purpose, yet it will use a lot of algorithms in order to efficiently deal with the lists of elements and arrays of data. The overall application should be finished for the 27th of April and should be tested and fully operation for the 9th of May.