TCSS450 - Mobile Programming   
Project Requirements

# Introduction

You and your group must create a Chat client and server. The requirements for the project are listed later in this document. Your group will use the Agile software development practice to complete this project.

# Due Dates:

Sprint 0 Review (5%) Thursday, 22 October

Sprint 1 Review (10%) Thursday, 5 November

Sprint 2 Review (10%) Thursday, 19 November

Sprint 3 Review (10%) Thursday, 10 December

Final Presentation (5%) Tuesday, 15 December

Final Submission (10%) Sunday, 13 December

Peer reviews (5%) Thursday, 17 June

# Scrum Process

For this project, we will use the Scrum framework to implement the software development process. Your work will be split into 3 distinct sprints, each with its own due date. Each sprint will consist of feature development, testing, code review, and group ceremonies.

## Ceremonies Explained

Sprint Planning: A one- to two-hour meeting during which the team determines what items from the backlog should be completed during this sprint.

Daily Scrum

Backlog Grooming

Sprint Code Review

Sprint Retrospective

Sprint Review

## Sprint Process:

* At the beginning of each sprint:
  + Perform a synchronous **Sprint** **Planning** 
    - Run by the scrum master
    - Take meeting notes
    - Manage/groom project board (for this upcoming sprint)
* During each sprint
  + Develop, test, and deploy new features
  + Meet daily (M-F) for synchronous **Daily Scrum**
    - Run by the scrum master
    - Take meeting notes
    - Manage/groom project board (current sprint and tasks)
  + Meet mid-spring for synchronous **Backlog Grooming**
    - Run by the scrum master
    - Take meeting notes
    - Manage/groom project board (looking forward to next sprint)
* At the end of each sprint
  + Perform a synchronous **Sprint Code Review**
    - Run by the scrum master
    - Discuss and review:
      * Commenting/documentation
      * Formatting
      * Refactoring
  + Perform a synchronous **Sprint** **Retrospective**
    - Run by the scrum master
    - Discuss and review:
      * What worked well?
      * What didn’t work well?
      * Discuss any team/administration related action items
  + Entire team will meet with the instructor in **Sprint Review**
    - Run by the scrum master
    - Display documentation of meetings
    - Demonstrate the Sprint’s acceptance criteria

Keep all meeting notes and documentation in a shared folder on Google Drive. Add your instructor as an observer to the shared folder. Keep each meeting type’s notes in a separate file but use the same file for each following meeting of that type. Templates are provided.

## Typical Sprint Schedule

1st Thursday:

Sprint Starts

Synchronous **Sprint** **Planning**

Begin implementation of new functionality

1st Friday – 1st Wednesday

Synchronous **Daily Scrum**

Continue implementation of new functionality

Test new functionality

2nd Thursday

Synchronous **Backlog Grooming**

Synchronous **Daily Scrum**

Continue implementation of new functionality

Test new functionality

2nd Friday and Monday

Synchronous **Daily Scrum**

Continue implementation of new functionality

Test new functionality

2nd Tuesday

Synchronous **Daily Scrum**

Continue implementation of new functionality

Test new functionality

END OF DAY – Code Freeze (No new functionality)

2nd Wednesday

Synchronous **Daily Scrum**

Test and Debug Sprint functionality

Prepare Demo for client

Synchronous **Sprint Code Review**

Synchronous **Sprint Retrospective**

3rd Thursday

Meet with Client (Instructor) **Sprint Review**

Next Sprint Starts

Next Sprint Synchronous **Sprint** **Planning**

Next Sprint Begin implementation of new functionality

Saturdays and Sundays

Synchronous and/or asynchronous communication

Continue implementation of new functionality

Test new functionality

# Project Requirements

The following list contains the minimum requirements of the project. Some of these may seem vague or confusing. It is your job to talk to the customer (your instructor) and gain clarification on the requirements. (EC) -> is extra credit.

* Register
  + Required info:
    - First and last name
    - Email (Verified)
    - Nickname (Used for in app display)
    - Password
  + Server-side and client-side checks for validity
    - Define a set of minimum requirements for password and nickname
  + Upon registration attempt, do not log the user in, display a page informing the user of the verification process
* Login
  + Must use email
  + Server-side and client-side checks
    - No empty fields, username and password matches the data base, etc.
  + Option to stay logged in
    - Credentials are saved and the user is “logged in” every time the app starts
    - Requires ability to log out
  + Upon successful verification of login credentials, display the home/landing page
* Connections (think friend, link, contact, etc.)
  + View existing Connections
  + Search for new Connections then send Connection request to search results
    - Search on:
      * Email
      * Nickname
      * First and last name
  + View Connection requests
    - Sent by you
    - Sent to you
  + Remove an existing Connection
  + Send invitations to email addresses to use this chat app/service (EC)
    - Server-side not client side
* Chat
  + Individual or “Group” chat with an existing Connection(s)
    - Start a chat
    - Send and receive messages
      * Messages must be stored (server side)
      * “See” when the other person is typing (EC)
      * Store previous messages locally (EC)
    - Continue an individual chat with an existing Connection
    - Open a new chat request from an existing Connection
    - Send/receive Images (EC)
* Notifications
  + All notifications should appear when:
    - The app is not in no longer visible to the user (not in the foreground)
    - The user is viewing the app but not the context of the notification
      * For example, while user A is viewing a chat with user B, a message from user C is received.
  + Notifications for the following reasons:
    - new connection request (*conn req*)
    - new conversation request (*convo req*)
    - new messages (*msg*) from existing conversations
  + Notifications when the app is not visible should only appear when “logged in” option is selected
  + Notifications display:
    - via status bar when the app is not in the foreground
      * Selecting the notification opens app to the correct state
      * Status bar/notification drawer will show the sender and message type
    - via some app UI feature when the app is in the foreground
      * Selecting the notification navigates to the correct state
* Weather:
  + Display the weather forecast for
    - the devices location
    - a location searched by zip code
    - a location chosen on a map
  + Ability to save any of the above locations to display that locations weather forecast at a later time
  + Weather display should include
    - current conditions
    - 24-hour forecast
    - 5 or 10-day forecast
  + Explore different 3rd party Weather APIs to find one that matches your requirements
  + Wrap this web call in your own web server
* Home/Landing page
  + When a user logs in, a home page will display.
    - The more dynamic content displayed the better. For example:
      * Current weather forecast for device location
      * Any notifications
      * “links” to most recent chats
* Look and Feel
  + A custom logo for the Android app and a graphic on the login page is required.

# Version Control Software

Your group will development and maintain two distinct (Client and Server) code bases for this project. You are required to use version control software to store your projects. It is recommended to use Git and GitHub but you may use other platforms such as SVN on CSSGate. You are required to add your instructor to the list of contributors for each project.

Your group will maintain a story board for your project. You are required to use GitHub’s story board software. You are required to add your instructor to the list of contributors for your story board. It is suggested that you create your story board inside of either your Client or Server project.

Please use **cfb3** when adding your instructor to your project.

# Lab Schedule

Lab 3: Connect to a web service

Concept:

Connect to a web service from Android App

Connect to a 3rd party web service (via Android App)

Example:

Hello World

Login and Register

Recycler View (3rd party API)

Lab 4: Write a web service

Concept:

Write the web service with a database used in Lab 3

Connect to a 3rd party web service (via web service)

Example:

Hello World

Login and Register

Recycler View (3rd party API)

Lab 5: Push Messaging

Concept:

Write a web service with a database and connect to it

Notifications, Background Services, and alarm manager

Example:

Simple Client/server chat application

Lab 6: Advanced Sign

Concept:

Store credentials, auto sign-in

Example:

Streamline Authentication ap from Lab 3/4

Lab 7: Locations

Concept:

Locations and Maps

Example:

Gain the device locations via GPS and display on Google Map using the Google Maps API

# Web Server

This project will require your group to write and administer an external web server. In class, you will have lectures and labs demonstrating Node.js, Heroku, and PostgreSQL (DBMS). It is recommended but not required that you use those technologies for your project. Other options include PHP on CSSGate.

# Database

This project will require the server to create and administer a database for your web server. It is recommended that you use the following tables for your database. You will be provided with scripts used to create these tables when Lab 4 is published.

*You are free to alter/ignore these suggestions to meet your group’s design.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Members | | | | | |
| MemberID (Primary Key) | First Name | Last Name | Username | Email | Verified |
|  |  |  |  |  | 0/1 or T/F |

|  |  |  |  |
| --- | --- | --- | --- |
| Contacts | | |  |
| Primary Key | MemberID\_A | MemberID\_B | Accepted |
| Auto inc |  |  |  |

|  |  |  |
| --- | --- | --- |
| Chats | | |
| Primary Key | ChatID | MemberID |
| Auto inc |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Messages | | | | |
| Primary Key | ChatID | Message | MemberID | TimeStamp |
| Auto inc |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Locations | | | | | |
| Primary Key | MemberID | Nickname(?) | Lat | Long | Zip (?) |
| Auto inc |  |  |  |  |  |

# Appendix A – Group Work Policies (Adapted from Prof. Josh Tenenberg’s group work policies)

As mentioned previously, the project involves the specification, design, coding, testing, and documentation within a group. Each group will meet outside of class on average between 3 to 6 hours weekly. The following are a minimal set of commitments that each student is expected to uphold:

* To meet weekly with your group at your regularly scheduled time; lack of attendance (considered to be missing more than one meeting) may result in your receiving a considerably lower grade than your group mates on the project.
* To read your email at least once per day on Monday through Saturday (taking Sunday as a legitimate day to not be in contact, though Saturday might be taken as the day off -- please negotiate this among yourselves, being sensitive of one another's religious and personal beliefs/practices) and to respond to any issues, questions, or requests from group mates. Note that by default, email should be sent to everyone's uw account; if a member would like group mates to send email to any other account, this should be made clear to all group members.
* To make a good-faith effort to meet with your group on an ad-hoc basis as needed.
* To treat your group mates with respect in speech and manner so as to create an environment conducive to learning; being friends with or liking your group mates is not the objective of the group experience, though it may enhance the experience—the objective is to work effectively as a group regardless of the personal feelings that one has toward the other group members
* To commit to perform a fair share of the group tasks on an individual basis
* To carry out your work commitments in a timely manner
* To uphold a high standard of quality in the work products that you bring to your group. You should complete a work product for presentation to your group at the same level of quality and quantity as if you were handing it in to the instructor
* To hold your group mates accountable for their work commitments, both in terms of quality and quantity
* To inform your group mates as far in advance as possible when you know you will be unable to attend a meeting
* To take responsibility for contacting your group after a missed group meeting to determine what was discussed and what are your new responsibilities for the upcoming week.
* To provide timely information to a group mate who has missed a group meeting concerning the missed meeting and the group mate’s new responsibilities.
* To contact the instructor immediately if any group member misses more than 2 meetings during the term. I will intervene in any case where a group member misses this number of meetings, but it is the group's responsibility to inform me of this.

If you believe that you are unable to meet these commitments due to external constraints, then please discuss this with the instructor immediately. This may result in your reducing some of your external commitments or deferring attendance in this course for a term in which you have fewer such constraints.