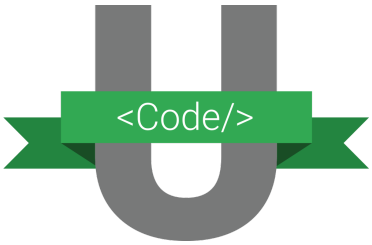


Project Design Doc

Basic Information

Team #: 39
Team Slack Channel Name: projectgroup39
Team Member Names: Anh Duong Nguyen, Leonie Reif, Neil Barooah
Github repo link: https://github.com/neilbarooah/codeu_project_2017

Answer the questions below as a group before your first Project Advisor meeting.
If you get stuck, that's OK! Do what you can and bring your questions to your Advisor.

Here are some ideas from the project designers!

Complete #1 & #2 before your first Project Advisor meeting!

1. Project Design:

What are we going to build?

1. **What is the mission of your product?** What aspect of the project are you most passionate about improving?

   We aim to provide a platform for tourists to connect and explore the city they are visiting. This could include communicating with locals and sharing tips, and meeting up with other tourists at their favorite locations. We are most passionate about improving the usability of the application. This includes working on vital features like persistence, connecting the Relay Server and adding groups. These features would make the product viable and fit the needs of our target group.

   Main Milestones:
   - Chat Functionality (Session management)
   - UI/UX
   - Modifying existing model to support a travel-based chat (cities, topics of conversation)
   - Google service integrations

2. Which features are your top three most important features to accomplish?

1. Persistence
2. Connecting the Relay Server
3. Adding groups

3. Of those three features, rank your first, second, third priority.
    1. Persistence
    2. Connecting the Relay Server
    3. Adding groups

How are we going to build the product?

1. **Outline of Approach:** For each of the three features, what is the high level of your approach to getting the user observable result?
   1. Persistence - When the user logs back into his/her account, previous conversations are still available.
   2. Connecting the Relay Server - The user can chat with people that are logged onto other servers.
   3. Adding groups - The app will be organized into different chat rooms based on major cities the app is available in. Each chat room will contain a list of users currently traveling in the city or those who would like to explore that city. Each chat room will consist of conversations based on topics of interests such as food, bars, night clubs, localities etc. For instance, if a user wanted to visit the Eiffel Tower, they could join a conversation about Eiffel Tower that already exists or create a new one if it doesn't already exist. Additionally, users will have the ability to switch between different chat rooms. We envision this app to be a platform for people with common interests to connect with each other so, at a later point of time, we will enable users within same conversations to DM each other, form smaller groups that will be useful for personal conversations/organizing meetups etc.

2. **Further Research:** What don't you know that you need to know to complete these features?
   1. Persistence - Which database fits our needs and this specific framework?
   2. Connecting the Relay Server - How does the Relay Server work? How do we identify valid servers to connect to?
   3. Adding groups - How do we do suggest groups and chatrooms for a user to join

3. **How will you break down the work?** Who will work on what? Can you divide each feature into smaller chunks? What are those chunks?

   1. Persistence - Loading data when user logs in; Setting up database connection; Designing the database layout; Updating database after new action (adding user, adding chatrooms, adding conversation, sending message)
   2. Adding groups - Implement window when opening app for initial joining of a city. Data class models for groups. Determining the factors we would use for suggestions. Ability to form personal groups and DM other users. Adding persistence for groups.
   3. Implementing UI - This can be broken down into many components. This includes

design (sketches, wireframes), creating UI layouts, creating controllers, hooking up backend functions to the controllers.

4. **What will the most challenging part be?** What do you expect to be the hardest part, and how will you approach it?

   The first challenge for us is to fully understand the existing framework, so we can add persistence and other features in the most. Other challenges include ensuring reliability and efficiency of data storage. Although challenging in this context, we'd also like to ensure security and privacy. As none of the members have prior experience with JavaFX applications, working on the UI is also bound to be challenging.

5. **What is your development timeline?**

We plan to follow Agile development principles for this project. We will divide the 12-week project into 4 sprints so we have sizable chunk of features ready by the end of each sprint. At the beginning of each sprint, we will have a sprint planning season in which we'll select features to work on based on priority, talk about the technicalities of each feature, discuss specifics of implementation and assign teammates to tasks within the features. We will organize tasks in a way that we have continuous flow of delivery at the end of each sprint. Working on tasks would involve spending time on design, coding, debugging and peer reviews.

Flow of tasks based on Project Advisor Meetings:

March 13th
- **Setting up persistence + relay server**

April 3rd
- **Adding Groups + UI**

April 24th
- **UI + UX**

May 15th
- **Google Service Integrations + Bug Fixing**

Persistence

- Sign up for google cloud account
- Investigate app engine service
- Investigate persistence options
- Design a schema for storage
- Come up with basic design sketch
- Online storage with Google Cloud Bigtable

UI + UX

- Investigate java UI frameworks
- General Structure -> new Buttons/Icons
- Add groups

- Nicknames
- Deleting conversations
- Delete users
- Log In / Registration screen/window
- Edit User screen

Backend

- Delete Groups
- Deleting permissions
- Approve access to groups
- Update user information (password, username, nickname)
- Handle duplicate username and conversation title
- Allow nicknames
- Communicate with Google Cloud Bigtable's database

2. Project Implementation

Before beginning each feature, submit the following to your Project Advisor:

- Feature name and description
- Development timeline
- Feature owners

Complete #3 for each feature you complete over the next 12 weeks.

3. Feature Completion Summary

<u>Upon finishing a feature</u> submit a document to your Project Advisor for review with the following information:

- **Feature name and description**
- **Actual development timeline**
- **Who worked on the feature and who did what**
- **Summarize the problems you run into developing this feature**
- **Summarize something interesting you learned while working on it**
- **Explain what resources and tools you found useful while working on this feature**

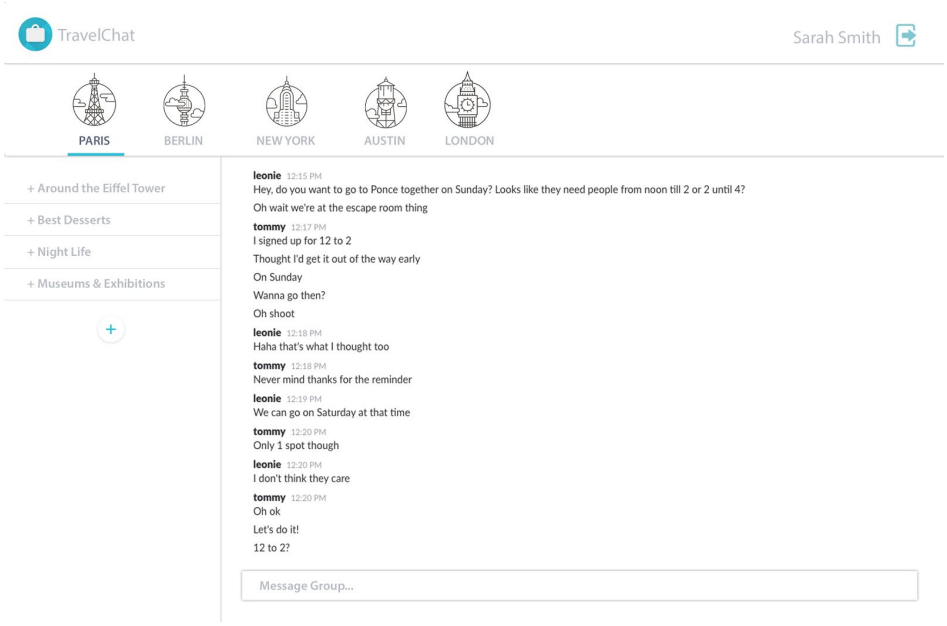
Group Feature

• Description: The Groups feature adds the functionality that messages can not only be categories by conversations, but also by group – which represent in our case cities.
Timeline: first 6 weeks
• Members: Leonie
• Problems: The code looks very similar to the conversation code, so it was hard to structure it in parallel to the existing code.

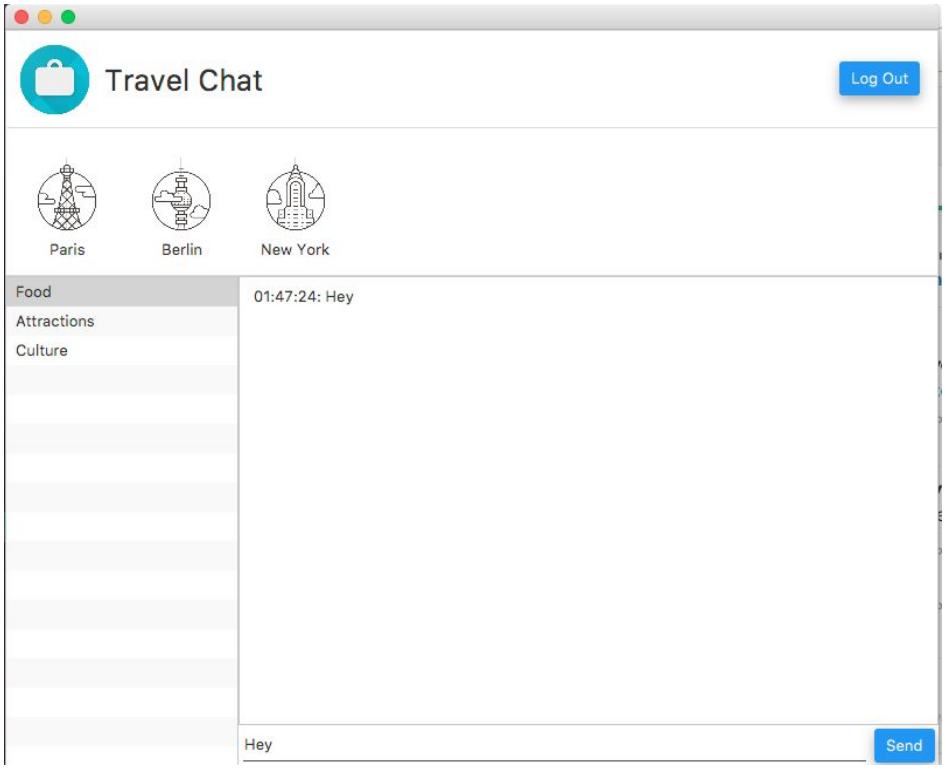
• What I learned: It was a good feature to start with, because it let me explore the complex code base and really understand the code.
• Resources: It was helpful to rebuild the provided conversation code structure.

Material Design UI Update Feature
• Description: The existing swing UI was replaced with a complete new UI using JavaFX and a library called JFoenix.
• Timeline: last 6 weeks
• Members: Leonie designed the new UI in Photoshop and rebuilt it using Scene Builder (Login + Main UI). Neil built the registration page, and everyone worked together on connecting the UI to the backend and fixing bugs.
• Problems: Dynamic resizing of the elements within the frames were tricky. It can be solved by using the automatic resizing and inheriting of preference sizes within Scene Builder settings. All team members were unfamiliar with JavaFX, and developing UIs for a Java-based desktop application. Hence, this was the most challenging of all features that we implemented. It took us a considerable amount of time to choose a viable framework and to implement it correctly.
What I learned: It takes a long time to hook up front end with backend. We also learned about user experience design while talking through features with each other.
• Resources: We downloaded Scene Builder for IntelliJ and installed the library JFoenix for the Material Design elements. Also helpful were the Google Material Design Guidelines and Photoshop to create the initial sketches. We learnt how building UIs can also be relatively easy at times (in comparison to struggling with CSS files for web apps) as there are tools such as Scene Builder to design the layout using a drag-and-drop function.

Initial Design:



Final Implementation:



Delete User

• Description: This provides users the ability to delete other users registered on the application. The purpose of this feature was to conceptualize the idea of an superuser (Admin) that can delete users based on suspicious activity. Another purpose could be for owners of topics to delete inappropriate users from their chat rooms.

• Development timeline: March 18 - March 22 (design and planning), March 22 - March 26 (development), March 26 - March 31 (testing, code reviews)

• Members: Neil. With help from Leonie and Elena on bug fixes and code reviews.

• Problems: Some problems we ran into were that it took us some time to understand the hierarchy of all the objects in the application. A lot of the objects are interrelated so a lot of refactoring is required. (for instance, when a user is deleted, he should also be deleted from any groups or conversations that he was belonged to).

• What I learnt: I learnt the importance of understanding the data models interacting with each other. I learnt that drawing out UML diagrams help with ensuring I cover all the edge cases while designing new objects or modifying the existing design.

• Tools Used: I used online UML tools, and generally had to go over foundational Java concepts.

Delete Conversation/Topic

• Description: This provides users the ability to delete conversations/topics that they own on the application. The purpose of this feature was to clear irrelevant topics. We envision that a lot of users will be using the app and generating topics of conversation. In order to ensure that users can have a smooth experience, it's important that the app is not cluttered with old or expired

topics that are no longer relevant.
• Development timeline: April 1 - April 4 (design and planning), April 4 - April 10 (development), April 11 - April 14 (testing, code reviews)
• Members: Neil. With help from Leonie and Elena on bug fixes and code reviews.
• Problems: Similar to the delete user feature, some problems we ran into were that it took us some time to understand the hierarchy of all the objects in the application. A lot of the objects are interrelated so a lot of refactoring is required. (for instance, when a topic is deleted, it should be removed from the city object and all users belonging to that conversation shouldn't be connected to it after it's deleted).
• What I learnt: I learnt the importance of understanding the data models interacting with each other. I learnt that drawing out UML diagrams help with ensuring I cover all the edge cases while designing new objects or modifying the existing design.
• Tools Used: I used online UML tools, and generally had to go over foundational Java concepts.

Nicknames
• Description: This enables users to choose nicknames. This is particularly relevant for those users that may want to remain anonymous online.
• Development timeline: March 18 - March 20 (design and planning), March 21 - March 26 (development), March 26 - April 1 (testing, code reviews)
• Members: Anh Duong (Elena)
• Problems: The first challenge was to understand the code base. I spent a lot of time trying to understand how to add new users and followed the same flow to add nicknames.
• What I learnt: How to dive into a large code base
• Tools Used: Debugging tool in Eclipse

Handle Duplicate Usernames/Topics
• Description: This ensures that no two users have the same usernames and no two topics in a city have the same titles. This ensures unique usernames and topic titles so Travel Chat is neat and organized.
• Development timeline: March 18 - March 20 (design and planning), April 1st - April 8th (development, testing and code reviews)
• Members: Anh Duong (Elena)
• Problems: The next challenge is to understand how the MVC architecture work in the code base. Whenever a new user or conversation is created, I had to connect to the server's model to check whether its name already existed.
• What I learnt: MVC architecture
• Tools Used: N/A

Password-based Sign-in
• Description: This enables users to choose a password for their accounts. Again, this is a

fundamental feature that is present in almost any chat application. This feature helps with protection of user data and privacy. Password is encrypted in the server side and safely stored in the database

• Development timeline: April 8th - May 1st.

• Members: Anh Duong (Elena)

• Problems: I had no experience in password encryption and didn't know how to store passwords safely.

• What I learnt: Password encryption and password storage scheme.

• Tools Used: readPassword method in java.io.Console, java.security library and its MD5 algorithm.

Data Persistence and Google Cloud Big Table Integration

• Description: This was a major goal of our project. Data persistence is important in any real-world functional chat application. We are using Google's NoSQL schemaless database Bigtable to implement this.

• Development timeline: May 15 - May 18 (design and planning), May 18 - June 1 (development), June 1 - June 2 (testing, code reviews).

• Members: Anh Duong (Elena)

• Problems: This is the first time I worked with Google Cloud Platform. I had lots of difficulties in learning how to use the platform, and deciding the proper schema to use in the models our team was working with. It was also my first time to convert an existing project to a maven project in order to run Google Bigtable.

• What I learnt: Maven projects, how to connect to Google Cloud Bigtable

• Tools Used: Maven, Google Cloud Bigtable