Project Proposal

Team Members: Zizhen Yu, Zhenyu Zhao, Amy Zhang, Kelly Yu, Shirley Yu, David Yang, Borun Xu, Jeremy Yiu

Email Addresses: zizhenyu@usc.edu, zzhao767@usc.edu, amyz@usc.edu, ksyu@usc.edu, yushirle@usc.edu, dyang886@usc.edu, borunxu@usc.edu, jeremyy@usc.edu

Weekly Meeting Time: Saturdays at 12 pm

Concept:

Our project aims to create a web application platform for students to connect with potential study partners and find study groups based on interests and availability.

Specifically, the platform will allow the users to see others who are looking for a study partner. The users will decide what information they put up about themselves on their profile, including their academic status, work habits, current courses, and availability. The system will use users' current courses to "recommend" study partners to them to make the finding process more efficient. Using this information, other users may decide whether or not they wish to communicate with this student.

The web application will also support a chat system to facilitate communication between students and study group formation. Users will only be able to chat with each other once both users demonstrate interest in communicating with each other.

High-Level Requirements

We hope to create a web application that allows students to communicate with each other and form/find study partners, similar to how Tinder and other dating apps help match people up. The application will allow for both registered users and unregistered users to use our services. Registered users are allowed to make their own profiles, view other user profiles, request and select study partners, and chat with their partners; registered users will also be able to create, join, and search for private or public study group rooms. A user can join a private study room with the correct password, whereas public study rooms are not password-protected.

Registered users may create accounts to store their user information and save their study partners. Regarding their user information, they may disclose information about their academic status (including year, classes, grades, etc…) for other students to see, input what kind of study partner they are seeking (short-term cram buddy, long-term course partner…), and choose whether to show a list of their current study partners. Users will have a profile where they can choose to display the relevant information that they have inputted. This profile will be able to be viewed by other users who are looking for study partners.

The application will recommend study partners to the user based on their inputted information, where the user can choose to request to be a study partner. Once the user's choice of study partner chooses them back, the user will be notified, and the application will officially pair them together as study partners. The application will also facilitate collaboration between paired students by offering chat features.

The application will also allow private group chats, which can be joined by anyone with the correct password, including guest users. Only registered users should be able to create private group chats.

Students without an account (guest users) will still have access to the web application, but the experience will be different compared to registered users. Guest users will not be paired up with a study partner; rather, they will be able to create, search, and join public study groups (which can include registered users and/or other guest users) that are currently online, and these study groups will communicate through chat as well. Guest users will be able to search for and join these public study groups

based on subject/course. Although guest users can only create public study group rooms, they will be able to join private study rooms created by registered users if they enter the correct password. When they join the chat, they can enter a temporary nickname for easier communication. Thus, guest users will still be able to study with others, but they will not receive the benefits of personalized matching and private study sessions with a study partner if they don’t register for an account.

Technical Specifications

Web Interface (15 hours)

Main Menu/Home page - A page that redirects to other pages containing actual features listed below. It will include links/buttons for each. There will be a login, sign up, and a dropdown menu with all the other features on a “header”; once the user is logged in, there will be a profile and logout option replacing login and sign up. The page will feature links to the study hall, discover page, and friends page.

Login Page - User input login information to authenticate. A simple page with a text input for the username and a password input for the password should send a secure request to the server requesting authentication. If successful, it should return a token that will be stored in a cookie to be used for future transactions.

Sign Up Page - Webpage with the form to input information to register for an account. The page would show text fields prompting users to input necessary account information including first name, last name, preferred name, username, GPA, and year; include dropdowns type of connection sought that prompt users to choose one or more provided options. Once a user finishes their input, there is a registration button that saves and submits such account information. A verification code will be sent to the user's email to confirm the legitimacy of the user's registration.

Study Hall - A page with all available public chat rooms and a search feature. The public chat rooms will be displayed in a list with information like chatroom name, chatroom capacity, current user count, and chatroom finder, as well as a button that will allow the user to join if the chatroom isn’t at capacity. There is also a side menu that will allow the user to input a password to join private rooms.

Chatroom Page - Interface for messaging on the platform. A simple messenger-like page with messages from the user on the right side and messages by the others from the left. Each message should indicate the content, speaker, and timestamp of the messages. The page does not keep a chat history log, meaning the chat is lost once the user closes or refreshes the page.

*Create Room Page - Webpage with the form to generate a new chat room. It should allow the user to input chatroom name, size, private/public, and select some friends they would like to invite. Once the user confirms the creation, it will respond with whether the chatroom is created successfully. If the chatroom is private, upon creation the user would receive a code used for other people to join. The number of chatrooms a user can create is rate-limited.

*Discover Page - The page users go to to discover potential new study partners. This is where students get to see other registered students and potentially match with similar interests. The page will display by default random registered students in the form of cards for visualization, and users could click on any of them to view the detailed information of their profile. Users could also search on this page for names, years, or other information on a profile to filter results. Each profile card should have the option to send a study partner request and view profiles in detail.

*Account Page - The page where the user may view/modify their user information. It contains names, courses, links to friends pages, and other information that the user can choose to add. It should have checkboxes for the boolean “seeking long-term study partner” and “seeking short-term study partner.” Will have an “edit profile” option that allows users to type in textboxes and enter new information to change. All changes will be reflected in the database.

*Friends Page - The page containing all previous study partners of the user. Can enter the chat room from this page. Friends and chatting rooms will be listed on this page. Each friend/chatting room will be a row or icon that the user can click on, the name should be displayed for friends, and the name and current user count should be displayed for chatting rooms. Users can click on it and then enter the chatting page. The user may also pull up a modal that contains the list of pending friend requests from other people. They may choose to accept or reject the friend request.

(Pages with a * in front are only available to registered users)

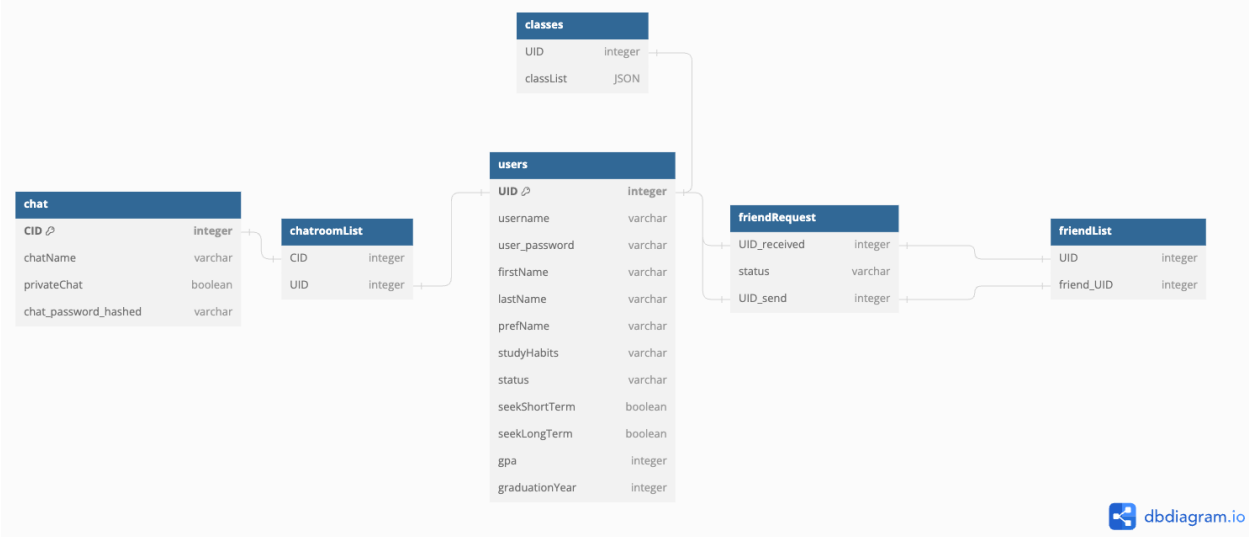
Back-end Server (15 hours)

Hosting source: To be decided.
Responsible for calling the web interface (hosting the html files).
Responsible for answering web queries from the interface requesting data.
    Including: getAllChatRooms, getChatRoom, getFriendRec, getFriends, login…
Responsible for storing user data.
Responsible for hosting email-sending service.
Responsible for updating the database upon receiving user forms.
    Including: register, friendRequest, updateUserInfo…
Responsible for hosting the “chatroom” threads.

Database (6 hours)

Store user data:
    Int UID
    String username, password.
    String firstName, lastName, prefName, status
        Status: a short user-inputted self-description displayed on their profile.
    String[] classes, friendRequests
    Boolean seekShortTerm, seekLongTerm
    Double gpa
    Int graduationYear
    Int[] friendList: contains UID of friends
Store chat data:
    Int CID
    User[] members
    String chatName
    Boolean privateChat
    String Password (hashed)
    *Entries are removed when the chatroom is empty (everyone left)

Detailed Design Document

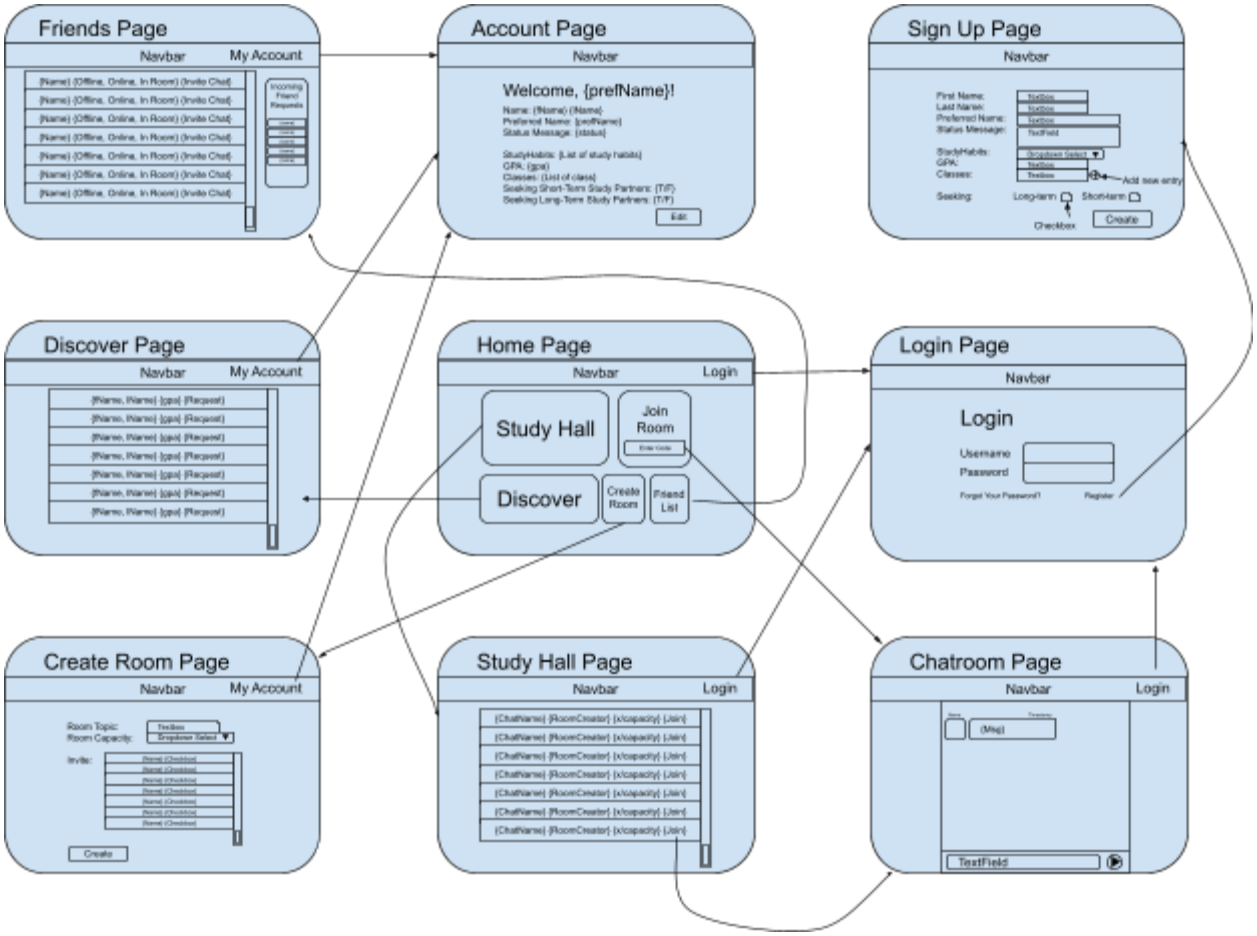


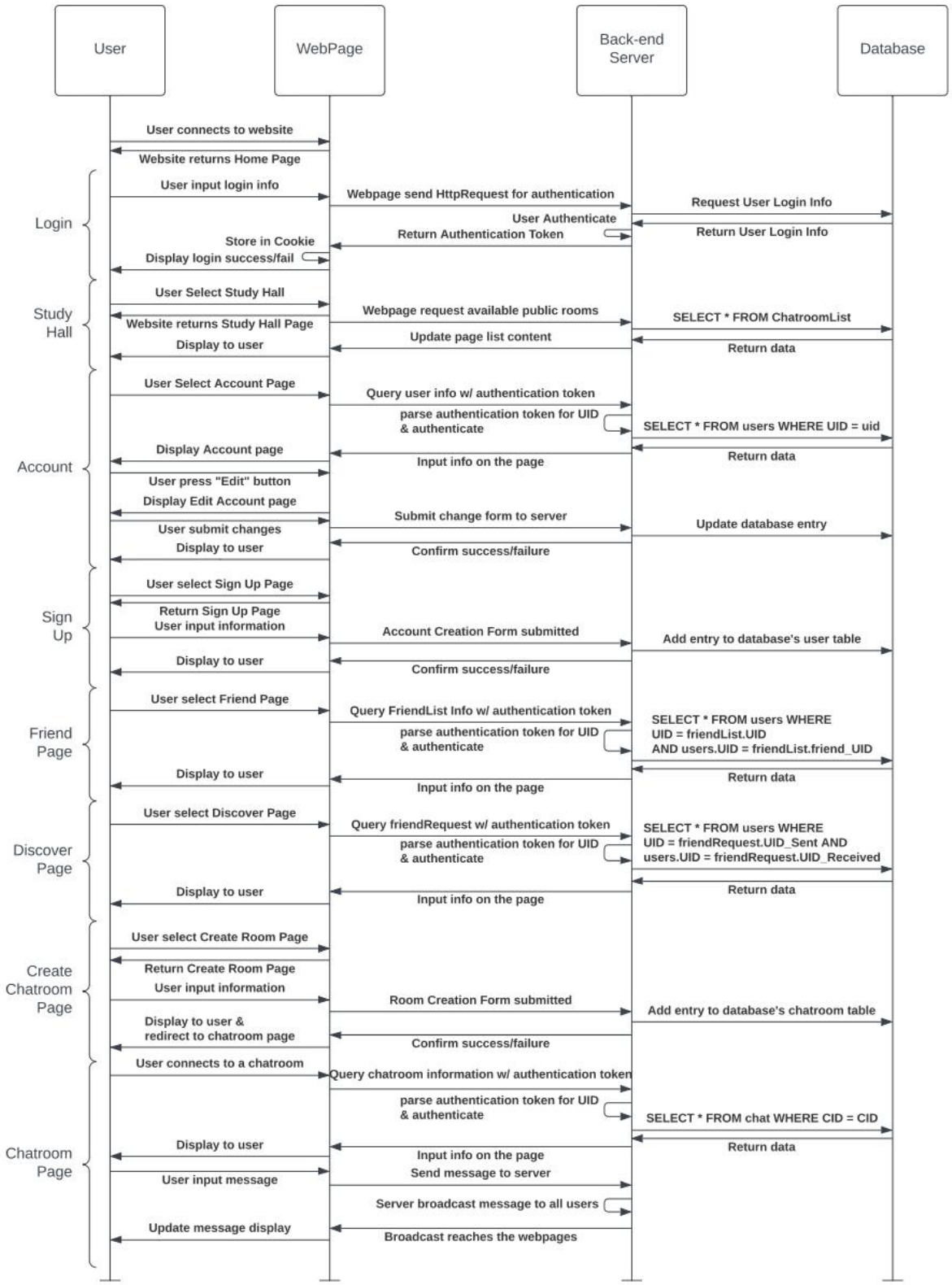
Hardware Requirements

- A computer complete with RAM and CPUs.
- A stable internet connection.

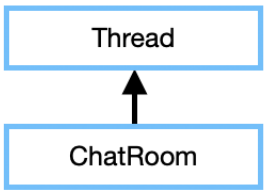
Software Requirements

- The latest version of any browser (Google Chrome, Firefox, Safari…)
- Permission to store cookies.


Friends Page
Navbar
My Account
Account Page
Navbar
Welcome, {prefName}!
Sign Up Page
Navbar
Discover Page
Navbar
My Account
Home Page
Navbar
Login
Study Hall
Join Room
Discover
Create Room
Friend List
Login Page
Navbar
Login
Username
Password
Create Room Page
Navbar
My Account
Study Hall Page
Navbar
Login
Chatroom Page
Navbar
Login


User
WebPage
Back-end Server
Database
User connects to website
Website returns Home Page
Login
User input login info
Webpage send HttpRequest for authentication
Request User Login Info
User Authenticate
Return User Login Info
Return Authentication Token
Store in Cookie
Display login success/fail
Study Hall
User Select Study Hall
Webpage request available public rooms
SELECT * FROM ChatroomList
Website returns Study Hall Page
Update page list content
Return data
Display to user
Account
User Select Account Page
Query user info w/ authentication token
parse authentication token for UID & authenticate
SELECT * FROM users WHERE UID = uid
Return data
Display Account page
Input info on the page
User press "Edit" button
Display Edit Account page
Submit change form to server
Update database entry
User submit changes
Confirm success/failure
Display to user
Sign Up
User select Sign Up Page
Return Sign Up Page
User input information
Account Creation Form submitted
Add entry to database's user table
Display to user
Confirm success/failure
Friend Page
User select Friend Page
Query FriendList Info w/ authentication token
parse authentication token for UID & authenticate
SELECT * FROM users WHERE UID = friendList.UID AND users.UID = friendList.friend_UID
Return data
Display to user
Input info on the page
Discover Page
User select Discover Page
Query friendRequest w/ authentication token
parse authentication token for UID & authenticate
SELECT * FROM users WHERE UID = friendRequest.UID_Sent AND users.UID = friendRequest.UID_Received
Return data
Display to user
Input info on the page
Create Chatroom Page
User select Create Room Page
Return Create Room Page
User input information
Room Creation Form submitted
Add entry to database's chatroom table
Display to user & redirect to chatroom page
Confirm success/failure
Chatroom Page
User connects to a chatroom
Query chatroom information w/ authentication token
parse authentication token for UID & authenticate
SELECT * FROM chat WHERE CID = CID
Return data
Display to user
Input info on the page
User input message
Send message to server
Server broadcast message to all users
Update message display
Broadcast reaches the webpages

Account	ChatRoom	ChatMember	Lists
First Name Last Name Preferred Name Status Message Online Status Study Habits GPA Classes Short-Term Partner Long-Term Partner Friends List Incoming Friend Requests List Outgoing Friend Requests List In Room Status Password	Room Topic Room Capacity # of Users in Room Members in Room Possible Invitees All Invitees	Name Account	Discover List Friends List Study Hall List
register() deregister() login() editInfo() getInfo() sendFriendRequest() acceptFriendRequest() rejectFriendRequest()	create() delete() join() broadcast()	sendMessage()	getDiscoverList() getFriendsList() getStudyHallList()



Testing Plan

Testing General feature

Test Case (White Box Test): General Display - Making sure the pages load as intended based on the Detailed Design.

Test Case (White Box Test): General Redirect - Making sure the page redirects to the correct places.

Testing Chatroom feature

Test Case (White Box Test): Incorrect format handling - Testing how the program handles incorrect input formats or unauthorized access attempts.

Test Case (White Box Tests): Data transmission of chats - Ensure chats are stored and retrieved correctly to make sure the chatting function is working properly.

Test Case (Black Box Tests): Messaging - Check message sending/receiving among participants in the chatroom, test cases like chatting between registered and guest users.

Test Case (Black Box Tests): Chatroom functions: Attempt to create, search, join, and interact within study groups by both registered and guest users.

Test Case (Stress Test): Message handling - make sure no errors occur when a message is really long and make sure the program handles messages from a wide range of characters like those in other languages.

Test Case (Stress Test): Chatroom capacity - Make sure the chatroom handles connections from multiple clients and a stress test could involve making hundreds of connections.

Testing Login feature

Test Case (White Box Test): Invalid account - should return “Account not found”

Test Case (White Box Test): Valid account, incorrect password - should return “Incorrect password”

Test Case (White Box Test): Valid account, correct password - should log in appropriately.

Testing Page Loading feature (Study hall, Friends, Discover)

Test Case (White Box Test): Nothing to load - The database query returns an empty list.

Test Case (White Box Test): Loading 1 page - The database query returns a list of items that all fit on one page. (A page should fit around 20 items) Checking the data is displayed correctly.

Test Case (White Box Test): Loading Multiple - The database query returns a list of items that would have to be split into multiple pages of results. Checking that the data is being split up into pages correctly.

Testing Sign Up feature

Test Case (White Box Test): Input type mismatch (ex. number instead of string for First Name) - should return “Invalid input”

Test Case (White Box Test): Missing input for First Name, Last Name, and Preferred Name - should return “Missing at least one of the required fields: Username, Password, First Name and Last Name or Preferred Name”

Test Case (White Box Test): Missing input for Username, Password, or both - should return “Missing at least one of the required fields: Username, Password, First Name and Last Name or Preferred Name”

Test Case (White Box Test): Missing input for First Name and Last Name, valid input for Preferred Name, Username, and Password - should sign up appropriately

Test Case (White Box Test): Missing input for Preferred Name, valid input for First and Last Name, Username, and Password - should sign up appropriately

Test Case (White Box Test): Username already in use - should return “Please enter a different username”

Testing Authentication feature

Test Case (White Box Testing): Regular Authentication - Checking the requests go through after the user logs in correctly.

Test Case (White Box Testing): Faulty Authentication - Checking the requests is blocked when the user doesn’t log in.

Test Case (White Box Testing): Expired Authentication - Checking the requests is blocked when the cookie storing the authentication token expires.

Test Case (White Box Testing): Missing Authentication - Checking the requests is blocked when the cookie is manually deleted before its expiration.

Test Case (Stress Testing): Overloading Authentication - Checking if the server delays a request when the rate limit is reached to prevent overloading.

Testing Accounts feature

Test Case (White Box Testing): Retrieving Account Data - Making sure the account page is loading correctly with the right user info.

Test Case (White Box Testing): Updating Account Data - Making sure the modification done on the account page is going through to the database.

Test Case (White Box Testing): Retrieving Discovery Data - Making sure the algorithm is recommending people who fit what the user is looking for.

Test Case (White Box Testing): Friend Requests - Making sure the friend invites are received/accepted correctly. (So the users would see incoming requests and once they accept the friends would be on their friend list.)

Test Case (White Box Testing): Chatroom Profile - Making sure the chatroom is displaying the preferred name correctly.

Testing database feature (unit testing)

Test Case (White Box Testing): Database structure creation - Making sure SQL script successfully creates schema and tables

Test Case (White Box Testing): Entry insertion - Making sure entries can be added to tables, and that relationships are correctly established

Test Case (White Box Testing): Query testing - Making sure database query statements work as intended

Deployment Plan

1. Host the web pages on GitHub
    a. Go into the GitHub Repository > Settings > Pages
    b. Publish the website through GitHub.
    c. Set the default page to the home page.
    d. Browse through the different pages to ensure deployment.
2. Push the back-end server on AWS and have it begin listening for connection requests.
3. Set up the database based on the schema of MySQL workbench and connect to the server using JDBC on the server's end.
4. Update the corresponding IP addresses on the webpage for communication with the server.
5. Run SQL scripts to ensure all data was migrated correctly.
6. Perform regression testing to ensure nothing is broken.
7. Notify the users of the updated application and provide them with the URL.