1. Introduction

This document outlines the fully implemented and unit/integration tested user stories for our StudentHUB project to date. It details each story that includes GUI functionality, identifies the main developers responsible for each story, and shows the distribution of testing responsibilities among team members. Additionally, the document includes bug reports and code review findings related to the user stories.

2. Updated User Story Assignments & Descriptions

2.1 Course Calendar

Assigned To: Abdulkadir Ahmed
Status: Complete
Expected ITR3 Enhancements:

- **UI/UX Improvements:** Better spacing and a refined 24-hour calendar display.
- **Backend Enhancements:** Smoother database interactions for course management.
- **Bug Fixes & Testing:** Continued testing to ensure reliability and responsiveness.

2.2 User Accounts (New)

Assigned To: Abdulkadir Ahmed
Expected Completion: March 25
Scope and Features:

Registration:

- The user provides a username, password, year of study, and program.
- Frontend validations enforce that:
  - **Username:** Must be more than 6 characters and less than 14 characters.
  - **Password:** Must be exactly 6 characters.
  - **Year of Study:** Must be a number between 1 and 20.
- The backend verifies that usernames are unique and stores the user's details in our Render-hosted PostgreSQL database.

Login:

- Users can log in with their username and password.
- Upon successful login, the app uses AsyncStorage for auto-login, displaying the user's account details.

Account Management:

- The logged-in view displays user details (username, year of study, and program) and provides options to "Logout" or "Delete Account."
- "Delete Account" sends a DELETE request to remove the user from the database and logs them out.

Auto-Login:

- The system retains the login state across app restarts, so users remain logged in until they explicitly log out or delete their account.

2.3 Group Chat (Without User Accounts)

Assigned To: Simmonly & Digvijay
Notes:

- A messaging group chat where you input messages and the messages appear in the group chat
- This feature allows students to participate in a shared group chat without the need for login or registration. All messages are sent using a placeholder user identity (e.g., "Guest") and stored in the project's backend PostgreSQL database through dedicated API routes.

Implementation Details:

- The backend exposes endpoints for posting and fetching chat messages.
- POST /api/chat/messages saves a message.
- GET /api/chat/messages retrieves all stored messages.
- The frontend (React Native) fetches and displays messages with sender info and timestamps.
- UI includes:
- Scrollable message list.
- Auto-scroll to the latest message on new send or keyboard show.
- Input bar with validation to prevent empty messages.
- Messages include static metadata like sender name, year, and program (currently hardcoded).

Limitations:

- All messages appear from a generic "Guest" user.
- No identity persistence across sessions.
- No authentication or real-time delivery (polling or refresh-based only.

Expected Enhancements:

- Once the User Accounts feature is fully integrated (see 2.4), the chat will fetch the actual logged-in user's name, year, and program.
- Backend and frontend will be updated to dynamically map chat messages to authenticated users.

Assigned To: Simmonly & Digvijay
Expected Completion: March 28
Notes:

- Once the User Accounts feature is finalized, integration into the Group Chat will be implemented to enable personalized messaging.

2.5 AI Advising Chatbot

Assigned To: Fardad Rashidian
Status: Ongoing
Expected Completion: March 31
Scope:

- **Backend:** Processes user PDFs and summarizes content using GPT-4.

- **Frontend Integration:** The chatbot is being integrated into the UI with ongoing improvements to the user experience.

---

3. Testing Assignment & End-to-End Manual Test Cases

Each team member is assigned user stories that they were not heavily involved in implementing to ensure independent testing. Below are sample test cases for the **User Accounts** story (note that the Course Calendar also has a comprehensive set of frontend tests, as documented in its test suite).

User Accounts – End-to-End Test Cases

Test Case 1: Messages are sent

- **Objective:** Verify that when you put a message as input and click sent you see the message appear in the group chat

- **Steps:**

    1. Open the website

    2. Navigate to the group chat

    3. Enter a text message in the input box

    4. Click send

    5. **Expected Result:** A message appears in the group chat that has the same content as that which was put in the input box

Test Case 2: Messages only exist in a single session

- **Objective:** Ensure that when you open the tab after it's been closed the messages that existed before don't exist anymore

- **Steps:**

    1. Repeat the steps for test case 1 of adding messages to the group chat

    2. Close the tab

3. Reopen the tab and navigate to the group chat tab
4. **Expected Result:** There are no messages in the group chat

Note: Many more test cases would be added once the user accounts are implemented (expected 10 test cases)

4. Bug Reporting & Code Review Summary

Bug Reporting

- **Bug 1:** Duplicate username error message appears only briefly; recommendation to improve the visibility and duration of error alerts.
- **Bug 2:** On some devices, the auto-login persistence does not refresh immediately upon app restart.
  (All bugs are reported on the team's GitHub repository with steps to reproduce, observed behavior, and expected behavior.)

Code Review & Code Smells

- **Security Concern:** Storing passwords as plain text is a major risk.
  - *Recommendation:* Implement password hashing (e.g., using bcrypt).

- **Repeated Validation Logic:** Registration validations appear in both the frontend and backend.
    - *Recommendation:* Abstract the validation logic into reusable functions to reduce duplication.
- **UI Feedback:** Error messages in the frontend are sometimes unclear or too brief.
    - *Recommendation:* Improve error messaging for better user guidance.
- **General Design:** Some functions, especially in the registration endpoint, are lengthy and could be refactored for clarity.
    - *Recommendation:* Refactor lengthy methods and enhance code modularity.

---

5. Conclusion

This document serves as the comprehensive assignment deliverable for our team. It outlines our current user stories with clear implementation details and testing assignments, alongside documented bugs and code review findings. All team members are responsible for updating this document on our team's GitHub repository as further issues and improvements are identified.