# Final Report: Study Sync Pro

**Group number:** 04

**Github URL:** https://github.com/JustinSui624/Study-Sync-Pro

**Team Members:** Justin Sui, Yeshua Colon, Luis Borruel, Mason Bush

## Table of Contents:

## Section 1: Project Description, How solution addressed challenged statement, Features and Functionality, System Models (Architectural pattern, system context model and use case model)

***Project Description**:*

A desktop application that joins hardworking university students together who are working on similar assignments and courses, enabling them to form intuitive study groups and collaborate effectively to develop, adapt, and overcome educational challenges.

***How Solution Addresses Challenge Statement**:*

Study Sync Pro addresses the challenge of students struggling with similar assignments or courses by providing a peer-to-peer connection platform. Our app bridges the gap between disconnected students and collaborative learning by offering a group-oriented solution designed to foster academic support and teamwork

***Features and Functionality**:*

- Peer-to-peer matching
  - Matches students based on subject area, university, or assignments
  - Priorities forming study groups with others taking the same or similar courses
- Group Creation and Management
  - Allows users to create and join study groups
  - Groups are organized by subject/topic
- Intuitive Front-End Design
  - Simple and clean UI with accessibility
  - Responsive layout for desktop

***System Models**:*

***Architectural Pattern***

- Model
  - Handles data-related logic
  - Interfaces with the MySQL database to store and retrieve user profiles, groups, and membership data
- View
  - Front-end interface that users interact with
  - Built to be clean, intuitive, and accessible
  - Responds to user actions such as logging in, creating groups, or joining study sessions

***System Context Model***

This section displays how Study Sync Pro fits into its environment and interacts properly

- Actors:
  - Students (Users): Create accounts, join groups, and interact with peers
  - Database system (MySQL): Stores persistent data such as users, group info, courses
  - API layer: Facilitates communication between front-end and back-end

Context Diagram (Text-Based):

```
============================================================
|                          [Students]                          |
|                              |                              |
|                              v                              |
[Study Sync Pro Front-End] <--> [Web Server/API] <--> [MySQL Database]       |
|                                                              |
============================================================
```

- Students interact via the UI
- Requests are sent
- Processes logic, queries database, and returns results

***Use Case Model***

Describes the key interactions users will have with the system. Below are the main use cases for Study Sync Pro:

| **Use Case** | **Description** |
|---|---|
| **Register Account** | Student signs up with email and academic info. |
| **Log In** | User securely accesses their profile. |
| **Create Group** | User creates a new study group with topic, description, and availability. |
| **Search Groups** | User browses existing study groups by subject or course. |
| **Join Group** | User joins a selected group and gains access to peer discussions. |
| **Matchmaking Logic** | Behind-the-scenes functionality suggests similar courses. |

## **Section 2**: Code Management, Project Management, and Test Plan

Efficient and organized code management was essential to the success of the Study Sync Pro project. We used Git and GitHub as our version control system and collaboration platform.

***Repository Structure****:*
The main project code is located in the /src directory of the GitHub repository. This includes React component files, assets, and Firebase configuration scripts.

***Commit Workflow:***

- Each team member committed changes directly to the main branch.

- Coordination between team members was essential to avoid overwriting each other's work, so we communicated regularly before pushing updates to GitHub, via discord.

***Project Management***

Project management was handled through direct communication and shared planning documents.

***Task Coordination****:*

- Team members used a shared Google Doc to outline weekly goals and task assignments such as the daily scrum.

- Regular check-ins (virtually or in person) ensured everyone was aligned with deadlines and responsibilities.

- Responsibilities were distributed by feature (e.g., login screen, task tracker, timer, etc.).

***Timeline:***

- Week 1: Defined project scope, created wireframes, set up database and React environment

- Week 2: Developed User Interface

- Week 3: Implemented task management functionality and dashboard layout

- Week 4: Refined styling, added calendar/timer features, and tested all components
- Week 5: Bug fixes, documentation, final deployment

***Team Roles:***

- Frontend Development: Luis Borruel – UI components, layout
- Backend Integration: Justin Sui – Database configuration, authentication
- Frontend/backend: Yeshua Colon – Database to UI integration, routing
- Testing & Debugging: Mason Bush – Functional testing and bug tracking

***Test Plan***

Testing was conducted continuously throughout development to ensure reliability and usability.

***Manual Testing***

All core features were tested manually by each team member, including:

- ***Authentication Flows*:**
    - Sign-up, login, and logout actions
    - Error handling for incorrect credentials
    - Session persistence after page reload
- ***Task & Study Management*:**
    - Add/edit/delete task functionality
    - Timer and countdown components
    - UI responsiveness on different devices
- ***Navigation*:**
    - Checked routing between pages such as login → dashboard

  - Verified protected routes for logged-in users only

- ***Database Testing:***
  - Verified security rules by attempting unauthorized read/write operations
  - Ensured users could only access their own data

## **Section 3**: Technical Details, Installation Instructions, Login and Access Credentials and API Keys

### ***Technical Details***

- Application Name: Study Sync Pro
- Purpose: Connect students working on similar assignments/problems
- Type: Desktop application with database integration
- Architecture: Object-oriented C++ with SFML graphics

### ***Technology Stack***

- Language: C++ (C++17 standard)
- Graphics Framework: SFML 2.5.1 (Simple and Fast Multimedia Library)
- Database: PostgreSQL (Supabase hosted)
- Build System: Make (MinGW-w64 on Windows)

### ***Libraries and Dependencies***

- SFML Components:
  - sfml-graphics (UI rendering)
  - sfml-window (Window management)
  - sfml-system (System utilities)
- Database Connector: PostgreSQL C library

### *Application Features*

User Profile Management

- Full name and graduation year
- Subject/course selection (multi-select)
- Profile editing capabilities

1. Study Group Matching

- Algorithm matches users based on:
    - Common subjects/courses
    - Graduation year proximity
- Real-time group recommendations

User Interface

- Pages: Login, Registration, Profile, Group Matching

### *Installation Instructions*

1. Install SFML:
2. Download SFML 2.5.1 for your compiler (MinGW)
3. Extract to C:\SFML-2.5.1

- If installed elsewhere, update the path in makefile

4. Ensure mingw32 is downloaded to C drive on your personal compute
5. Download the zip via the Github

- Github: [https://github.com/JustinSui624/Study-Sync-Pro.git](https://github.com/JustinSui624/Study-Sync-Pro.git)

6. Open .zip
7. Extract .zip to desired location
8. Open newly extracted .zip folder and locate "required_libraries"
9. Extract "required_libraries" to main folder of "Study Sync Pro" alongside other folders such as src, fonts, exactly\
10. Open terminal, (right click inside file explorer and select "Open in Terminal")
11. While in "\Study-Sync-Pro-main>" folder, run: mingw32-make

- This command will build the project

12. After waiting for the command to finish processing, type: .\bin\StudySyncPro.exe
13. Now, enjoy the application! :)

### *Login and Access*

The application connects to a Supabase-hosted PostgreSQL database at db.iekosjtwireodvbaqhcm.supabase.co:5432 using the connection string with username "postgres" and password "cen3031group4". A fallback test account (username: "test", password:

"test") provides access when the database is unavailable. The current implementation stores passwords in plain text. Users can register new accounts through the application interface, with all data synchronized to the cloud database for cross-session persistence.

## **Section 4**: Risk Management Plan, Software Quality Attributes and explanations

### ***Risk Management Plan***

Risk management is critical in software development to anticipate potential problems and prepare mitigation strategies. The following table outlines key risks that were identified during the development of Study Sync Pro, along with their impact and response strategies.

| **Risk ID** | **Risk Description** | **Category** | **Likelihood** | **Impact** |
|---|---|---|---|---|
| R1 | Security vulnerabilities in user data handling | Technical | Medium | High |
| R2 | Poor user adoption or engagement | Business | High | Medium |
| R3 | Database connection or failure issues | Technical | Medium | High |
| R4 | Incomplete or late implementation of features | Project | Medium | Medium |
| R5 | Accessibility or UI issues for users with disabilities | Usability | Medium | Medium |
| R6 | Input validation errors leading to app crashes | Technical | Medium | High |
| R7 | Miscommunication between team members | Team/Process | Medium | Medium |
| R8 | Lack of scalability for larger user bases | Technical | Low | Medium |

### ***Software Quality Attributes***

We designed Study Sync Pro with a focus on key software quality attributes to ensure it was not only functional but also reliable, secure, and user-friendly.

### *Usability*

- **Simple and intuitive UI:** Clean design, minimal clicks to navigate core features.
- **Responsive layout:** Optimized for both desktop and mobile users using responsive CSS.
- **Accessibility:** Font sizes, color contrasts, and layout spacing were considered to improve readability and user experience.

### *Reliability*

- **Consistent performance:** All core features, task creation, login/logout, and real-time data sync, were tested multiple times under different conditions.
- **Crash-free:** The application was stable during all manual test sessions, with error boundaries in place for unexpected failures.

### *Security*

- **Authentication-based access:** Only registered users can access the dashboard and create or modify study tasks.
- **Firebase security rules:** Prevent users from accessing or editing other users' data.
- **No sensitive information** such as API secrets is exposed in the codebase.

### *Performance*

- **Optimized data handling:** Firestore's real-time syncing allows fast loading and updating of task lists.
- **Lightweight frontend:** React ensures fast component rendering and smooth transitions.

### *Maintainability*

- **Modular file structure:** Components are broken down by function Login, TaskList, Dashboard, making the code easy to update or expand.

- **Consistent naming and formatting:** Improves readability for future developers or contributors.

- **Comments and documentation:** Key functions and logic are well-commented to aid in understanding and onboarding.

***Portability***

- **Web-based access:** No installation needed, users can access the app from any browser without dependencies or system requirements.