Project Report

Project Video Link:
https://drive.google.com/file/d/1HZJDDS9FeUJCKxGv8iTBwsXhCAqDKovX/view?usp=drive_link

1. **Project Direction Changes**

Throughout the project creation process, our team encountered several challenges that led to changes in the direction of our original proposal. While we maintained the core purpose of the project—helping students find and join study sessions—we adapted several of our original ideas to better fit technical limitations and usability needs.

Originally, for the Geolocation Tracking Visualization, we proposed displaying circles over building regions on the map that would vary in color based on the number of study sessions held there. However, after integrating the Google Maps API, we realized that creating color-based clustering while keeping the map interactive was significantly more complicated than anticipated. Instead, we leveraged the natural variance in longitude and latitude captured by user-submitted pins. This allowed users to visually observe areas of high activity without getting rid of the interactivity of individual session pins. Users could still easily differentiate between nearby sessions and click on them for more details, making the interface more intuitive.

Another major pivot involved the study session planner. Initially, we envisioned a separate planner page where users could schedule and organize future study sessions. However, during development for Stage 3, we built a prototype centered around a map interface with buttons to add sessions directly. For Stage 4, we decided to maintain and streamline this design instead of adding a more complex planner. We enhanced the navbar with new buttons that allowed users to quickly add or remove sessions in real-time, keeping the experience simple and user-focused.

Finally, we decided to drop the chat feature from the original proposal. We determined that the core goal of StudyLync was to facilitate in-person study meetups. Since users would physically gather at the designated session locations, live chat functionality became less critical. Removing it allowed us to focus more on real-time map updates, session management, and search features, all of which contributed more directly to the app's usability and success.

Instead, we chose to introduce a Review feature, supported by a new Reviews table in the database. Since one of our long-term goals is to eventually make StudyLync a publicly available application, having a review system allows us to gather direct feedback from users about their

experience, helping to guide future improvements. This shift allowed us to focus on collecting actionable insights while keeping the application lightweight and easy to use.

2. **Achievements and Failures**

Achievements

Despite the changes, our final application successfully met many of the goals outlined in the original proposal:

- Functional, real-time map: We created a live map showing all available study sessions, which updates automatically when sessions are added or removed.
- User authentication and permissions: Only authenticated users can join study sessions. Furthermore, only users who joined a session are authorized to delete it, adding a layer of security and accountability.
- Keyword course search: Users can search for study sessions based on course keywords. As they type, pins on the map dynamically filter to show only the relevant sessions.
- Session discovery before creation: Before adding a new session, users can easily check if one already exists for their course, encouraging group formation rather than duplication.
- Real-time refreshing: The application includes a manual refresh option so users can ensure their map view stays up-to-date with any newly added sessions.
- Initial feedback system: We gathered preliminary reviews to better understand how the app could continue to evolve and improve.

Failures

However, we were not able to fully build out some of the features:

- Dropping the chat feature: Although a chat function would have provided an additional communication channel, we believe its absence did not detract significantly from the app's purpose. Users meeting in person at the agreed location fulfills the main goal of facilitating collaboration.
- Advanced visualization: A more sophisticated clustering system would have made it easier to see how many sessions were in each area. However, given the density of sessions and user experience considerations, the current solution of individual clickable pins proved to be a more user-friendly alternative.
- Scheduled study session planner: This feature, which would have allowed users to plan sessions for future times more easily, was ultimately dropped. While it could have streamlined the planning process, focusing on real-time session creation kept the platform simpler and faster for users to use.

3. **Schema / Data Source Changes**

We made several important modifications to our original database schema throughout the development of our application, based on both practical needs during implementation and logical inconsistencies we discovered in our initial design.

First, we added a CourseName attribute to the Courses table. When downloading course data from UIUC's Course Explorer API, we noticed that the CourseTitle field only contained the course number (e.g., "CS374") rather than the full name of the class. To make the map search feature and session listings more informative and user-friendly, we introduced a separate CourseName attribute, which stores the full course title (e.g., "Introduction to Algorithms & Models of Computation"). This change ensured that users could easily recognize and search for classes without ambiguity.

Second, we added a SessionId foreign key to the Users table. This allowed us to track which study session a user had joined and, more importantly, enabled us to implement advanced SQL queries (such as counting the number of users in each session) during Stage 3. It also provided a clean and efficient way to enforce the rule that a user can only be part of one study session at a time, simplifying both our backend logic and frontend checks.

Additionally, we encountered a critical issue in our original relational design. Initially, the relationship between Users and StudySessions was described inconsistently — it was labeled both as a many-to-many relationship (allowing users to join multiple sessions) and as a many-to-one relationship (where each user is part of only one session) across different parts of the schema and project description. To resolve this, we removed the UserSessions join table and instead relied solely on the SessionId foreign key within the Users table. This change enforced a many-to-one relationship, where each user is associated with exactly one active session at a time. This decision aligned much better with our application's intended use: encouraging students to join a single study group that matches the course they are currently studying.

We originally planned for a Chat table to support live messaging between users. However, we ultimately dropped this feature, focusing instead on meeting in-person as the core goal of the platform. Maintaining a chat feature would have unnecessarily complicated the backend without adding meaningful value to our use case. In place of the Chat feature, we introduced a Reviews table. The table allows users to submit a rating and optional comment about their study session experience. It includes: A primary key: ReviewId A foreign key relationship to SessionId (linking to StudySessions) and UserNetId (linking to Users) Additional fields for Rating, Comment, and CreatedAt.

4. **Diagram Changes**

Overall, the changes we made to our UML diagram and table implementations were minimal but important for clarity and consistency with our final database structure.

The main adjustments involved adding new attributes to existing tables, as described previously. Specifically, we added the CourseName attribute to the Courses table and the SessionId attribute to the Users table. These additions ensured that our diagram accurately reflected the actual structure and functionality of our database.

We also made a small but important formatting correction: we removed explicit foreign key labels from the UML diagram. Initially, in our early versions of the diagram, we labeled foreign keys directly on the connecting lines between entities. However, after reviewing standard UML conventions, we realized that foreign key relationships are implicitly represented through the associations (lines and cardinalities) between tables. Explicitly labeling them was unnecessary and potentially confusing, so we cleaned up the diagram to follow proper UML standards.

Additionally, we removed the Chat table from the UML diagram and added the new Reviews table instead. The Reviews table includes all attributes described earlier (ReviewId, UserNetId, SessionId, Rating, Comment, and CreatedAt), along with the appropriate primary and foreign key constraints. This change made the diagram better aligned with our final database schema and reflected the updated focus.

5. **Functionalities Added and Removed**

Functionalities Removed:

Originally, our project proposal included an integrated messaging system (chat) to allow communication between members of a study group. However, as development progressed, we realized that implementing a real-time chat system including setting up WebSocket connections, managing message storage, and ensuring scalability would introduce significant complexity within the time constraints of the project. Therefore, we decided to remove the Chat feature to prioritize delivering a polished core experience (study location map, session scheduling, and filtering).

Functionalities Added:

To make sure we still had a way to engage with users and receive feedback for future improvements, we added a Reviews form. This feature allows users to leave comments, suggestions, or feedback directly through the app. Our reasoning was that while we couldn't support real-time messaging between users, we could still build a two-way communication channel between users and developers, helping us iterate and grow StudyLync based on real user input.

6. **Advanced Database Project Complementation**

1. Find the top 3 courses with the highest session attendance

We are tracking attendance dynamically with this specific advanced queries — the number of people per session isn't static, and you can report live data to your users. It complements the purpose of StudyLync because it lets users choose sessions based on size (e.g., if they want a big group to cram together or a smaller quiet group).

2. Count the number of students in a study session at a location

This advanced query shows that you can analyze user engagement — not just build functionality, but also extract insights from the data your app collects. It complements the purpose of our project because knowing the most popular courses could help you prioritize features, optimize notifications, or even recommend study sessions for project updates or evolvements in the future.

7. Technical Challenges Encountered

Studysession Table Not Updating Properly - Sonika

- **Problem:** When we first implemented the ability to drop a pin and create a new study session, we encountered an issue where the session data was not properly being inserted into the database. Although the frontend would show a pin being dropped, no entry was actually created in the backend.
- **Cause:** The server-side API route was not correctly handling the incoming request or validating the required fields before insertion.
- **Advice:** Make sure that whenever you design a feature that creates a new database entry (like a pin drop), you validate all required fields on both the frontend and backend. Another tip is to also immediately fetch and verify that the new record actually exists after creation (for example by returning the newly created object and displaying success/failure to the user).

User Update Issues - Shrida

- **Problem**: Initially, when a user created a new session (by dropping a pin), their session_id field in the user table was not being updated automatically. They would have to manually join their own session to be linked to it. Additionally, newly signed-up users were not being inserted into the user table properly, and when sessions were deleted, the user's session ID was not being cleared back to null.
- **Cause:** We had inconsistent logic between session creation and user updates, and we lacked proper cleanup steps when sessions were deleted.
- **Advice for Future Teams:** Carefully plan the lifecycle of your relational links. If one table depends on another (e.g., users depending on sessions), every create, update, or delete action must be paired with corresponding updates elsewhere. Always build explicit functions like updateUserSession(userId, sessionId) and clearUserSession(userId) and

call them immediately when session states change, rather than hoping updates happen elsewhere in the code.

-

Location Fetching Issues - Celina

- **Problem:** We had problems where the frontend would not display any study sessions because our backend API route for fetching locations was incorrectly implemented. Specifically, database queries were not structured to properly return the expected data format for frontend consumption, and the route sometimes failed silently.
- **Cause:** Inconsistent API response structures and lack of thorough error checking after querying the database.
- **Advice for Future Teams:** Always clearly define your expected API request and response formats early on and enforce them strictly. If your API endpoint is supposed to return an array of location objects, always check that it's doing so — both on success and failure. Also, testing your API endpoints independently using tools like Postman before was very helpful with testing API calls.

Review Updating Issues - Apoorv

- **Problem**: When implementing the Reviews feature, we initially encountered issues where users were unable to submit reviews successfully. The problem was due to improper handling of the POST request and missing backend validation.
- **Cause:** Backend routes were not parsing form data correctly and were not giving clear error responses to the frontend when failures happened.
- **Advice:** Always set up proper request body parsing. Also, return clear success and error codes so that frontend developers can debug issues quickly. Silent backend failures are especially hard to detect without good error messages.

8. Future Work

There are several areas where the StudyLync application could continue to improve beyond the current version:

1. Visualization Improvements: The current real-time map works effectively, but future versions could add heatmaps, clustering techniques, or customized pin designs to improve how session density is visualized without overwhelming the user. This would make it even easier for users to quickly find popular study locations.

2. Messaging System or Session Planning: Adding an in-app messaging system would allow users within the same study session to communicate and coordinate before meeting. Similarly, reintroducing a lightweight session planning tool would help students organize sessions in advance and encourage higher session retention and participation.

3. Session Parameter Enhancements: Adding additional attributes to study sessions, such as a live count of the number of participants, would simplify some of our current backend queries and provide users with more useful information when deciding whether to join a session instead of the queries we depend on right now.

By focusing on these future enhancements, our project could become a more dynamic, user-centered platform for academic collaboration.

9. Division of Labor and Teamwork

The division of labor throughout the project was pretty equal and highly collaborative throughout the entire process. All team members actively contributed to every part of the project — whether it was brainstorming, coding, debugging, or adding updates to the repository, we worked together in real time. Any time we made major changes or additions, the entire team was present, ensuring that everyone had a clear understanding of the codebase and the evolving functionality of the application.

We organized our commits into clear sections based on project requirements. Each member took primary responsibility for certain features or deliverables, but contributions overlapped often, especially when larger components needed to be built, integrated, or debugged. Importantly, everyone also contributed to writing the final project report, so that the documentation accurately reflected the team's shared understanding of the project.

Overall, we believe our team managed the workload and collaboration process very well. We consistently communicated deadlines, personal challenges, and action items at every stage of the project. We used an ongoing list of tasks that was updated as the project progressed, allowing team members to pick up tasks according to their current availability and strengths. Whenever an individual encountered difficulties implementing a feature, other team members immediately stepped in to brainstorm and debug solutions. This approach kept our project moving forward steadily and maintained a strong team dynamic from start to finish.