CMPT 276 – Planning Report

TITLE Cirkle – Collaborative Study for Better Results

Group Members

1. Julianna Morena

2. Mudasser Mashal

3. Yogya Agrawal

4. Nafeesa Leena

Link to Github Repository:

https://github.com/CMPT-276-SPRING-2025/final-project-07-hills

1. API Selection and Usage

API 1: Google Drive API

Description: The Google Drive API lets users interact with Google Drive, allowing users to upload, organize, and manage files programmatically. In our study application Cirkle, we'll use this API to give students an easy way to access, and share study materials like lecture notes, assignments, and practice questions. By integrating Google Drive, users will have a cloud-based system where they can keep all their study resources.

Usage in Project:

1. Let students upload and organize study materials directly from their devices.
2. Allow users to share notes and files with classmates easily.
3. Provide real-time previews of study materials so users can check their content without downloading files.

API 2: Google Docs API

Description: Google Docs API helps to programmatically create and edit Google Docs from within an application. It also helps format the document. The Google Docs API helps applications create new documents, modify existing documents, and format text while styling them in various ways. The Google Docs API is extensively used in productivity applications, collaborative workspaces and educational platforms to automate document creation and organizing.

Usage in Project:

1. Allow students to create and manage collaborative notes within study groups.
2. Enable real-time document structuring, including bullet points, numbering, tables, and equations.
3. Integrate Google Drive file links within notes to keep all study materials easily accessible.

2. Features Description:

Google Drive API:

Feature 1: Upload and Organize Study Materials

- Description: Users can upload study materials (PDFs, Word documents, presentations, etc.) directly from the app to their Google Drive. These files will be stored in a dedicated folder linked to their account, keeping study materials easily accessible.
- User Benefit: This feature allows students to store all their learning resources in a single place, making it convenient to access from any device.

- Relevance: Ensures that students can organize their notes, assignments, and reference materials efficiently without worrying about losing important files.

Feature 2: Search and Retrieve Files Efficiently

- Description: The app integrates with Google Drive's search functionality to allow users to quickly find files using keywords, file types, or modification dates.
- User Benefit: Saves time by allowing users to instantly locate notes, assignments, or resources instead of scrolling through multiple folders.
- Relevance: Enhances productivity by making study materials easily searchable, especially useful for students managing large amounts of content.

Feature 3: File Sharing and Collaboration

- Description: Users can share study materials with classmates or group members directly from the app by generating shareable Google Drive links with custom permissions (view, edit, comment).
- User Benefit: Makes it easy to collaborate on notes, projects, and assignments, enabling efficient teamwork.
- Relevance: Supports group study sessions, peer collaboration, and file exchange without the need for constant email attachments or manual file transfers.

Google Docs API:

Feature 4: Create Notes

- Description: Users can create new notes in their study group folder easily. The app will rely on the Google Docs API to create a new document optimized for collaborative efforts within the group.
- User Benefit: Students can create and edit notes in real time for discussion and studying in groups. Because notes automatically get organized into shared folders, your team can access, edit, and contribute to them without document worries. This guarantees that everything is streamlined and arranged.
- Relevance: Students can create, manage, and work on each other's study notes without worrying about managing documents in folders.

Feature 5: Organize Notes

- Description: Users can organize notes using bullet, number, table, and collapse options for better readability. Other advanced formatting options such as equations and images are also supported.
- User Benefit: By utilizing structured notes, students are able to enhance their learning capacity and material review. Collapsable sections help users ease out while navigating through the space

to learn a point. By giving users advanced formatting tools, the app ensures that students write smart, professional notes to study and present.

- Relevance: This will help students make organized and visually clear notes to study better.

Feature 6: Link Files to Notes

- Description: Users can insert a link to any Google Drive file directly into Google Docs notes. All study notes are readily available in one place. Group members will automatically be given a link preview generated by the app along with an access permission check.
- User Benefit: Students can now easily refer to other notes like lecture slides, previous assignments, research papers, and more without jumping tabs. Since all the required materials and files will be linked in one place, it will save the students time in searching for auxiliary files.
- Relevance: Helps students keep all items linked up to study together so that documents and references are available.

3. Medium-Fidelity Prototype

We have created a medium-fidelity prototype for our application “Cirkle” using figma. The user flow has been defined in the prototype and a demonstration of such can be found in our video. Please refer to the screenshots below to get an overview of our prototype.

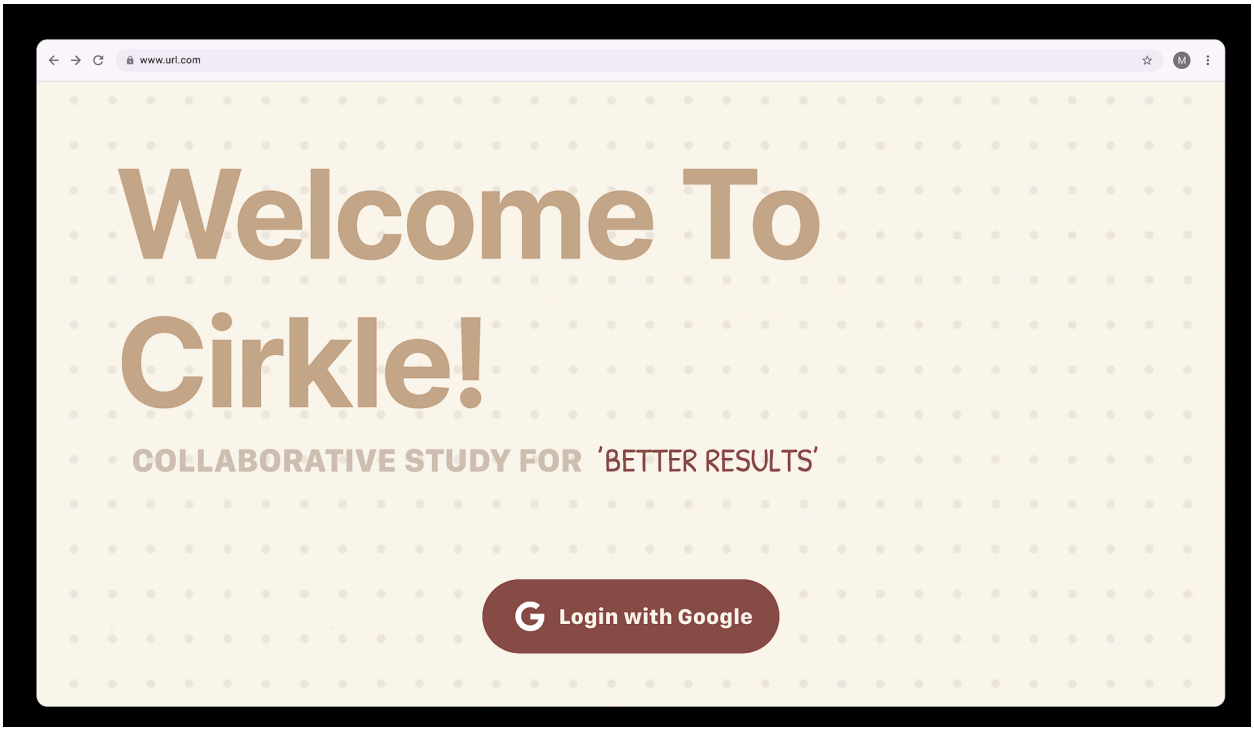


figure 1: Landing Page

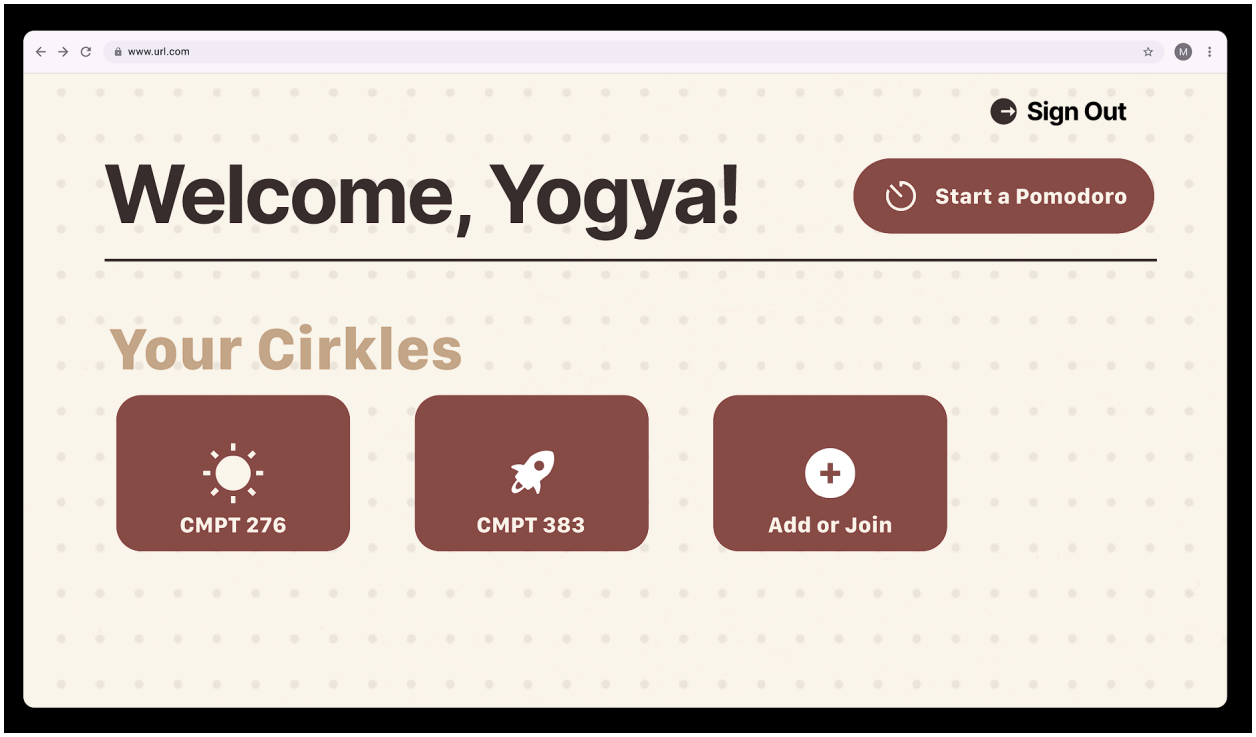


figure 2: Home Page

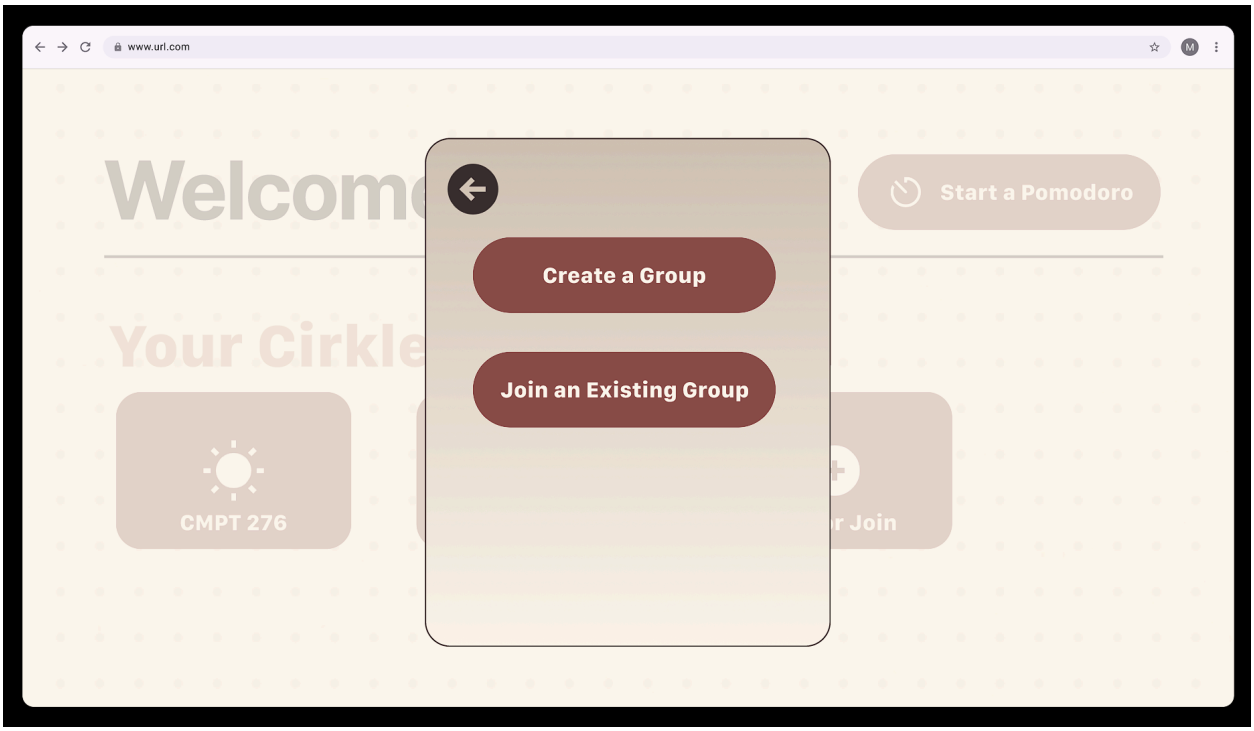


figure 3: Home page with a pop-up to create a group or join an existing one

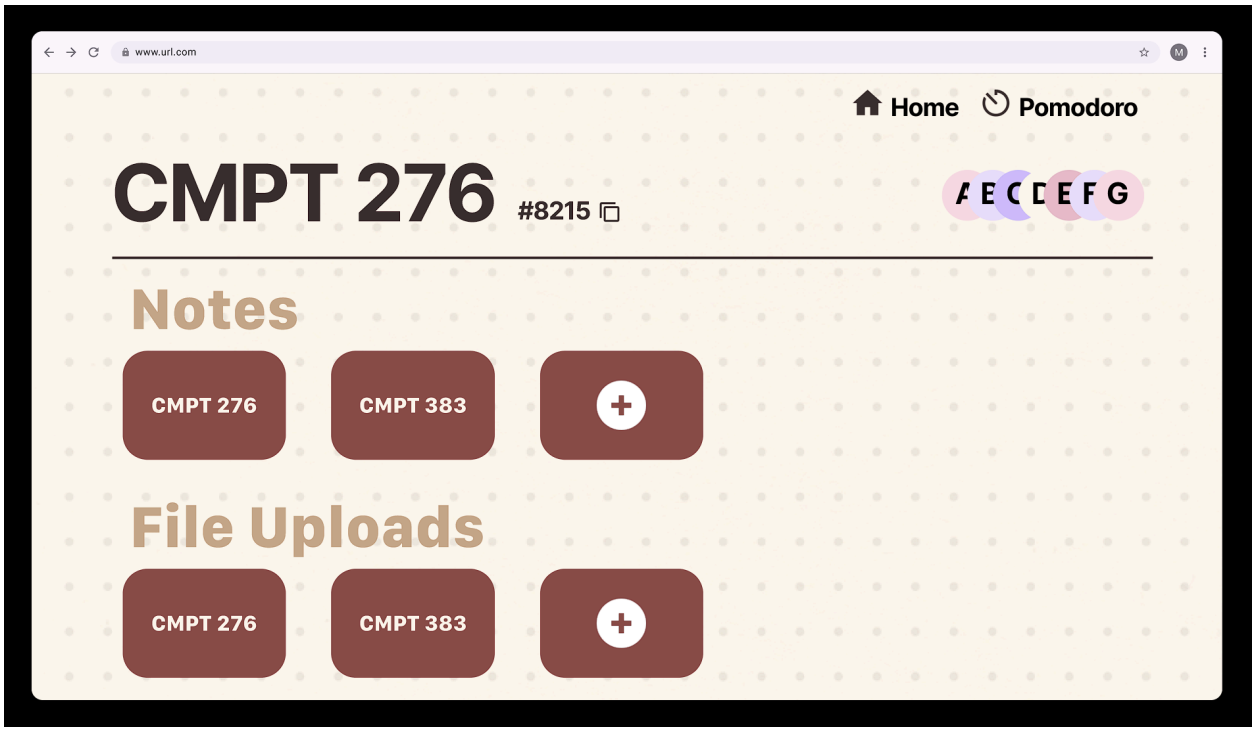


figure 4: Group Page for CMPT 276

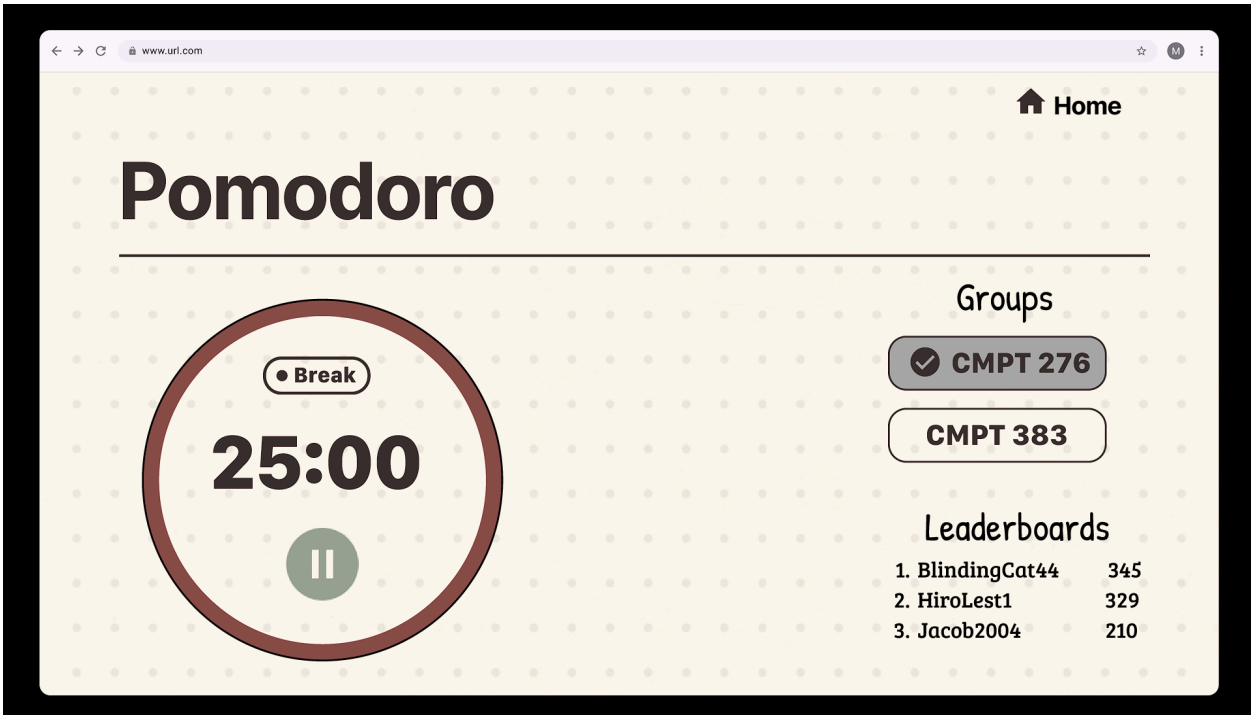


figure 5: Pomodoro Page

4. SDLC Model

For Circle's development, the Agile SDLC model is the most suitable choice due to its flexibility, iterative approach, and emphasis on teamwork; these are key elements that align with the project's academic requirements and collaborative nature. This application is designed to serve as a streamlined learning tool, incorporating essential features like shared notes, file uploads, flashcards, and a Pomodoro timer to enhance group study efficiency. Given the evolving nature of project development within a school setting, Agile allows for continuous iteration, incremental improvements, and regular feedback integration, ensuring the platform meets both functional and academic expectations. We will implement a Kanban framework, providing a visual workflow to track progress, manage individual workloads efficiently, and maintain a steady development pace within the limited time frame of the project. This structure will help our team dynamically prioritize tasks, ensuring steady progress while allowing room for necessary revisions based on feedback.

5. Work Breakdown Structure

Phase 1: Planning & Setup

- Define project scope, goals, and target users.
- Finalize core features and functionality, including API usage.
- Create wireframes and mid-fidelity prototypes to map out the user interface.
- Set up version control with GitHub and create an initial repository.
- Choose the technology stack and set up the development environment.

Phase 2: Website Development

- Build the front-end structure, including homepage, navigation, and study dashboard.
- Implement user authentication and account management.
- Integrate Google Drive API for file uploads, storage, and retrieval.
- Integrate Google Docs API to allow users to create, edit, and manage documents within the platform.
- Develop a search and organization system for uploaded study materials.
- Implement file-sharing and collaboration features for group study sessions.

Phase 3: Testing & Refinement

- Conduct functional testing to verify file uploads, retrieval, and document editing.
- Perform API request efficiency tests and optimize for performance.
- Check security measures, including user authentication and data access controls.

- Gather user feedback and refine UI/UX for better usability.
- Fix bugs and resolve any integration issues.

Phase 4: Deployment & Launch

- Deploy the website
- Monitor for post-launch issues and address any final optimizations.

6. Project Schedule

Phase 1: Planning & Setup (Before March 7)

- Define project scope, goals, and target users *(Completed)*
- Finalize core features and APIs *(Completed)*
- Create wireframes & mid-fidelity prototype *(Completed)*
- Set up GitHub repository & version control *(Completed)*

Phase 2: Website Development (March 7 - April 1)

- Develop homepage, navigation, & study dashboard (March 7 - March 10) *[2-day buffer]*
- Implement user authentication (March 11 - March 13) *[1-day buffer]*
- Integrate Google Drive API (file upload/storage) (March 14 - March 16) *[1-day buffer]*
- Integrate Google Docs API (document editing) (March 17 - March 19) *[1-day buffer]*
- Develop search & organization system (March 20 - March 22) *[1-day buffer]*
- Implement file-sharing & collaboration (March 23 - March 25) *[2-day buffer]*
- Internal review & refinements (March 26 - April 1)

Phase 3: Testing & Refinement (April 2 - April 5)

- Functional testing (file uploads, retrieval, document editing) (April 2 - April 3) *[1-day buffer]*
- Optimize API request efficiency & performance (April 4)
- Security testing (authentication, data access) (April 5)

Phase 4: Deployment & Finalization (April 6 - April 8)

- Deploy website (April 6) *[1-day buffer]*
- Monitor for post-launch issues & final optimizations (April 7) *[1-day buffer]*
- Final review & submission (April 8)

7. Risk Assessment

Low-Risk Issues & Mitigation Strategies

1. Minor UI Design Issues: Small design inconsistencies or flaws could arise during development.

   Mitigation Strategy: We will conduct testing, follow design guidelines, and collect feedback from classmates.

2. Version Control Conflicts: Multiple team members working on the same file could cause merge issues.

   Mitigation Strategy: We will use Git with a planned branching strategy (such as creating branches for any new features) and communicate before merging changes.

3. Scope Creep: Adding unnecessary features could take time away from core requirements.

   Mitigation Strategy: We will follow the project proposal, prioritize essential features, and review the scope with our TA before making changes.

4. Unfamiliarity with Development Tools: Some team members may struggle with the technologies used in the project.

   Mitigation Strategy: We will assign tasks based on expertise, provide knowledge-sharing sessions, and use online resources to fill gaps in understanding.

5. Team Member Availability Issues: Conflicting schedules due to other coursework or personal commitments.

   Mitigation Strategy: We will use a shared calendar for meetings, set clear deadlines, and distribute workload evenly.

Medium-Risk Issues & Mitigation Strategies

6. Integration Challenges: Ensuring all components (e.g., notes, flashcards, Pomodoro timer) work together smoothly.

   Mitigation Strategy: We will develop features in a modular way, and conduct frequent integration tests while working on the project.

7. Conflicting Opinions: Disagreements on feature design or coding approaches may slow down progress.

   Mitigation Strategy: We will agree on a common design approach early, and use majority votes for decision-making (utilising collaboration/compromise if the majority vote should fail as well).

8. Third-Party API Reliability: Using external libraries or APIs (e.g., for timers or file storage) could cause failures.

Mitigation Strategy: We will choose stable, well-documented libraries and have fallback options in case an API fails.

9. Team Workload Imbalance: Some team members may end up doing more work than others.

   Mitigation Strategy: We will Notion as a task-tracking tool to ensure fair workload distribution and adjust tasks as needed.

10. Cross-Browser and Device Compatibility Issues: Features may not work well on all devices.

    Mitigation Strategy: We will test on multiple browsers (Chrome, Firefox, Safari) and ensure responsive design for mobile users.

High-Risk Issues & Mitigation Strategies

11. Data Loss or Corruption: A system crash or accidental deletion could result in lost work during the development process.

    Mitigation Strategy: We will use Github's cloud-based storage as a safety feature, enabling auto-save features and keeping local backups.

12. Team Member Unavailability: A team member may become unavailable due to personal reasons, illness, or academic workload.

    Mitigation Strategy: We will ensure that communication remains smooth and clear through conducting regular check-ins, and we will guarantee that each part of the project has at least two people that can work on development if needed.

13. Unresolved Bugs or Functional Errors: Unfixed issues may negatively impact the user experience and project evaluation.

    Mitigation Strategy: We will allocate time for debugging, conduct testing, and fix critical errors before submission.

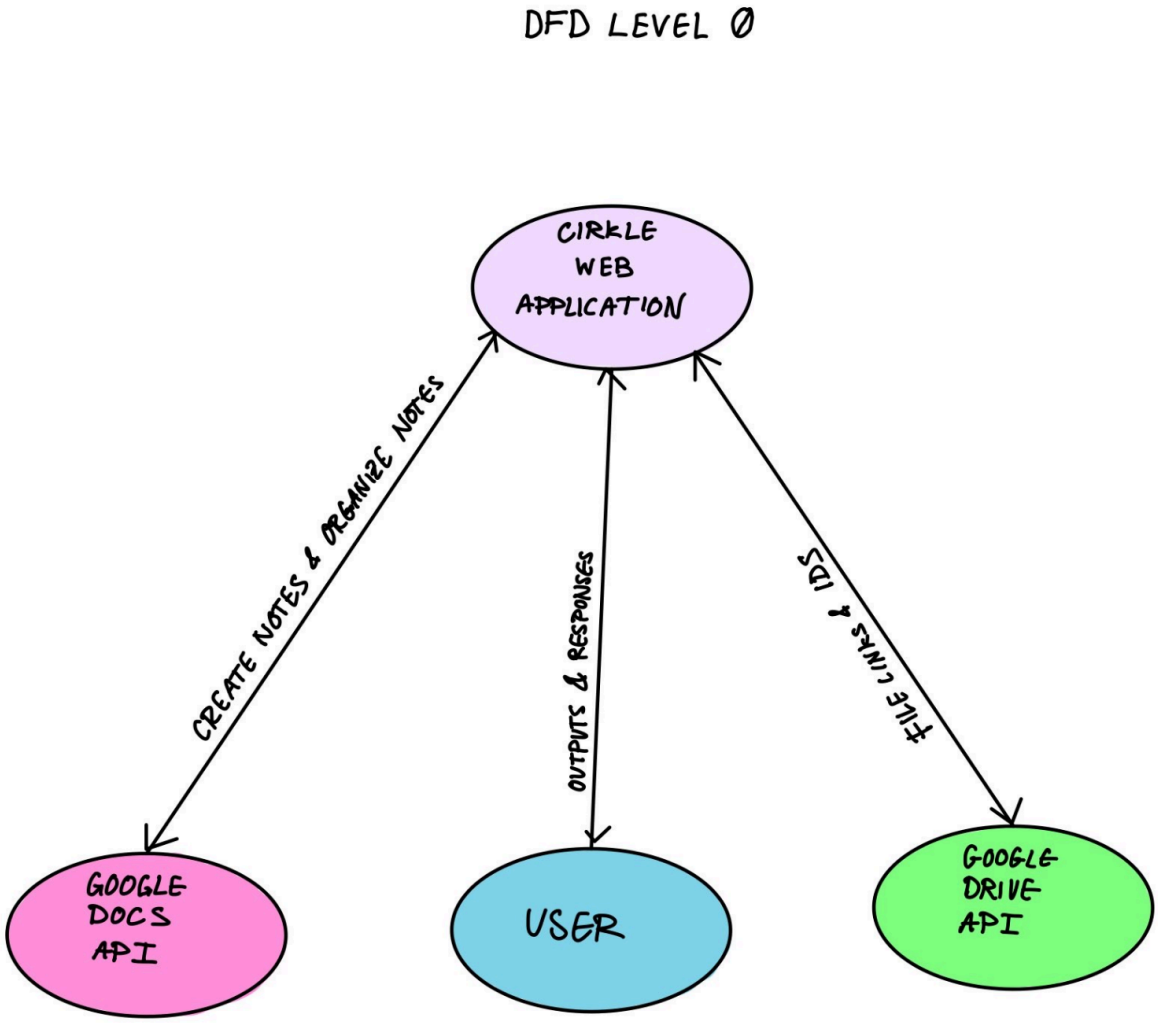
14. Delays in Development Timeline: Unexpected academic workload may cause missed deadlines.

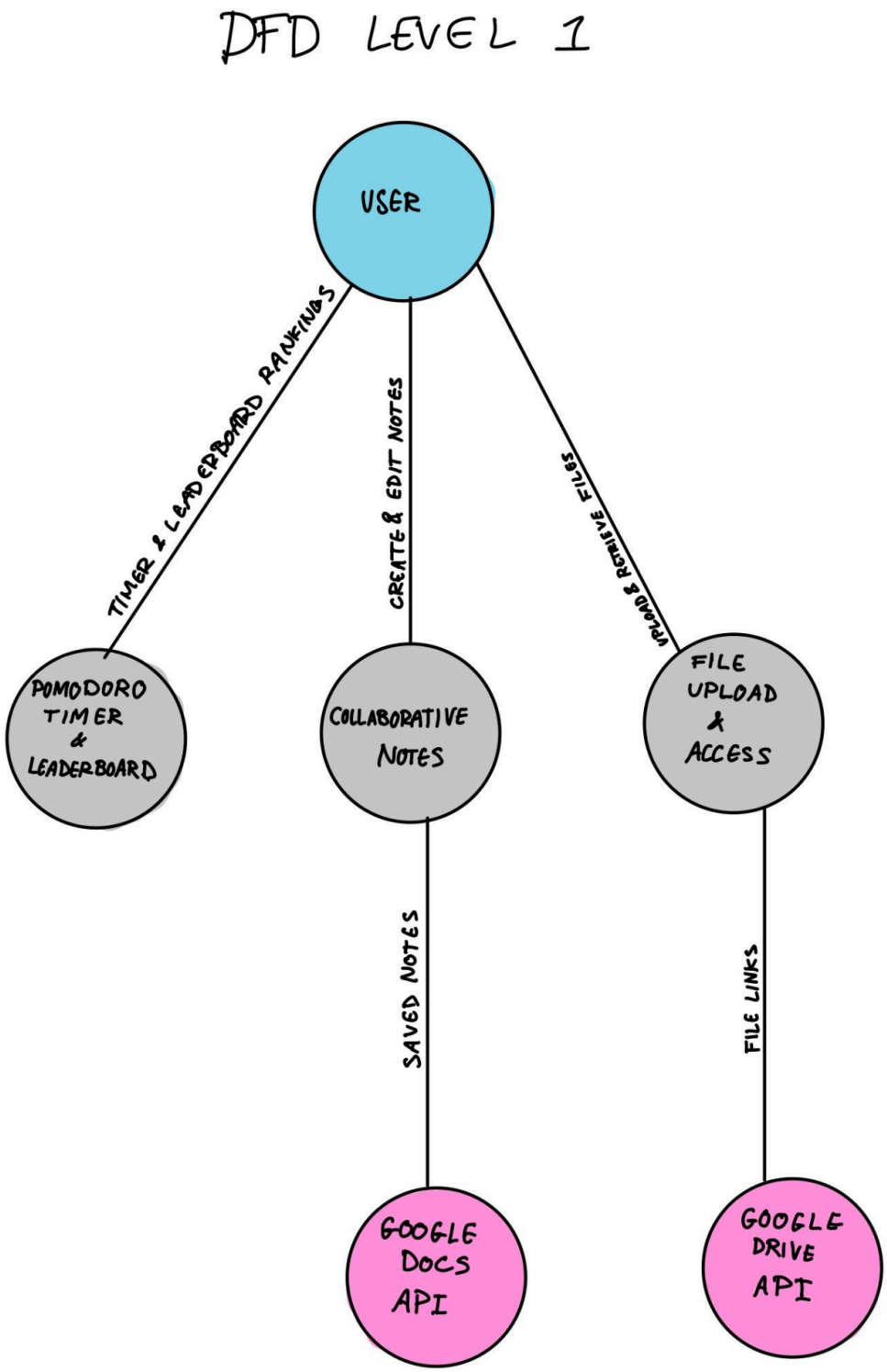
    Mitigation Strategy: We will follow a strict project schedule, set internal deadlines ahead of the official due date, and allocate extra buffer time.

15. Failure Before Submission: The platform could break or become unresponsive right before the final demo.

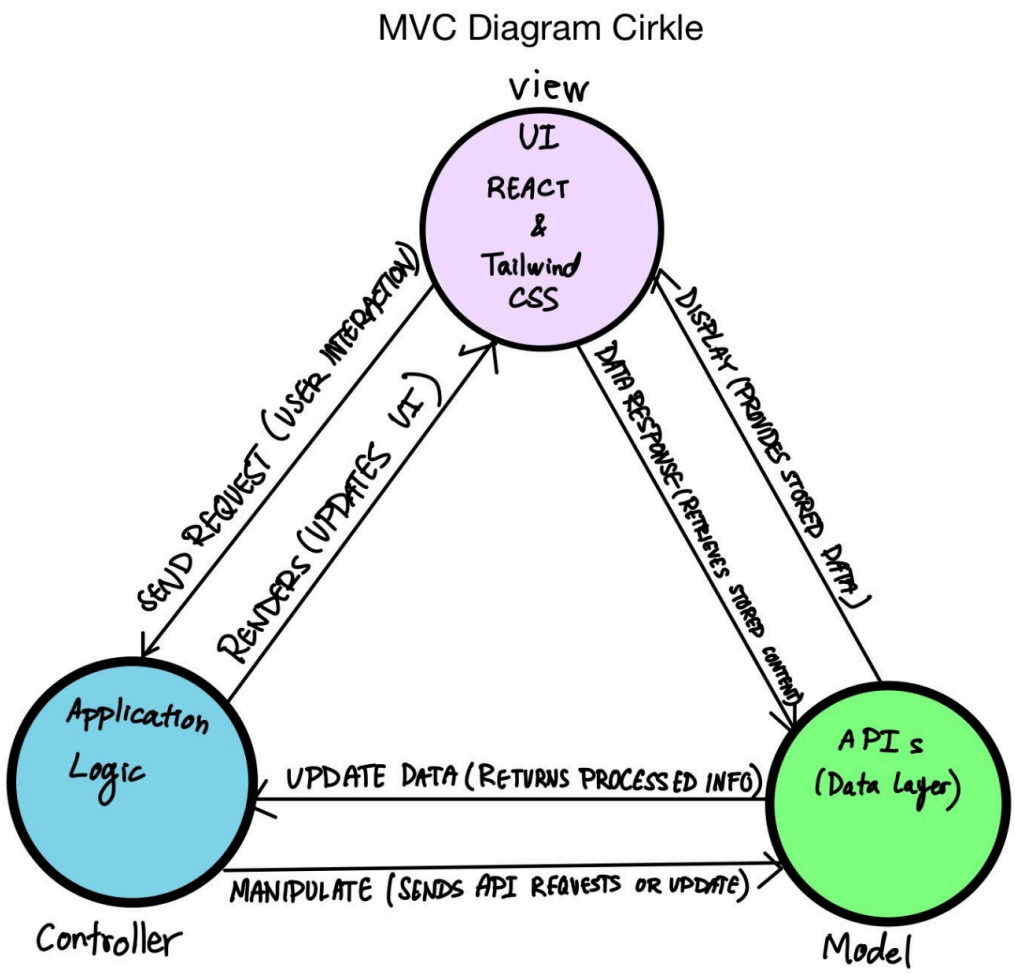
    Mitigation Strategy: We will conduct thorough testing before submission, keep a stable backup version, and have an emergency plan (recording or screenshots).

8. Data Flow


DFD LEVEL 0
CIRKLE WEB APPLICATION
CREATE NOTES & ORGANIZE NOTES
OUTPUTS & RESPONSES
FILE LINKS & IDS
GOOGLE DOCS API
USER
GOOGLE DRIVE API


DFD LEVEL 1
USER
TIMER & LEADERBOARD RANKINGS
CREATE & EDIT NOTES
UPLOAD & RETRIEVE FILES
POMODORO TIMER & LEADERBOARD
COLLABORATIVE NOTES
FILE UPLOAD & ACCESS
SAVED NOTES
FILE LINKS
GOOGLE DOCS API
GOOGLE DRIVE API

9. MVC Model



Model - The Model of Cirkle handles storage, retrieval, API data generation, and interaction storage. The purpose of the Model is to manage files and notes and provide study materials.

Key Responsibilities:

Google Drive API

- Stores uploaded files and managed access permissions.
- Returns file IDs and access links after successful uploads.

Google Docs API

- Manages real-time collaborative note-taking.
- Ensures that notes are updated and synced across all users.

Example Workflow

When the user uploads a file, the Model sends a request to the Google Drive API, which uploads the file and returns the file link. The attached link is then stored and fetched once the study material is accessed by the user.

View (User Interface - React + Tailwind CSS) - The user interface in Cirkle is designed using React and Tailwind CSS, making it simple to use and interactive.

Key Responsibilities:

- Shows files uploaded, joint notes, and studying gadgets.
- The application will change as per what the user does without refreshing the page.
- Displays current study sessions, leaderboards, and decks of flashcards.

Example Workflow:

When the user opens the study group dashboard, the View pulls down the saved files and notes and places them in a structured and interactive way. Any changes made to notes appear instantly, without having to reload the page.

Controller - The Controller takes on the role of the intermediary between the Model and View. It is the component that processes requests made either by users or from the API.

Key Responsibilities:

- Handles user actions such as uploading files, creating notes, and managing study groups.
- Sends requests to Google Drive and Google Docs using an API to save and fetch data.
- Updates the View dynamically when data changes.

Example Workflow:

When a user makes a new note, the Controller sends a request to Google Docs API, which saves the content. The Controller then gets this updated new note and updates the View, so all users see the latest version.

How MVC Works Together in Cirkle -

File Upload Process:

A user clicks “Upload File” in the study group dashboard. The Controller sends a request to Google Drive API, which stores the file and sends an access link to the user. After this, the Controller receives the link to the file and passes it onto the View, which helps display the newly uploaded file in the dashboard.

Collaborative Note-Taking Process:

A user opens a shared note and makes edits. The Controller sends the content to Google Docs API, which saves it. Furthermore, the API syncs the updates for all users. The Controller gets the newest version of the note and updates the View to show all users the most recently updated note.

Appendix:

Contributions:

Julianna Morena: The SDLC model, risk assessment, the overview of the project in the video, editing and subtitles in the video, appendix.

Mudasser Mashal: Google Drive API description and features, work breakdown structure, project schedule, overview of the chosen APIs in video.

Yogya Agrawal: The mid-fidelity prototype, walkthrough of prototype in video.

Nafeesa Leena: Google Docs API description and features, Data Flow Diagrams, MVC model diagram, overview of application data flow in video.

Changelog:

- Changed Notion API to Google Docs API.

No Additional Diagrams/Charts/Tables