Deliverable 2 Report

Part 1: Project Planning

A. Summary

A summary of the project plan outlining the sequence of user stories and development milestones.
Jobba is a web application that helps users manage their job search by simplifying tasks like tracking applications, networking, and handling email communication. While there was an existing version of Jobba, its backend functionality had already been implemented. In **D2**, our focus was on enhancing the frontend. We converted all the existing HTML components into **React**, integrated the frontend with the backend functionality, and successfully deployed the application.

User Stories:

- **As a job seeker,** I want to log into my account and link it to my Gmail so that the system can access my emails, extract relevant job application data and show it to me.
- **As a job seeker,** I want to view the CSV file generated that contains my job application details in a well-organized format so that I can easily track my application statuses.
- **As a job seeker**, I want a well-organized and easily accessible navbar so that I can efficiently navigate between different features, pages and integrations without confusion.

Development milestones:

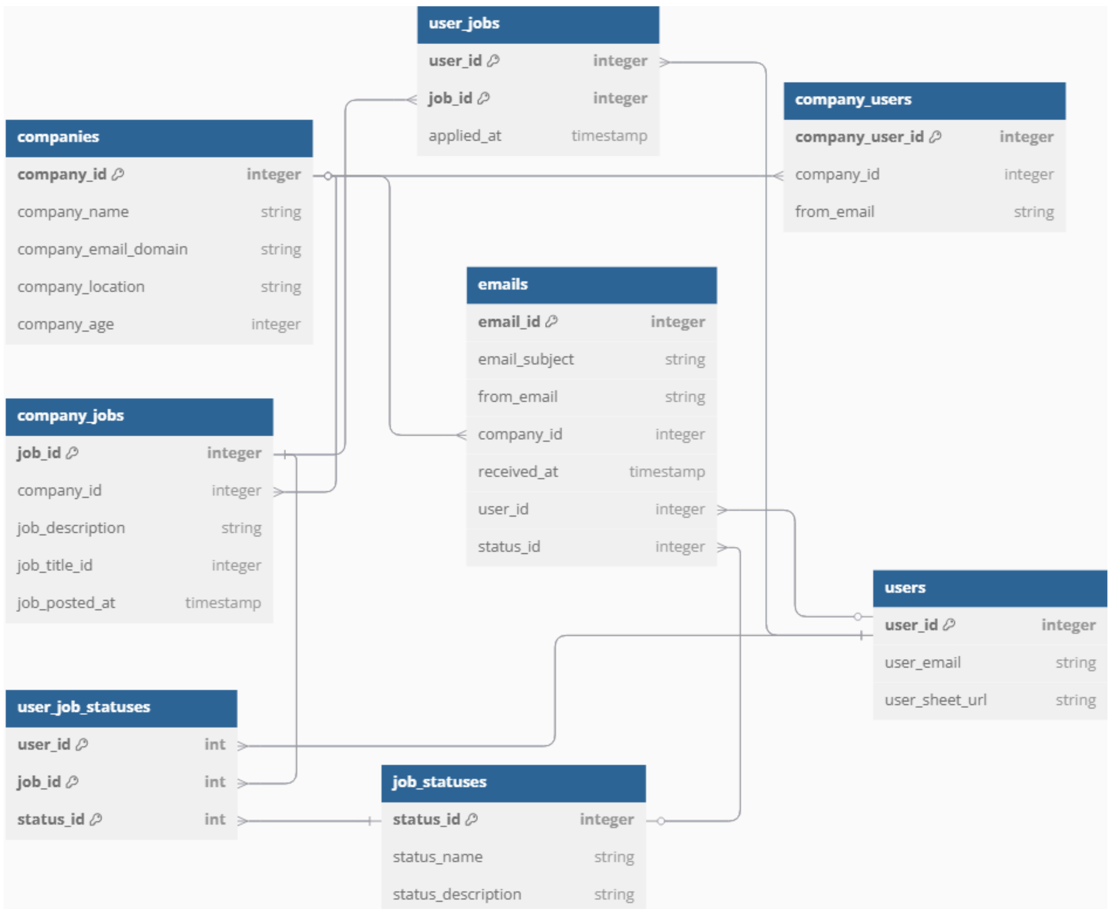
1. **Frontend Development:**
    a. Take the existing HTML/CSS structure and convert it into React components (e.g., Navbar, etc.).
    b. Implement a toggle feature for switching between light and dark themes.
    c. Design and style the features on the homepage (Navbar), ensuring it’s both visually appealing and functional.
    d. Set up authentication UI using Google login for easy user access.

2. **Backend Development:**
    a. Connect the front end to the Gmail API to allow email access.
    b. Set up backend routing to handle easy navigation between pages, allowing smooth transitions via buttons or direct links.
    c. Implement the functionality for downloading the CSV file containing the job application data once it's processed.

3. **Testing & Debugging:**
    a. Verify that the Gmail API is properly fetching emails and extracting job application data as expected.
    b. Identify and resolve any visual or interactive bugs (e.g., buttons not responding, misaligned elements) and improve the user experience.
    c. Test UI on multiple devices to ensure responsiveness.
4. **Deployment:**
    a. Deploy the application to Render (seperate backend and frontend)
    b. Test the deployed version to ensure all features are working as expected.

B. The Final System Model

- Database Diagram:



user_jobs	
user_id	integer
job_id	integer
applied_at	timestamp

company_users	
company_user_id	integer
company_id	integer
from_email	string

companies	
company_id	integer
company_name	string
company_email_domain	string
company_location	string
company_age	integer

emails	
email_id	integer
email_subject	string
from_email	string
company_id	integer
received_at	timestamp
user_id	integer
status_id	integer

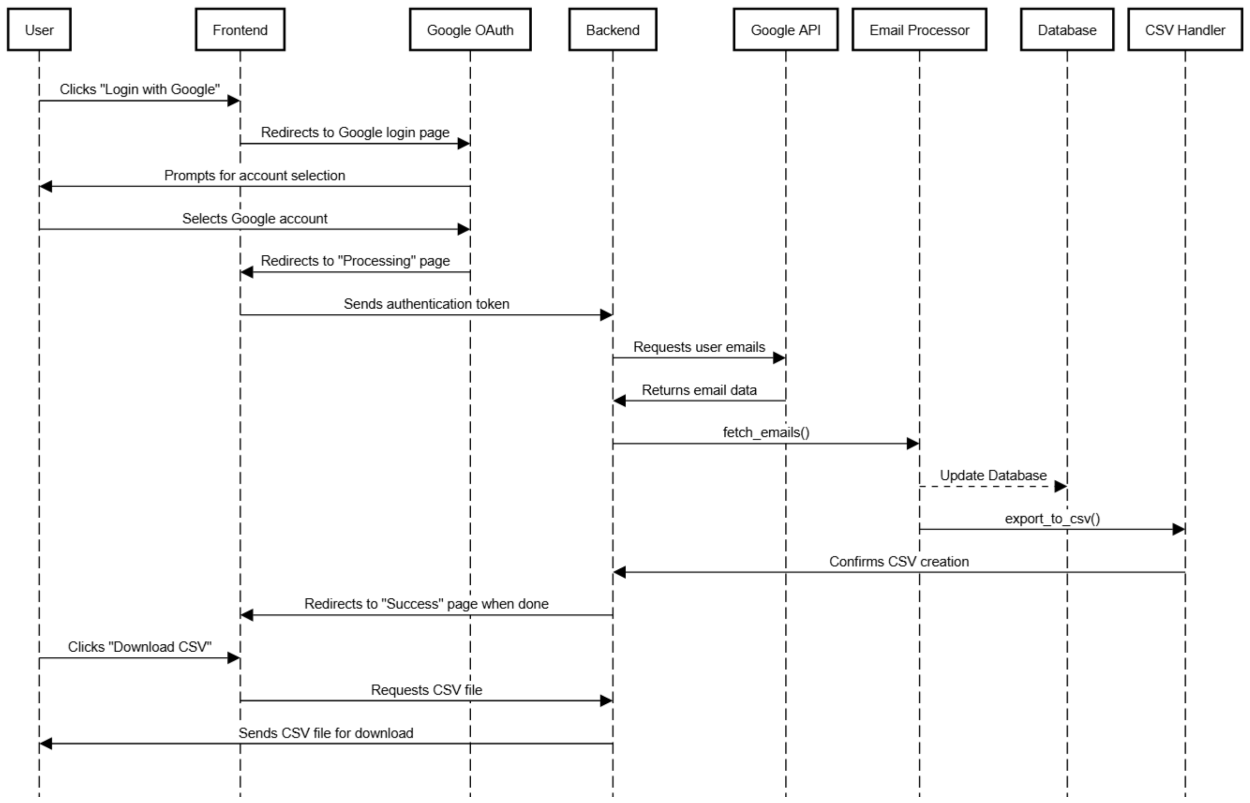
company_jobs	
job_id	integer
company_id	integer
job_description	string
job_title_id	integer
job_posted_at	timestamp

users	
user_id	integer
user_email	string
user_sheet_url	string

user_job_statuses	
user_id	int
job_id	int
status_id	int

job_statuses	
status_id	integer
status_name	string
status_description	string

- Sequence Diagram:



C. Division of Work Between Subteams + Explanation

Subteam 1:

- Google API linking
- Navigation bar (theme changing)
- Success Page
- Testing
- Deployment

This team had the most front end experience and most members, and had already been working with our partner to determine Tech stack specifications and repository set up.

- Subteam 2:
    - Home page
    - Processing page
    - Converting original HTML/CSS pages to React
    - Testing

This team was enthusiastic about updating the old UI using the new tech stack, and thus went on to work on more of the UI and related testing.

- Subteam 3:
    - Email Data processing
    - Job application data exporting and downloading
    - Testing

This team had the most backend experience, and was most looking forward to improving upon existing functionality so we had them work on the data processing.

D. Technology Stack Chosen for the Project + Explanation

- **ReactJS/Nextjs**
- **javascript/typescript**
- **HTML/CSS/TailwindCSS**
- **Javascript**
- **Python/FastAPI**
- **Playwright -** We are new to testing frameworks and chose this one because it is easy to install and use. Writing tests is straightforward, and running them is simple. The framework provides detailed information about test failures, including the reasons behind them. Additionally, it generates an HTML report, making it easier to track and analyze failing tests.
- **Docker**

Part 2: Sub-team Reports

Subteam 1

A. Team Members and Roles:

- Alice Sedgwick - Frontend, Backend, Google Login, Product Manager, Deployment
- Elyse Ando - Frontend, Google Login/out, Navigation Bar, Testing, Deployment
- Jediel Antalan - Frontend, Repo setup, Tech support, Deployment

B. Features implemented by the subteam:

1. Google Login Integration (Backend)

- Description: Connected the Google Login button to the backend API, enabling secure authentication for users. Ensured proper handling of authentication tokens and user sessions.
- Technologies Used: FastAPI, OAuth2, Google Identity Platform

2. Download Button (Backend & Frontend)

- Description: Implemented a download button on the success page, allowing users to retrieve relevant data or files. Connected frontend interactions with backend logic to enable seamless downloads.
- Technologies Used: FastAPI, Next.js, React, Tailwind CSS

3. Monorepo Setup

- Description: Converted the existing repository into a monorepo structure to streamline development across the frontend and backend, making dependency management more efficient.
- Technologies Used: npm workspaces, Git, NextJS/ReactJS, FastAPI/Python

4. Frontend Design & UI Components

   a. HeroUI Integration
      - Description: Installed and integrated HeroUI components to create a visually cohesive and responsive user interface.
      - Technologies Used: React, Next.js, Tailwind CSS, Heroicons

   b. Navigation Bar Updates
      - Description: Updated and improved the navigation bar for better usability, ensuring proper link routing and styling.
      - Technologies Used: React, Next.js, Tailwind CSS

   c. Google Login Button (Frontend)
      - Description: Created and styled the Google Login button on the homepage, enabling users to authenticate with their Google accounts.
      - Technologies Used: React, Next.js, Tailwind CSS, OAuth2

   d. Logout Button on Success Page
      - Description: Implemented a logout button on the success page to allow users to securely sign out.
      - Technologies Used: React, Next.js, Tailwind CSS

5. Backend-Frontend Integration

   - Description: Connected the FastAPI backend with the Next.js frontend, ensuring proper API communication for authentication and data handling.
   - Technologies Used: FastAPI, Next.js, Axios

6. CI/CD and Code Quality Enforcement

   a. Pre-commit Hooks
      - Description: Installed Husky to enforce formatting and linting before commits, improving code consistency across the project.
      - Technologies Used: Husky, ESLint, Prettier

   b. GitHub Actions Workflow
      - Description: Created a CI/CD pipeline to automate frontend code checks, ensuring formatting, linting, and successful builds before merging code.
      - Technologies Used: GitHub Actions, ESLint, Prettier

7. <u>Success Page Development</u>

- Description: Designed and developed the success page layout, including styling, content, and messaging for users after successful login.
- Technologies Used: React, Next.js, Tailwind CSS

8. <u>Testing</u>

- Description: Developed tests to verify the navigation bar functionality and success page UI elements.
- Technologies Used: Playwright

9. <u>Deployment</u>

- Description: Used Render to deploy the backend server and frontend application to allow the app to be available to the public.
- Technologies Used: Render

C. **Lessons learned from this phase.**

I. What worked well
   - A. MonoRepo
     1. Allowed for easy code referencing, and simplicity as far as working across subteams.
   - B. Using component library (HeroUI)
     1. Visuals were much cleaner.
     2. HeroUI added a lot of ease and depth to the application.
   - C. Teamwork
     1. Subteam was responsive in the Discord and everyone was available to help others when needed.
     2. Jed made sure to help anyone who encountered similar issues to him.
     3. Calls were made spontaneously when multiple developers were online and wanted to collaborate or ask for advice.

II. What didn't work well
   - A. Third Party Routing and authentication is complicated
     1. The initial implementation of Google Login did not reroute to the correct page, nor track user login data effectively.
     2. Once properly routed, determining what the partner's technique had been for leveraging the Google login information to extract job application data was quite complicated.
     3. Testing the system was unclear, since the test would not be able to effectively login, then have its job application data extracted (this might require a dummy job application email in the future).

B. Testing methods should be determined at an earlier stage in the development process.
    1. Testing for the backend had already been implemented by the partner, but there were initially no frontend tests. We had to quickly decide what testing method to use so a lot of time was spent researching which method was the fastest to learn and easiest to implement.

C. Tasks need to be assigned such that the overlap between tasks is clearly assigned and not designated as floating work.
    1. A lot of last minute coordination to pick up on work that was left unassigned had to happen to make the deployment work.

Subteam 2

A. Team Members and Roles:

1. Mugdha Banthwan - Frontend, Backend, Testing, Documentation
2. Happy Nasit - Frontend, Backend, Testing, Documentation

B. Features implemented by the subteam:

1. Processing Screen:
    - Description: The front-end design of the processing screen was transformed into a React component. This involved converting static HTML/CSS into a dynamic and modular layout.
    - Technologies Used: React, Next.js, Tailwind CSS, Heroicons.
2. Homepage Screen:
    - Description: The front-end design of the homepage screen was transformed into a React component. This involved converting static HTML/CSS into a dynamic and modular layout, adding a google login button and adding multimedia links.
    - Technologies Used: React, Next.js, Tailwind CSS, Heroicons.
3. Testing:
    - Description: Researched to find an appropriate testing framework and helped other team members set up the framework. Developed tests to check the homepage elements and the spinner component of the processing page.
    - Technologies Used: Playwright
4. Class Models:
    - Description: Developed structured data models to efficiently manage application data and relationships. The models implemented include: Users, UserJobs, User Job Status, Companies, Company Jobs, Job Status and Job Titles.
    - Technologies Used: SQLModel
5. Deliverable 2 Report:
    - Description: Completed the documentation for the **D2 Report**, including a comprehensive overview of the **Subteam Report** and the **Project Planning** sections, detailing the progress, and the milestones.

6. README.md:
   - Description: Completed the **README.md** file, providing a detailed overview of the application along with step-by-step instructions on how to access and use the app.

C. **Lessons learned from this phase.**

I. Effective Collaboration: The communication and teamwork were really strong. We were able to work in sync, which made the process much smoother. Having clear coordination helped us stay aligned and move forward without major issues.
   Prior Experience: Since we both had experience with React, the transition from static HTML pages to dynamic React components was smoother. We knew the key concepts like JSX, components, etc., which made it easier to break down the HTML pages into reusable components.

II. Test Setup: As a team, we were new to the testing framework, so it took us a lot of time to learn it. This slowed down our progress, especially when setting up the tests.
   Improvement: Next time, we'll make sure to spend more time upfront learning the testing tools. This will help us start testing earlier and avoid delays. We can also use online resources or get help to speed up the process.

Subteam 3

A. Team Members and Roles:
1. Kevin Lam: Backend developer, Database model creation
2. Justin Li: Backend developer, Database model creation

B. Features implemented by the subteam:

During this implementation phase, we focused on enabling job seekers to export their job application data outside the app.

1. Export Job Application Data to CSV
   - Implemented functionality to allow users to export their job application data as a CSV file.
   - Technologies Used: FastAPI, Google Gemini API, BeautifulSoup4 (bs4), Google Login
2. Integration of File Download in the Success Screen
   - Integrated a FastAPI endpoint to generate and serve the CSV file for download.
   - Technologies Used: FastAPI, Next.js, Tailwind CSS
3. Database Models
   - Implemented database models for managing email storage and tracking export history.
   - Created an email table for storing fetched emails.
   - Developed export and export_type tables to log export activities when users download their data.
   - While these models are not yet used in the current implementation, they are structured for future integration with a global database.
   - Technologies Used: SQLModel
4. Unit Testing

- Developed unit tests using Pytest to ensure data integrity and validate the functionality of the database models.
- Wrote test cases to verify email fetching and CSV export processes.
- Technologies Used: Pytest

5. Deliverable 2 Report
    - Completed the documentation for the D2 Report, which includes:
        - A detailed summary of the Subteam Report
        - The Project Planning section, detailing development progress, milestones and project architecture.

C. **Lessons learned from this phase.**

What worked well?

- Effective Collaboration – Once we got comfortable with PRs and branching, working on the same branch became smoother, allowing for asynchronous development and more efficient workflows.
- Understanding the Codebase – Taking the time to analyze existing code helped us implement new features without breaking functionality.
- Experience with Unit Testing – Familiarity with Pytest and Python sped up testing and improved reliability.

What didn't work well? What can be improved?

- Automated Testing – Adding CI/CD testing would catch issues earlier and improve efficiency.
- Cross-Team Communication – Coordinating earlier with other sub-teams would ensure better backend-frontend integration and prevent misalignment.

Part 3: Team Progress

A. Common Foundation Summary + Explanation

We began by agreeing on a MonoRepo configuration for our web application for simplicity and locality. We then began work on the database and a frontend framework.

We began work on the models using PostgreSQL and planned on using AWS for the database, but then determined (at the mid-sprint meeting) that it was out of the scope of the second deliverable, and that we would have to come back to the database setup after the first deployment.

The frontend framework was done using HTML, Tailwind CSS, ReactJS, NextJS, and JS/TS. It was a simple introduction page which we would build upon in our subteams.

B. Sub-teams Contribution to the Project

We endeavored to divide the work relatively equitably between the subteams, and adjusted the workload at our halfway meeting for the sprint.

Subteam 1 was in charge of front-end backend communication setup (and other repo setup), the navigation bar, and Google Login- and authorization functionality for the first half of the sprint, and the success page and testing for the second half of the sprint.

Subteam 2 was in charge of Database model setup for the first half of the sprint, and the homepage and processing page, and related testing for the second half of the sprint. They also worked on the Planning part of the D2 Report and Readme.md.

Subteam 3 was in charge of improving existing data scraping functionality and database model setup for the first half of the sprint, and job application data downloading / exporting functionality as well as related testing in the second half of the sprint.

C. Overall Progress

For this deliverable, we succeeded in implementing all existing functionality (previously implemented by our partner) in the new tech stack we had agreed upon (as specified in Part1: Section D).

This involved creating 3 pages; the Home page, the Processing page, and the Success page; adding Gmail login and authorization for user based data scraping, the data collection and csv export and download functionality, and lastly testing for all of it.

D. Major Technical or Organizational Challenges

Google authorization, testing, and time management all proved to be challenges on their own. Ensuring that the 3rd party API correctly passed us the data we needed for

We experimented with quite a few frontend testing frameworks (Jest, Selenium IDE and Playwright), in order to come to the conclusion that Playwright would work best for our purposes, since its relatively beginner friendly, and has an IDE as well as an optional html report for easier testing flow.

Our time management was a pretty big challenge, as demonstrated by our submission of the delay form, since we underestimated how difficult testing would prove to be, and mistakenly directed a lot of resources towards working on the database when in fact it had not been relevant to our intended deployment for Deliverable 2.