**SW Engineering  
CSC 648/848  
Milestone 2**

**Project Beehive  
Section 04 Team 02**

**October 11th, 2023**

**The Team:**

Team Lead - Satvik

Github Master - Matthew

Front-End Lead - Aaron

Back-End Lead - Sungmu

Front-End Lead - Ben

Scrum Master - Pearl

**Revision Table**

| Switched DB from MySQL to MongoDB, M1 to M2 | Changed Data Definitions to actually represent data we are using; users, job postings, etc | Initial list of functional requirements changed to suit revisions relating to non functional requirements, separated functions, and improved specifications |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Content and structure for Milestone 2 document for review**

1. **Data Definitions V2**

User Profiles - Profiles will contain data such as their emails, passwords (hashed to provide security), and any images they would like to have associated with their profile. Users will be able to sign up using Google and Facebook accounts in the future as well as Discord (Discord functionality is the only one implemented for prototype). Users will be saved into MongoDB.

Job Postings - Users that want to solicit positions in their company can do so by making a public posting. Postings may include salary, hours, company history, requirements, and any other information that would be necessary for a job listing.

Posts - Users can post anything they want that falls under guidelines for our application, as this is a project aimed for casual-business activity. Post may include media like videos, images, or gifs. Most posts will contain text, with a profanity filter to avoid nasty language. Posts will be stored in MongoDB.

API linking - As mentioned earlier, users will be able to link their Google, Facebook, and Discord accounts. These accounts will be present on their profile, with the data relating to their accounts stored in the database. API linking is based on secrets or API links from each platform, stored in backend code, hidden from users through hashing.

User privileges - Standard users will be able to make posts, edit their profiles, and send out applications for their organizations they choose to work for. Companies will have to pay a premium to have users be able to access their applications. Users cannot access API keys to other users, as that can cause scams.

1. **Initial List of Functional Requirements V2**

### **Requirement ID: 01**

* Title: User Registration
* Description: Enable users to register, create their profile.
* Priority: High
* User Story Reference: Users want to register their profile so that they can use the platform with their own personal information.

### **Requirement ID: 02**

* Title: Profile Management
* Description: Enable users to manage their profiles with privacy settings.
* Priority: High
* User Story Reference: Users want to manage their profile so that they can personalize their own experience on the platform.

### **Requirement ID: 03**

* Title: Job Posting
* Description: Employers can post job openings; users can search, view, and apply to jobs.
* Priority: High
* User Story Reference: Users want to view and apply to job postings that match my skills and preferences.

### **Requirement ID: 04**

* Title: Casual Posting
* Description: Users can share their experiences using pictures, videos, and quotes.
* Priority: High
* User Story Reference: Users want to access information in real time, to connect with others, and to stay in communities.

### **Requirement ID: 05**

* Title: Networking
* Description: Users connect with other students, alumni, and employers and utilize messaging and discussion boards.
* Priority: High
* User Story Reference: Users want to connect and communicate with other professionals to expand my network.

### **Requirement ID: 06**

* Title: Notifications and Alerts
* Description: Customizable notifications and alerts for users about new job postings and application status updates.
* Priority: Medium
* User Story Reference: Users want to receive timely alerts and notifications to stay informed about opportunities and updates.

### **Requirement ID: 07**

* Title: Job Matching and Recommendations
* Description: System recommends jobs to users based on their profiles, preferences, and search history.
* Priority: High
* User Story Reference: Users want to receive job recommendations that align with my career goals and skills.

### **Requirement ID: 08**

* Title: Learning Resources with Career Advice
* Description: A repository for career-related articles, videos, and advice from professionals and counselors.
* Priority: Medium
* User Story Reference: Users want access to resources and advice to make informed career decisions with provided multimedia such as articles, videos, etc.

### **Requirement ID: 09**

* Title: Event Calendar and Scheduling
* Description: Users can view, register for events, and book appointments with counselors or recruiters.
* Priority: Low
* User Story Reference: Users want to schedule appointments and register for events to enhance my career knowledge.

### **Requirement ID: 10**

* Title: Accessibility and Responsiveness
* Description: The platform should be accessible and responsive across devices and for users with disabilities.
* Priority: High
* User Story Reference: Users want to access the platform seamlessly, regardless of the device I am using or any disabilities I may have.

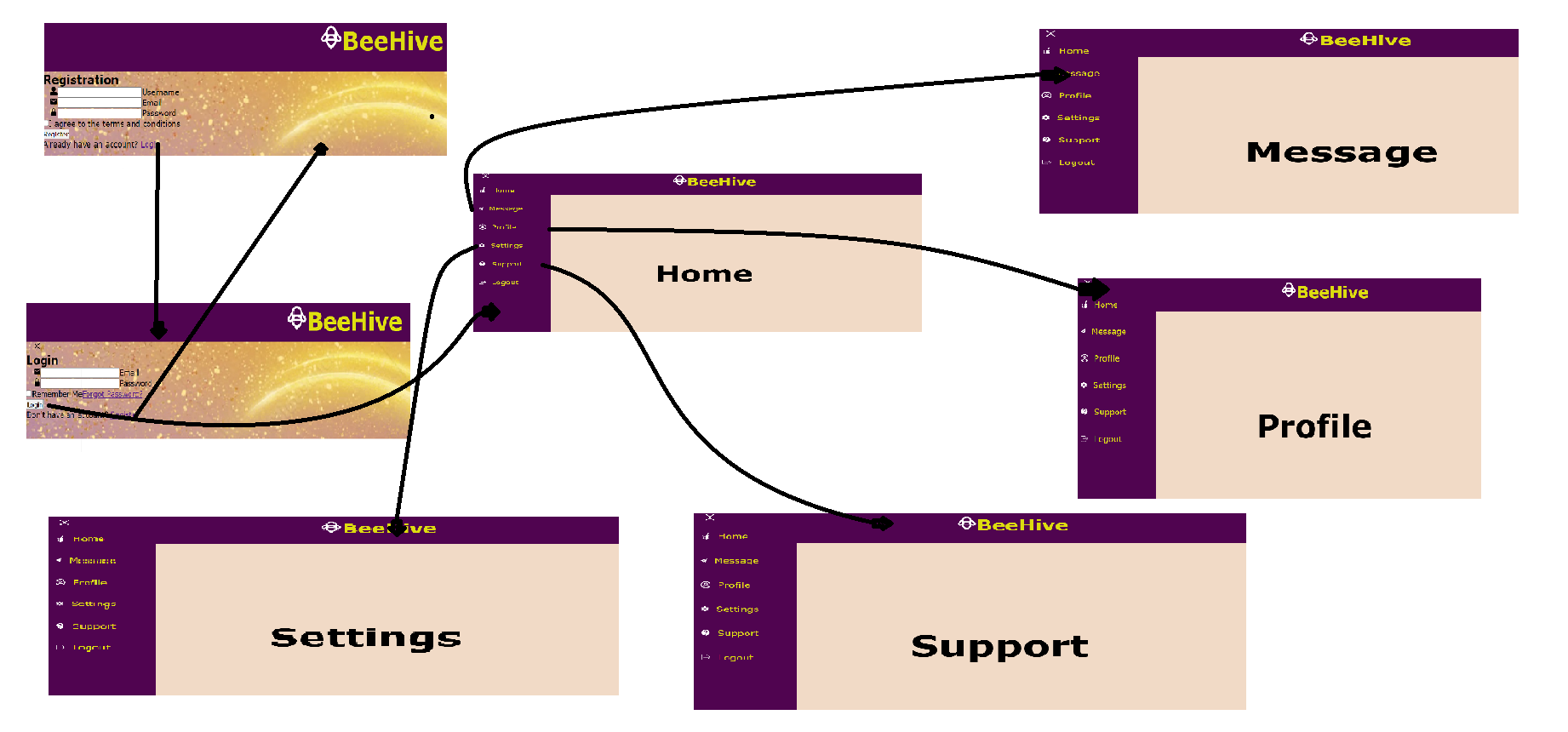
### **Requirement ID: 11**

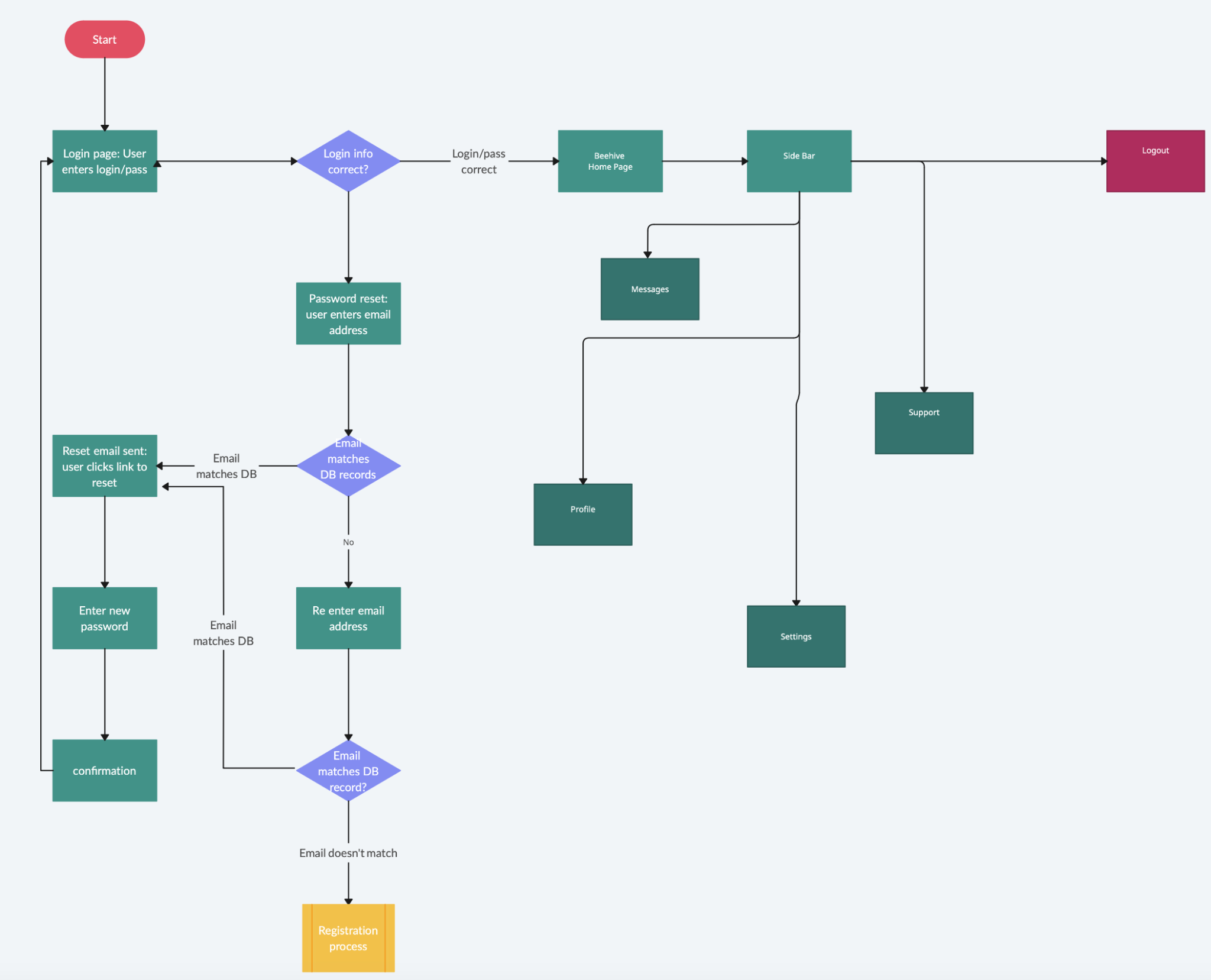
* Title: Advanced Search and Filtering
* Description: Users can perform advanced searches with filtering options to find jobs, people, and content. Filter options: location, date of job offered, experience level, company, job type, industry, etc.
* Priority: Medium
* User Story Reference: Users want advanced search options to find opportunities and connections more efficiently.

### **Requirement ID: 12**

* Title: Direct Messaging
* Description: Users should be able to send and receive private, secure messages to/from other users, allowing for direct communication between students, alumni, and employers.
* Priority: High
* User Story Reference: As a user, I want to send and receive private messages to communicate directly and securely with other users.

1. **UI Mockups and UX Flow**

****

****

1. **High Level Architecture | DB Organization**

Database - MongoDB

MongoDB, as a non-SQL database, uses JSON documents to store data. We have the following databases for our application:

Users - This document will contain the users’ emails, passwords, images associated with their profile, and when the accounts were created. This section will also contain the information necessary for connections to other accounts for different platforms such as Google, Facebook, and Discord.

Job Postings - Users that send out job positions on the app will have those links to jobs stored in a similar JSON document, however this database document will exclusively have job opportunities.

Posts- Posts that users make that do not have any links to job postings will be contained in this MongoDB document.

Add functionality - New users will be added to their respective database JSON documents.

Delete functionality - Users that choose to delete their accounts will have their information related to them on the database removed.

Search functionality - Users can search for other users that have accounts on the application.

APIs - ExpressJS

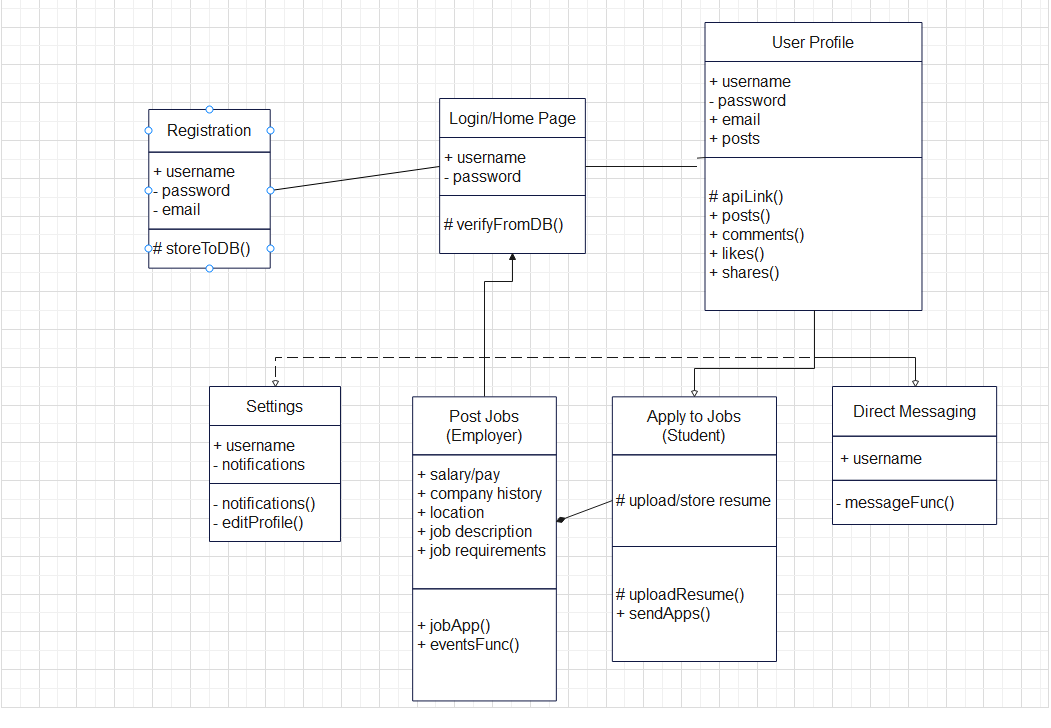
App has the ability to display cookies, using the “cookie-parser” express package. Backend implementation also has authentication functionality using PassportJS package. Provided emails and passwords, the backend will provide messages to console stating if authentication was successful or not, as well as throwing errors if a user is attempting to log in to an account that does not exist.

Third-party APIs include Discord, Facebook, and Google account functionality. PassportJS has easy packages to download for each of these three platforms, as passport-discord, passport-facebook, and passport-google respectively. Each third party has an associated schema.

Backend will communicate with frontend through API routes, an example would be our current implementation to handle media, as the route they would communicate by would be “/upload”. For authentication, such routes include: “/login”, “/register”, “/discord”, and “/discord/redirect”.

As mentioned in the revisions table, we switched databases from MySQL to MongoDB, as it provided us with better compatibility.

1. **High Level UML Diagrams**

****

1. **Key Risks**

### **Skills risks and mitigation plan**

### Risk: Transitioning from MySQL to MongoDB means the team has to adapt to a new database technology, which may lead to delays and potential mistakes due to unfamiliarity.

* + - As a mitigation, we schedule additional training sessions on MongoDB for the team. Allocate time for team members to experiment and get comfortable with MongoDB before integrating it into the project. Consider accessing online resources, tutorials, or even bringing in a MongoDB expert for a workshop.

### **Schedule risks**

* + Changes in project requirements or tasks might not be communicated or reflected immediately, causing misalignment in the team's efforts.
    - We can adopt a project management tool like Jira or Trello, ensuring real-time updates. Regularly review the project timeline and tasks in team meetings to ensure everyone is aligned with the latest changes.
  + Lack of a detailed team schedule for each member can lead to overlaps, missed deadlines, or tasks falling.
    - To resolve this, we will implement a comprehensive team schedule that highlights each member's detailed tasks. Use tools like Google Calendar or Notion to visualize and synchronize team availability and task deadlines.

### **Teamwork risks**

* + The absence of team members may lead to an unequal distribution of workload. If a member misses a meeting, they are assigned tasks that can be managed independently to avoid blocking other team members.
    - We can set up a system for team members to report their progress and any blockers regularly, ensuring that any lag due to absence can be quickly identified and addressed.

1. **Project Management**

As we work on creating our program, we've followed a clear and organized plan, making sure everyone on the team is on the same page. A big goal during this step was to set up a database that fits the special features of our platform.

During our daily scrum meetings, each member offers a concise update on their tasks related to the program “Beehive”. Making the most of Discord as our primary communication and task management medium, our daily scrum meetings were conducted effectively. The Team Lead ensured that our discussions were structured and directed towards the consistent forward motion of our project.

Our Front-End Lead accomplished a significant milestone by creating a login page, an important feature ensuring user accessibility. Then, the Github Master and Team Lead worked together to connect this new page to our main system. This big move helps make sure everything looks good and works well together on our platform.

Despite unexpected challenges, we were able to stay committed to communication. Our ability to adapt and plan our tasks well has been key to bringing our application closer to our goal for milestone 2.