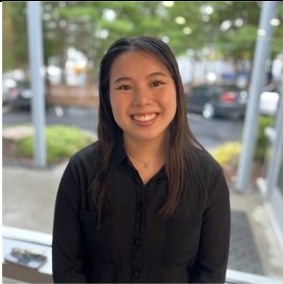




**INDY 7 — Mentoring App**  
**Final Report**  
**CS 4850 - Section 02 / 03 – Fall 2024**

**Dec 2, 2024, Professor Perry**

**GitHub Link:** <https://github.com/KSUMentorshipMatching/INDY-7-Mentorship-App>

**Website Link:** <https://ksumentorshipmatching.github.io/INDY-7-ProjectWebsite/>

 Trisha Nguyen (Section 03) Team Leader/ Documentation	 Khang Nguyen (Section 03) Developer	 John Huynh (Section 02) Developer
---	---	---

**Project Team**

<b>Roles</b>	<b>Name</b>	<b>Major responsibilities</b>	<b>Contact (Cell Phone)</b>
Team leader	Trisha Nguyen	Documentation; project management; communication; conflict resolution	678-640-7750
Team members	Khang Nguyen	Developer	470-558-6808
	John Huynh	Developer	678-704-5619
Advisor / Instructor	Sharon Perry	Facilitate project progress; advise on project planning and management.	770-329-3895

Total Lines of Code: 2,821 Lines

Total Project Components / Tools used: 10

Total Man Hours: 192 hours

## Table of Contents

1. Introduction .....	3
2. Planning.....	4
2.1 Objectives and Scope .....	4
2.2 Technology Selection .....	4
2.3 Project Timeline and Milestones .....	4
2.4 Risk Assessment and Mitigation.....	5
3. Requirements .....	5
3.1 Definitions and Acronyms .....	5
3.2 Design Constraints .....	5
3.2.1 Environment .....	5
3.2.2 User Characteristics .....	6
3.2.3 System.....	6
3.3 Functional Requirements.....	7
3.3.1 User Interface .....	7
3.3.2 Hardware Interface .....	7
3.4 Non-Functional Requirements.....	8
3.4.1 Security.....	8
3.4.2 Capacity .....	8
3.4.3 Usability.....	9
4. Analysis .....	9
5. Design .....	9
5.1 Design Considerations.....	9
5.1.1 Assumptions and Dependencies .....	9
5.1.2 General Constraints.....	10
5.1.3 Development Methods.....	11
5.2 Architectural Strategies .....	12

5.3	System Architecture .....	12
5.4	Detailed System Design .....	13
6.	Development.....	16
6.1	Project Architecture and Structure.....	16
6.2	Core Functionalities and Features .....	16
	<i>Landing Page</i> .....	16
	<i>Authentication and Authorization</i> .....	16
	<i>User Profile Management</i> .....	17
	<i>Mentor-Mentee Matching Algorithm</i> .....	17
	<i>Communication and Scheduling</i> .....	17
	<i>Dashboard</i> .....	17
6.3	Detailed Implementation .....	17
	<i>Backend Development</i> .....	17
	<i>Frontend Development</i> .....	19
	<i>Hosting and Deployment:</i> .....	21
	<i>Version Control:</i> .....	22
7.	Test (plan and report).....	22
7.1	Test Objectives: .....	22
7.2	System Overview: .....	22
7.3	Features to be Tested: .....	23
7.4	Testing Procedures and Pass/ Fail Criteria: .....	23
7.5	Test Report.....	26
8.	Version control .....	26
9.	Summary or Conclusion .....	26
10.	Appendix 1 - Project Plan .....	27
11.	Appendix 2 – Challenges and Future Plans.....	31

# 1. Introduction

For academic and professional development, mentorship can be pivotal to an individual's experience. Mentorship can provide those with support through knowledge exchange, skill enhancement, and personal growth. Inspired by the College of Computing and Software Engineering's Mentorship program at Kennesaw State University, the INDY 7-Mentorship App was created for facilitating mentorship. The INDY 7-Mentorship App aims to provide an efficient and user-friendly platform designed to connect mentors and mentees through preference-based matching system.

The INDY 7- Mentorship App was created by a project team of three Kennesaw State University students under the guidance of their Senior Project Professor, Sharon Perry. The application is the culmination of a semester long project, starting from August to December 2024. During that time span, the development of the Mentorship Application was organized by the stages of the Software Development Life Cycle (SDLC which includes the stages of planning, requirements, design, development/ implementation, testing, and maintenance. By adhering to the SDLC, the project team successfully developed key functionalities, such as login, account creation, account management, and account matching. Throughout the development of the application, some technical and logistical challenges were encountered, but regardless of these challenges, the app provides a strong foundation for facilitating mentorship at KSU.

The Mentorship App's system architecture can be organized into three subsystems: the front-end, the back-end, and the database. The front-end was developed utilizing HTML, CSS, JavaScript, and React. The back-end is implemented with Python (Flask), which handles the data processing, and MySQL, which stores the user data. RESTful APIs facilitate communication between these components. Hosting is supported and managed by PythonAnywhere for the Flask Application and MySQL database. GitHub Pages also hosts the project website, which contains information about the project, such as the final report and final presentation. Flask-Session is also used for secure session handling, ensuring token-based authentication and access control.

This final report outlines the development of the INDY 7-Mentorship App and explains in detail the app's architecture, functionalities/ requirements, challenges, and future plans. The following sections provide an in-depth look into the app's development process, technical implementation, challenges, and more.

## 2. Planning

It was important that a clear route was charted for the Mentoring App if the project's objectives were to be achieved within the August to December timeline. Planning included defining objectives, determining requirements, defining the tools of use, developing a project timeline for delivery, and mitigating risks.

### 2.1 Objectives and Scope

The major aim of this work was to develop a web application to connect mentors and mentees of Kennesaw State University via a preference-based matching system. The scope of work included designing the user interface, developing a platform on which users can input their preferences, and implementing a matching algorithm. Additional functionalities included notifications and profile management, which would enhance the user experience.

### 2.2 Technology Selection

The team assessed different technologies for the purpose of choosing the most appropriate development tools. Among the important decisions made are:

Frontend: At the beginning, React was chosen for modularity, but for on-time delivery, it had to be HTML, CSS, and JavaScript.

Backend: For simplicity and due to existing expertise within the team, Python with Flask was used instead of the initially intended PHP.

Database: MySQL was chosen for reliability and ease of use with the selected backend.

Hosting: PythonAnywhere was selected for hosting the Mentoring website while GitHub Pages was selected for hosting the project website.

### 2.3 Project Timeline and Milestones

A timeline was created to break down the project into manageable phases following the Software Development Life Cycle (SDLC):

Planning (Weeks 1-2): Requirement analysis, selection of the tech stack, and initial project setup.

Design (Weeks 3-4): Wireframes, database schema, and system architecture diagrams will be created.

Development (Weeks 5-10): Implementation of core features, including user authentication, profile management, and matching algorithm.

Testing (Weeks 11-12): Functional and performance testing, debugging, and validation of features.

Deployment (Week 13): Hosting the application and preparing for final presentation.

## 2.4 Risk Assessment and Mitigation

The team identified potential risks, including:

**Hosting Reliability:** PythonAnywhere's limitations were mitigated by exploring alternative hosting platforms for future iterations.

**Algorithm Complexity:** The Gale-Shapely algorithm required extensive debugging, which was planned for earlier in the development cycle.

**Password Recovery Challenges:** A fallback method was developed for users unable to recover passwords, with plans for future

## 3. Requirements

### 3.1 Definitions and Acronyms

#### **Definitions**

**User:** Any individual who interacts with the mentoring app.

**User Interface:** Everything a user interacts with on a screen i.e., buttons, menus, icons.

**Load Time:** The time it takes for a webpage to display its content after a request.

**Backend:** The behind the scenes aspects of an application i.e., servers and databases.

**Frontend:** The interface that users interact with directly.

**Scalability:** The ability of system to handle changes in workload and performance while still maintaining stability and cost.

**Version Control:** A system that keeps track of different versions of an application.

#### **Acronyms**

**API:** Application Programming Interface

**HTML:** Hypertext Markup Language

**CSS:** Cascading Style Sheets

**SQL:** Structured Query Language

### 3.2 Design Constraints

#### 3.2.1 Environment

- Cross Browser Compatibility
  - Website has to be able to function across all the major web browsers
  - Ex. Chrome, Firefox, Safari, Edge
- Responsive Design
  - Website should be fully responsive and usable on different screen sizes as well as different devices (ex. Mobile, desktop)
- Load Time

- The website should not take an excessive amount of time to load and perform functions.
- Scalability
  - The website should be able to handle different levels of traffic without any dip in performance.

### 3.2.2 User Characteristics

- Technical Proficiency
  - There are a wide range of uses with different levels of technical proficiency.
- Language Support
  - If there is a global audience (I.E international students) we will need to support different languages.
- Accessibility Needs
  - Different users with different abilities. Include things like keyboard navigation, screen reader compatibility, high contrast modes, etc.
- Age Range
  - Design should be intuitive for young and older students as well as young and older mentors.
- Privacy Concerns
  - Disclose privacy settings and options clearly, allowing users to choose what data they are willing to share.

### 3.2.3 System

- Backend Infrastructure
  - Choosing a backend that can handle the expected load
  - For our project GitHub pages should suffice, however with scaling, we could move over to a more scalable website hosting platform.
- Database
  - Ability to store user information in a secure, reliable, and scalable database such as MySQL.
- Third-Party API Integration
  - Ensuring the system is designed easily to make third-party API integration simple.
  - Version Control
  - GitHub will drive collaboration efforts and code changes
- Maintenance
  - Ensure prompt updates and bug fixes

## 3.3 Functional Requirements

### 3.3.1 User Interface

- Landing Page
  - Prompt the user to log in or sign up along with information about the program and its origin.
- Login Page
  - Prompts the user to log into the website with their credentials if they already signed up for an account.
    - Email field
    - Password field
    - Forgot Password button
    - Sign up button
- Register Page
  - Email (Not used as username) field
  - First Name field
  - Preferred Name field
  - Last Name field
  - DOB Dropdown
  - Password (Will require at least one uppercase letter and one symbol) field (will need to be hashed for privacy)
- Profile Page
  - Displays profile
  - Edit profile button
    - Edit password
    - Edit relevant information
    - Edit preferences
    - Edit profile picture
- Matching Page
  - Tinder esque design for mentee
    - Swipe left for not interested
    - Swipe right for interested
  - List of mentee candidates for mentor
    - List of mentees with link to their profile to view
    - Interested button
    - Uninterested button

### 3.3.2 Hardware Interface

- Since this is a web application, any internet accessible device can be used to view.



- Software Interface
  - Host – Github Pages
  - Website Languages – HTML, CSS, Javascript
  - Website libraries/frameworks – React (possible OpenAI / langchain)
  - Development IDE – Visual Studio Code
  - Version Control – Github
  - Database – MySQL
- Communication Interface
  - Data transfer between client and server occurs over the internet. No sensitive data is asked for on this page, however, security will be ensured with a secure connection to a database so that only we can manipulate the data.
- Authentication Interface
  - Student authentication can be done via API call from Sheer ID or ID.me

## 3.4 Non-Functional Requirements

### 3.4.1 Security

Users will be prompted to create a password when creating their account for security reasons. This is because users need to be able to log out and log back into their account. This is to confirm the identity of the user to ensure that they are the owner of the account they are trying to access. A strong password will be required to prevent hacking. To ensure a strong password is being used, the password will need to contain at least one uppercase letter and at least one special character.

There will also be a “Forgot Password?” feature that would allow users to change their password if the user forgot their password. It will do so by asking the user to enter in their email address that they used to create the account. If the email address matches an email address in our system, our system will send an email to that email address to change the password.

To discourage the creation of fake accounts, users will need to create an account with their student email to confirm that users are members of Kennesaw State University. Users cannot make an account with an email that has already been used to make an account.

### 3.4.2 Capacity

The website needs to have a large capacity because it needs to be able to store the user data. When an account is created, it will prompt the user to enter their information, such as their name, email, password, etc. The program needs to be able to store these users’ inputs and not lose the data. Multiple accounts will be created, resulting in a lot of user data that needs to be stored. The website needs to be able to store a lot of data and recall the data.

### 3.4.3 Usability

The website will have a clean design to ensure usability. The UI design will be simple to allow users to easily use and navigate through the website. Simple features, such as swiping left or right on potential candidates, will make it easy for users to use the website as well.

## 4. Analysis

Analysis focused on evaluating user needs, technical constraints, and external dependencies. This phase highlighted the following:

### 1. **Technical Needs:**

- The app should run on all major web browsers (e.g., Chrome, Firefox, Safari) and be compatible with mobile and desktop devices.
- The backend should handle expected loads without downtime or major performance issues, particularly for user profile storage and matching.

### 2. **User Characteristics:**

- Users range from students to professionals, with varied technical skills and accessibility needs.
- Account security and privacy are paramount, particularly around password protection and user preferences.

### 3. **Time Constraints:**

- Limited to a four-month development cycle due to academic schedules, team members had to balance other course requirements.
- Only free resources were available for the project, resulting in some limitations in infrastructure and tools.

## 5. Design

### 5.1 Design Considerations

#### 5.1.1 Assumptions and Dependencies

##### ***Software and Hardware Assumptions:***

- Web Browser
  - Website is assumed to be accessed using modern web browsers such as Chrome, Edge, Firefox etc.

- Hardware
  - Users are assumed to have a device capable of running a modern web browser.
- Server Infrastructure
  - The website relies on a stable scalable server infrastructure such as GitHub Pages, AWS, Google Cloud etc.

***Operating System Assumptions:***

- Supported OS's
  - Website is assumed to be compatible with major operating systems such as Windows, macOS, Linux, iOS, and Android.
- OS Versions
  - Website is expected to function properly on the newest versions of an operating system.
- Security
  - Users are assumed to have an OS with up to date security to prevent vulnerabilities.

***End-User Characteristics Assumptions:***

- Technical Proficiency:
  - Users are assumed to have basic internet navigation skills.
- Internet Access:
  - Users are assumed to have reliable internet access with sufficient speeds to load the website's content and videos.
- Device Usage:
  - Users can access the website from multiple various devices.
- Language:
  - Users are assumed to be proficient in English to read the information on the website.

## 5.1.2 General Constraints

***Time Constraints***

- **Four-month window:** This project began in August 2024 and will be completed by December 2024. Time is a constraint because four months is the allotted time for the start and end of this project's Software Development Life Cycle.
- **Time dedicated to other courses:** As full-time students at KSU, the members of this group will dedicate some of their time to other courses as well. This is another time constraint because the time the members can dedicate to this project has been reduced.

***Availability or volatility of resources***

- **No funding:** This project has no funding; therefore, our members are limited to utilizing only free resources and free tools.

***Security requirements***

- **Strong password:** Users will be prompted to create a password when creating their account for security reasons. This is because users need to be able to log out and log back into their account. This is to confirm the identity of the user to ensure that they are the owner of the account they are trying to access. A strong password will be required to prevent hacking. To ensure a strong password is being used, the password will need to contain at least one uppercase letter and at least one special character.
- **Forgot password feature:** There will also be a “Forgot Password?” feature that would allow users to change their password if the user forgot their password. It will do so by asking the user to enter in their email address that they used to create the account. If the email address matches an email address in our system, our system will send an email to that email address to change the password.
- **KSU email/ unoccupied email required:** To discourage the creation of fake accounts, users will need to create an account with their student email to confirm that users are members of Kennesaw State University. Users cannot make an account with an email that has already been used to make an account.

#### ***Verification and validation requirements (testing)***

- **Verification of email and password:** Users will need to make an account with their KSU email address and password. Our program would need to be able to verify that the given KSU email is a valid KSU email when creating the account. Our group would need to test if our program is able to verify if the given KSU email is valid when creating the account. Our program needs to be able to verify the password when logging back in as well.
- **Saving user data:** Our group will create accounts and enter user input. Our group will test to see if user data is saved regardless of logging out and logging into the account.
- **Testing functionalities:** Our group will also need to test the functions of the website, such as swiping left and right, and seeing if those functions register the user input correctly.

### 5.1.3 Development Methods

#### ***Development Method: Object-Oriented Design (OOD) and Agile***

The application will be created with Object-Oriented Design principles. Object-Oriented Design focuses on creating a system around objects. These objects represent both data and the operations performed on that data. This approach leverages principles that allow for modular, reusable, and maintainable system design.

Additionally, the development process will follow Agile methodologies. Aspects from Agile practices, such as iterative improvements and frequent delivery, will address potential issues promptly and provide an adaptable environment. This approach ensures that the system remains responsive to user and developer needs.

## 5.2 Architectural Strategies

### ***Choice of Programming Language and Framework***

- Decision: The mentoring app will use JavaScript, React, HTML, and CSS.
- Reasoning: React provides an easy to use Framework. JavaScript, HTML, and CSS, provide a well-known foundation for building interactive web applications.

### ***External Data Storage***

- Decision: Data will be stored using MySQL with backup and recovery mechanisms.
- Reasoning: The use of MySQL ensures reliable data management with established practices for backup and recovery.

### ***Future Plans for Extending or Enhancing Software***

- Decision: Design the app architecture to be modular, which allows for easy integration of new features.
- Reasoning: Modular Design allows for updates without significant modifications of existing code. This aligns with the goal of maintaining flexibility for future updates.

### ***Communication Mechanisms***

- Decision: The app will use RESTful APIs for communication between the frontend and backend.
- Reasoning: RESTful APIs are simple to implement and widely supported. It allows for straightforward integration and maintenance.

### ***Error Detection***

- Decision: The app will incorporate error handling mechanisms including try-catch blocks and error messages.
- Reasoning: Error messages provide clear explanations to users when errors occur. Try-catch blocks will let the application catch errors before the program crashes.

### ***User Interface Paradigms***

- Decision: The mentoring app will adopt a modern, responsive design paradigm that focuses on clarity and ease of use.
- Reasoning: A clean and intuitive interface minimizes user confusion and enhances ease of navigation.

## 5.3 System Architecture

### ***Frontend Subsystem***

- Front-end is responsible for the User Interface and User Experience of the website. It handles the interactions between the user and system. It aims to be responsible but also visually appealing
  - Components:

- UI/UX layer
- Client-Side logic

### ***Backend Subsystem***

- Handles the business logic, data processing and interaction handling. It is the backbone of the website.
  - Components:
    - API Layer
    - Business Logic Layer
    - Database Interaction

### ***Database Subsystem***

- Responsible for storing and managing all the data used by the website. This includes user data, content, and other critical information.
  - Components:
    - Presentation Layer
    - Business Layer
    - Staging Layer

### ***Authentication and Authorization Subsystem***

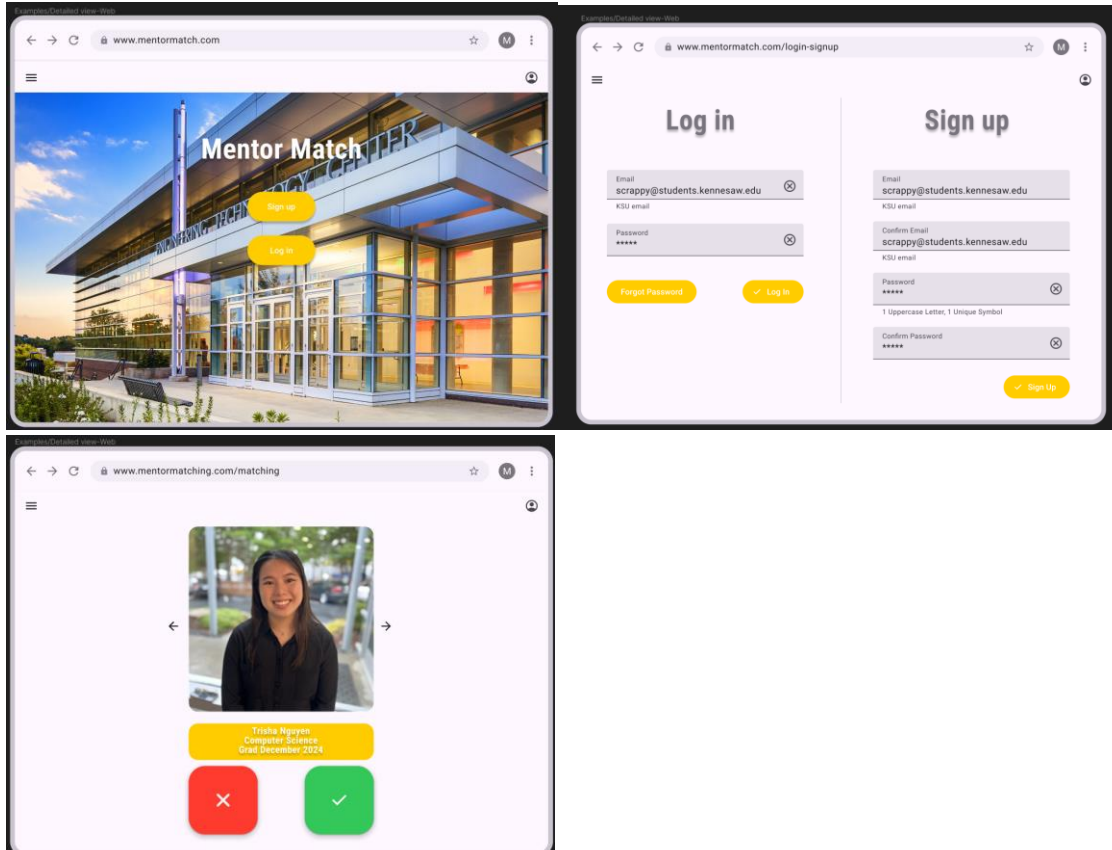
- Manages user identity and access control making sure that only authorized users can access specific parts of the website.
  - Components:
    - User Authentication:
    - Handles user login, registration, and session management. Uses something like OAuth2 or JWT.

## **5.4 Detailed System Design**

### ***Front End***

- Classification:
  - Component Type: Subsystem
  - Modules: UI/UX Layer, Client-side logic
- Definition
  - Purpose: The frontend system manages the presentation layer of the website, providing the user interface and handling interactions with users. It ensures that the experience is responsive, accessible, and visually appealing.
  - This subsystem serves as a point of interaction between the user and the system enabling user actions.
- Constraints
  - The frontend will operate on modern web browsers that comply with web standards (HTML5)

- The website must load quickly to maintain user engagement and must be compatible across devices and screen sizes.
- Resources
  - Browser memory, cache, and external assets (CSS, JS files, images)
  - Race conditions from API calls or when rendering React components. State management tools can help with these issues.



### Back End

- Classification
  - Component Type: Subsystem
  - Modules: API Layer, Business Logic Layer, Database Interaction
- Definition
  - Purpose: The backend is responsible for processing business logic and interacting with databases and sending data to the front end.
  - It acts as the backbone that ensures the app works as it is supposed to.
- Constraints
  - The backend can process requests concurrently and maintain data integrity
  - It must respect API rate limits and restraints on how many concurrent connections there can be
  - Must have secure ways to access sensitive data

- Resources
  - Server memory, processing power, databases, networking
  - Concurrent writes to the same database must be handled as transactions to prevent data corruption.

### **Database**

- Classification
  - Component Type: Subsystem
  - Modules: Presentation Layer, Business Layer, Staging Layer
- Definition
  - Purpose: It is responsible for manipulating data required by the system. This can include user profiles, content, configurations etc.
  - It ensures that the data management stays persistent and organized for the application
- Constraints
  - Data integrity must be maintained during updates and deletions
  - Backup constraints due to storage limitations
- Resources
  - Database engine (MySQL), backups, disk storage
  - Possible deadlock depending on how transactions are managed.

### **Authentication and Authorization**

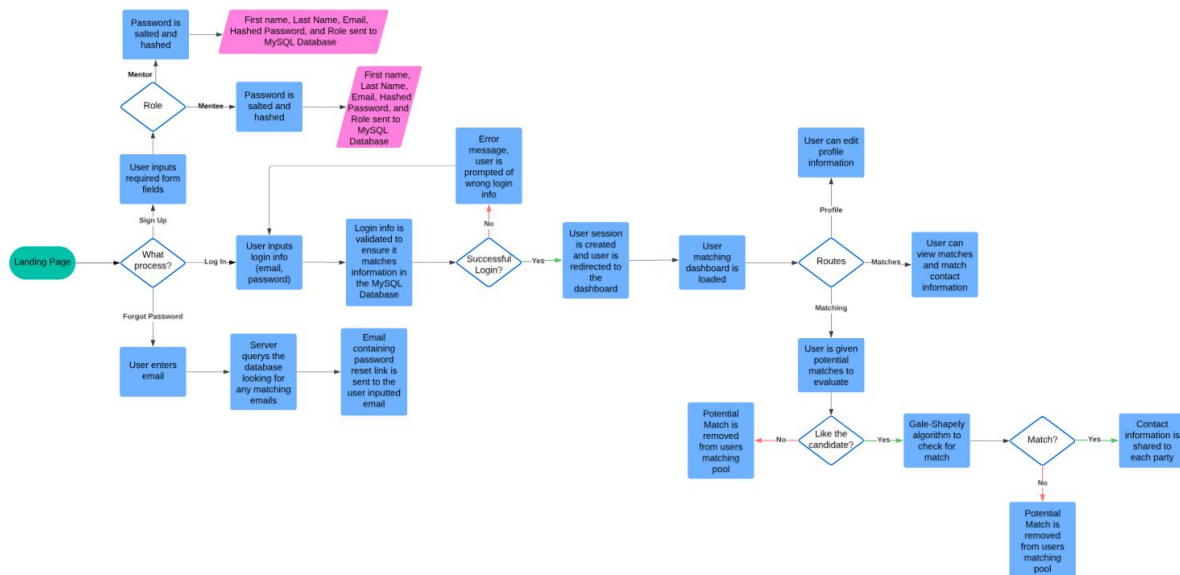
- Classification
  - Component Type: Subsystem
  - Modules: User Authentication
- Definition
  - Purpose: Manages user identities, controls access to resources, and implements user session management
  - It makes sure that only valid users can access certain resources on the website
- Constraints
  - Users must provide valid credentials matching stored data.
  - Brute force attacks are possible which would prompt a rate limit on login attempts
    - Must manage a session so that brute force attacks cannot bypass the rate limit
- Resources
  - User credentials storage (securely via hashing), session tokens
  - Race conditions can occur by concurrent login attempts. This can be solved by synchronizing the login attempts.



## 6. Development

### 6.1 Project Architecture and Structure

The project employs a client-server architecture, where the backend serves as a data source for the frontend interface. Technologies include HTML, CSS, and Javascript for the frontend interface with Python and Flask used to connect to the MySQL database. MySQL is used to store information on users, sessions, and matching data. The file structure is organized into backend and frontend directories, each containing controllers, models, and routes. The project also includes configuration files for environment settings, database access, and third-party integrations.



### 6.2 Core Functionalities and Features

#### Landing Page

- Users will land on a page with a nice graphical interface where the user will be prompted to either log in, create an account, or retrieve a forgotten password. Signing up will require a validated form where the password gets hashed and stored in the MySQL database along with email, first name, last name, and whether they are a mentor or mentee. This information will be queried during login as well, to ensure that the user is logging in to the right account. When the user logs in, it creates a session for them where they can access pages that only those with accounts can access.

#### Authentication and Authorization

- Users are authenticated using a token-based system. Role-based access control (RBAC) defines the different permissions for mentors, mentees, and admins, restricting certain features based on the role assigned to the user.

### *User Profile Management*

- Users create profiles with fields specific to either mentors or mentees, such as skills, preferences, and goals. This feature includes editable profiles that update real-time in the MySQL database.

### *Mentor-Mentee Matching Algorithm*

- The app's core functionality is the matching algorithm, which utilizes the Gale-Shapely algorithm to match mentors and mentees together. Each party (mentor or mentee) will rank their candidates and propose their top choice from the other party. The other party will then evaluate any proposals they have and choose the one they prefer the most. This will be iterated through each participant until everyone has a match. This algorithm is guaranteed to find a stable match meaning no one would rather be with someone else than their current match.

### *Communication and Scheduling*

- When matches are made, mentors will be able to exchange their contact information previously input into the database to connect and begin their mentor/mentee relationship.

### *Dashboard*

- The dashboard is the main hub for matching as this is where the user will evaluate new potential matches. The dashboard contains routes to the profile page where users can edit their profile and add preferences, goals, or skills to their profile. Additionally, the dashboard will contain contact information for matches that the user has made to facilitate connections.

## 6.3 Detailed Implementation

### *Backend Development*

The backend of the INDY-7 Mentorship App is built using Flask in Python, with a MySQL database for data storage and retrieval. This setup is designed to provide core functionalities such as user registration, login, profile management, and a simple session-based dashboard. Below is a breakdown of the major components and implementation specifics.

#### 1. Database Connection and Configuration

- The Flask app connects to the MySQL database using the `mysql.connector` library. Database credentials (e.g., host, user, password, and database name) are securely loaded from a text file (`dbInfo.txt`) using the `loadDbInfo` function, which reads each line and uses these credentials to establish a MySQL connection. A context manager within each request ensures that the connection is refreshed if disconnected, maintaining smooth interactions with the database.

## 2. User Authentication and Session Management

- **Registration (Sign-Up):** New users can register as either mentors or mentees via the `/sign_up` route, which processes form data from the frontend. User passwords are hashed with an additional salt ("69ggez") using MD5 hashing to ensure security before storage. Based on the user's role selection (mentor or mentee), registration information is stored in either the mentor or mentee table.
- **Login:** The `/log_in` route handles both GET and POST requests. Upon login, the user-provided password is salted and hashed, then checked against the stored hash in the database. Successful authentication sets a session variable containing the user's email, allowing session-based access to other parts of the app.
- **Session Handling:** The app uses Flask-Session with filesystem-based sessions. The session type and session key are configured for added security, enabling persistent user login between requests. The email of the logged-in user is stored in the session and is used to check access for protected routes.

## 3. Key Endpoints and Route Handling

The backend is organized with route functions, each of which serves a specific page or action:

- **Home Page (/):** Displays the homepage (e.g., `index.html`) for new and returning users.
- **Sign-Up Page (/sign\_up):** Processes user registration with different handling for mentors and mentees.
- **Login Page (/log\_in):** Authenticates users, initiates a session, and redirects them to the dashboard if credentials are valid.
- **Dashboard (/dashboard):** Provides a personalized dashboard, accessible only to logged-in users. If no session is active, the user is redirected to the login page.
- **Profile Page (/profile-page):** Displays user profile information based on their email stored in the session.
- **Logout (/log\_out):** Clears the session to log out the user and redirects them to the home page.

## 4. Password Security

Passwords are hashed using MD5 combined with a static salt. Although this method provides basic hashing, further security enhancements could include more robust algorithms such as `bcrypt` or `Argon2`, along with dynamically generated salts to strengthen data protection.

## 5. Template Rendering and User Interface Integration

Flask's `render_template` function loads HTML templates for each route, providing the necessary frontend views. For example:

- `index.html`: Serves as both the homepage and the login page, depending on the context.
- `dashboard.html`: Provides access to mentor or mentee functionalities based on user roles.

- create-page.html, forgot-page.html, and profile.html: Offer pages for account creation, password recovery, and profile management, respectively.

## 6. Database Interaction with Context Managers

Each interaction with MySQL is wrapped in a context manager to manage cursors efficiently. Queries are executed based on user roles and inputs, and transactions are committed only when necessary to maintain database integrity. Parameterized SQL queries are used to avoid SQL injection vulnerabilities.

## 7. Flash Messages and User Feedback

The app uses Flask's flash function to provide user feedback, such as login errors. Messages are displayed on the frontend to guide users if, for instance, login credentials are invalid, enhancing the user experience with clear communication.

## *Frontend Development*

The frontend of the INDY-7 Mentorship App is built using HTML, CSS, and JavaScript. It offers a user friendly and responsive interface that allows for easy navigation.

### 1. Landing Page

The landing page welcomes users, allowing them to log in, create an account, or recover their password.

- Login Form: A flexbox-based form with email and password input fields, each with visual icons
- Styling and Visual Design: An external CSS page (index.css) is used for layout and design.
- Full-Screen Video Background: A looping <video> with autoplay and muted attributes takes up the background of the page. The video is sourced through Flask.
- Externals Icons and Fonts: Utilizes icons from boxicons.com and Google Fonts for the "Roboto" typeface.
- Real-Time Flash Messages: Error messages are displayed with Flask in styled <p> elements.

### 2. Password Recovery Page

The password recovery page allows users to request an email to change their password.

- Animated Gradient Background: An animated gradient is applied to the background of the page.
- Styling and Visual Design: An external CSS page (forgot-page.css) is used for layout and design
- Externals Icons and Fonts: Utilizes icons from boxicons.com and Google Fonts for the "Roboto" typeface.

- **Recovery Form:** A flexbox-based form containing an email input to send password recovery procedures to.
- **Reset Link Button:** A styled button triggers the form submission initiating the password reset process.
- **Back-to-Login Navigation:** Linked text that allows users to navigate back to the login page

### 3. Create Account Page

The create account page allows users to register for the app as either a mentor or a mentee. This page requests user specific information to be used later for mentor matching.

- **Account Creation Form:** A flexbox-based layout that includes fields for the first name, last name, email, confirm email, password, and confirm password
- **Styling and Visual Design:** An external CSS page (create-page.css) is used for layout and design
- **Animated Gradient Background:** A full-page gradient background is used to enhance visual appeal.
- **Email and Password Validation:** JavaScript-based form validation for matching email passwords via `validateEmail()` and `validatePasswordRew()`
- **Sign Up Button:** A styled button for submitting the form, triggering the sign-up process up
- **Back-to-Login Navigation:** Linked text that allows users to easily navigate back to the login page
- **User Role Selection:** Radio buttons allow users to select between "Mentee" or "Mentor" roles.
- **Externals Icons and Fonts:** Utilizes icons from boxicons.com and Google Fonts for the "Roboto" typeface.

### 4. Dashboard

The dashboard is the main hub, displaying information such as who users have matched with, links to other areas of the app, and the main swiping functionality.

- **Layout:** The page utilizes a flexbox layout with a centered swipe section, a banner section, and a match's section
- **Styling and Visual Design:** An external CSS page (dashboard.css) is used for layout and design
- **User Profile and Menu Icons:** The top banner includes interactive icons for navigating the menu and accessing the user's profile page
- **Match Listings:** The match-container section displays cards of mentor/mentee profiles that users have matched with

- **Dynamic Card Loading:** The swipe-container dynamically loads mentor or mentee profiles through JavaScript (dashboard.js)
- **Swipe Interaction Buttons:** Two swipe buttons, "Dislike" and "Like", are placed to the left and right of profile cards
- **User Information:** Each user-card holds their name, email, phone number, and role that will be used for future contact

## 5. Profile Page

The profile page lets users view and update personal information, such as skills, preferences, and user details.

- **User Profile Layout:** A flexbox-based layout displays the user's profile image, bio, preferences, and skills
- **Styling and Visual Design:** The page uses external CSS (profile.css) for layout and design, with icons from boxicons.js for interactive elements.
- **JavaScript Integration:** The page uses external JavaScript (profile.js) for handling dynamic loading and other functionalities of the page.
- **Profile Information Display:** The user's name, role, bio, and profile image are displayed within a flexbox at the top of the profile.
- **Preferences and Skills:** A list of preferences and skills are displayed within flexboxes with options to enable or disable each one
- **Dynamic Content Loading:** Preferences and skills are dynamically loaded through profile.js
- **Interactive Elements:** Toggle indicators are next to each preference and skill button which allow users to enable or disable them according to their needs.

### *Hosting and Deployment:*

#### 1. Hosting the database:

- PythonAnywhere.com is used to host the database. PythonAnywhere was chosen over other databases, such as AWS and Oracle, because the cost to use it was free.

#### 2. Hosting the Project website:

- The project website is hosted by Github Pages through the team's Github account.

#### 3. Hosting the Mentor Match website:

- PythonAnywhere.com is used to host the Mentor Match website. The Mentor Match website is a Flask website and PythonAnywhere hosts Flask websites.
- PythonAnywhere was chosen over Github Pages since Github Pages does not host Flask websites. Additionally, Github Pages can only host one website at a time per account and the team's Github account is already hosting the Project website.

#### 4. Deployment of the Project website:

- The Project website can be access via any web browser with an internet connection using this URL link: <https://ksumentorshipmatching.github.io/INDY-7-ProjectWebsite/index.html>

#### 5. Deployment of the Mentor Match website:

- The Mentor Match website can be access via any web browser with an internet connection using this URL link: <https://ksumentorshipapp.pythonanywhere.com/>

#### *Version Control:*

##### 1. Github:

- The team had a shared Github account that was used as the repository

##### 2. Github Desktop

- Github Desktop was used to push and pull updates to the code on different devices. If one member made an update and pushed it, another member could pull that and have the same updates that were just made in almost an instant

*(Future plans for additional improvements moved to Appendix 2)*

## 7. Test (plan and report)

### 7.1 Test Objectives:

Testing of the INDY7 Mentorship App will be done to see if the application meets the following requirements:

- Support for Mentor and Mentee matching functionality
- Mentor and Mentee registration and profile creation work correctly
- Security and data protection controls are present
- User-friendly interface and accessible to all users
- Mobile responsiveness to ensures usability across different devices
- Error handling functions work correctly
- Matching algorithm correctly matches compatible mentors and mentees
- Capacity and load testing to ensure proper support for the expected number of users
- User activity tracking for administrator statistics

### 7.2 System Overview:

The INDY7 Mentorship App is a web application developed to streamline the connection process between mentees and mentors. The application analyzes recommends and matches mentors and mentees based on their skills and preferences. The INDY7 Mentorship App will serve as a platform for users to establish mentoring relationships.

## 7.3 Features to be Tested:

Below are the features of the mentoring app that will be tested:

- Login
- Create Account
- Password Recovery
- Profile Page Functionality
- Landing Page Content
- Matching Page Functionality
- Security and Data Protection
- User Interface / Usability
- Capacity / Load
- Error Handling
- Notification System

## 7.4 Testing Procedures and Pass/ Fail Criteria:

1. Login:
  - a. Testing Procedures:
    1. Enter valid credentials (email and password) and log in successfully
    2. Test login with invalid credentials (wrong email/ password)
  - b. Pass Criteria:
    1. Successful login with valid credentials
    2. Error message saying, “Invalid email/ password. Try again” appears when credentials are incorrect
  - c. Fail Criteria:
    1. The system allows login with invalid credentials
    2. Error message does not appear when credentials are invalid
2. Create Account:
  - a. Testing Procedures:
    1. Go to the sign-up page
    2. Enter valid information for all input fields (name, email, password, etc.)
    3. Submit form to create account
  - b. Pass Criteria:
    1. Account is created successfully with valid information
  - c. Fail Criteria:
    1. Account is not created successfully with valid information/ information is not saved correctly
    2. Account is created / form submits with incomplete data without error
3. Password Recovery:
  - a. Testing Procedures:
    1. Go to the “Forgot Password” page
    2. Test using invalid/ unregistered email address
    3. Enter a valid registered email address



4. Use the link in the “Reset Password” email to reset password
- b. Pass Criteria:
  1. The “Reset Password” email was sent successfully
  2. The password can be reset successfully using the link in the email
  3. Login successful with new password after resetting it
- c. Fail Criteria:
  1. Does not send “Reset Password” email
  2. Link to reset password does not work
  3. Password does not reset/ new password does not save
4. Profile Page functionality:
  - a. Testing Procedures:
    1. Log in and go to the profile page
    2. Edit profile information/ settings (name, preferences, etc.)
    3. Save changes
  - b. Pass Criteria:
    1. Profile information can be updated and saved successfully
    2. Changes made are saved and updated instantly
  - c. Fail Criteria:
    1. Changes are not saved
5. Landing Page Content:
  - a. Testing Procedures:
    1. Load the landing page. All the content loads properly (pictures, buttons, etc.)
  - b. Pass Criteria:
    1. All content loads properly
  - c. Fail Criteria:
    1. Missing content/ content does not load properly
6. Matching Page/ Algorithm:
  - a. Testing Procedures:
    1. Log in and go to the matching page
    2. Can see users who are potential matches and their information (name, etc.) are shown
    3. Potential matches (mentors/ mentees) displayed are based off the user’s preferences
    4. Can select “Yes” (green button) or “No” (red button) on potential matches
  - b. Pass Criteria:
    1. Users who are potential matches are displayed and their information (name, etc.) are shown
    2. Page displays potential matches based on the user’s preferences
    3. Can select “Yes” (green button) or “No” (red button) on potential matches
  - c. Fail Criteria:
    1. Other users’ information is not displayed
    2. Potential matches shown are irrelevant/ not based on preferences
    3. Cannot select “Yes” or “No”
7. Security and Data Protection

- a. Testing Procedures:
    - 1. Test password storage by ensuring that passwords are hashed and not stored as plain text in the database
    - 2. Check if sensitive data is encrypted
  - b. Pass Criteria:
    - 1. Passwords are hashed and data is encrypted
  - c. Fail Criteria:
    - 1. Passwords are not hashed, and data is not encrypted
8. User Interface/ Usability
- a. Testing Procedures:
    - 1. Test ease of use
    - 2. Ensure that all interactive elements, such as buttons, are functional
    - 3. See if UI is visually appealing
  - b. Pass Criteria:
    - 1. Easy to navigate and visually appealing
    - 2. All interactive elements work
  - c. Fail Criteria:
    - 1. Difficult to use/ navigate
    - 2. Interactive elements do not work
9. Capacity/ Load
- a. Testing Procedures:
    - 1. Conduct load testing by simulating multiple users using the site simultaneously
  - b. Pass Criteria:
    - 1. The system does not crash/ the server does not fail
    - 2. Response times are not long (no longer than 5 seconds)
  - c. Fail Criteria:
    - 1. The system crashes/ the server fails
    - 2. Poor performance (long response times, etc.)
10. Error Handling:
- a. Testing Procedures:
    - 1. Test invalid user input
    - 2. Ensure error messages are shown
    - 3. Test how the system handles failures, such as server crashes
  - b. Pass Criteria:
    - 1. Appropriate and helpful error messages are shown
    - 2. Can handle errors without crashing
  - c. Fail Criteria:
    - 1. Does not show error messages
    - 2. Errors cause the system to fail
11. Notification System
- a. Testing Procedures:
    - 1. Notifications are sent after an action (new match, message notification, etc.)
    - 2. Can turn notifications on or off in settings

## b. Pass Criteria:

1. Relevant notifications are sent promptly
2. Users can turn notifications on or off in settings

## c. Fail Criteria:

1. Notifications are not sent promptly/ are irrelevant
2. Users cannot turn notifications on or off in settings

## 7.5 Test Report

Requirement	Pass	Fail	Severity
Login	X		High
Create Account	X		High
Password Recovery		X	Medium
Profile Page Functionality	X		Medium
Landing Page Content	X		Low
Matching Page/ Algorithm	X		Medium
Security and Data Protection		X	High
User Interface / Usability	X		Medium
Capacity / Load	X		Low
Error Handling	X		Low
Notification System		X	Low

## 8. Version control

GitHub was used to manage the version control for source code and documentation. Key aspects include:

- **Team Collaboration:** The team used a shared GitHub repository to manage code, ensuring consistent updates across members.
- **GitHub Desktop:** Enabled quick syncing of changes across devices, allowing team members to pull updates after each other's contributions.
- **Change Log:** Documented key updates, making it easier to trace any issues to specific changes.

## 9. Summary or Conclusion

The INDY-7 Mentoring App is a web application developed for connecting mentors and mentees at Kennesaw State University. This project ran from August 2024 to December 2024 under the guidance of Professor Sharon Perry. The Mentoring App has aimed to offer a smooth, concerted mentorship experience. The development used a modern tech stack, leveraged 2,821 lines of code

in writing, and was collaboratively managed over GitHub for version control and team synchronization.

The application has different features to ensure functionality and user accessibility. Upon opening, users are taken to a landing page that allows them to log in, sign up, or recover their passwords. Account authentication is secured through password hashing, and profiles can be managed to include personal details, skills, and preferences. The core of the app implements the Gale-Shapely matching algorithm for stable mentor/mentee pairings. The dashboard shows matches and provides a navigation hub, while notifications update users on changes. Security is enhanced with role-based access control, whereby permissions are assigned based on whether the user is a mentor or mentee. The system architecture is designed based on three main subsystems: the frontend, backend, and database. The frontend is developed with HTML, CSS, JavaScript, and React, giving much importance to responsive design and intuitive navigation. The backend, implemented on Python through Flask, takes care of business logic, data processing, and API interactions, while MySQL is used to securely store user data. RESTful APIs facilitate seamless communication between these components. Hosting is handled by PythonAnywhere for the Flask application and MySQL database, while the project's informational website is hosted on GitHub Pages. Secure session handling is handled with Flask-Session, including token-based authentication and access control.

The development process followed the Software Development Life Cycle, which includes the stages of planning, requirements, design, development/ implementation, testing, and maintenance. Functionalities include account creation, login, password recovery, profile management, and mentor-mentee matching. Extensive testing validated the system's performance, testing login, account creation, password recovery, and more. During testing, most functionalities passed testing successfully, but the requirements that failed were password recovery, security and data protection, and notification system. Various issues arose during development, including PythonAnywhere hosting reliability and the switch from PHP to Flask for better alignment with team experience. Also, the matching algorithm and password recovery feature are still in the development stage. Despite these obstacles, the project achieved its primary objective of facilitating mentor-mentee connections. Future plans for this project include transitioning the frontend to React for better modularity, stronger password hashing using methods like bcrypt, optimizing the matching algorithm with Jaccard similarity, and moving to a more stable hosting platform.

## 10. Appendix 1 - Project Plan

### Project Team & Roles/Responsibilities:

- **Trisha Nguyen (Team Leader/Documentation):** Oversees project management, documentation, communication, and conflict resolution.

- **Khang Nguyen (Developer):** Backend and frontend developer, responsible for coding and testing key components.
- **John Huynh (Developer):** Backend and frontend developer, responsible for building and maintaining functionalities.
- **Sharon Perry (Advisor):** Provides guidance on project planning, management, and technical issues.

**Contact Information:**

- Trisha Nguyen: 678-640-7750
- Khang Nguyen: 470-558-6808
- John Huynh: 678-704-5619
- Sharon Perry (Advisor): 770-329-3895

**Meeting Schedule and Communication Plan:**

- **In-Person Meetings:**
  - **When:** Every Wednesdays, 3:30 -6:30 pm
  - **Where:** KSU Marietta Campus, Engineering Technology building
  - **Objective:** Discuss project progress, assign tasks, and address any issues.
- **Outside Communication:**
  - **Method:** Text messaging and phone calls for quick updates.
  - **GitHub:** Primary platform for code sharing and version control.

### Milestones and Deliverables:

Milestone	Completion Date	Deliverable
Project Plan Document	09/01/2024	Detailed project plan outlining scope, objectives, and team roles.
Requirements and Design Completion	09/01/2024	Software Requirements Specification (SRS) and System Design Document (SDD).
Prototype and Presentation	10/15/2024	Demonstration of app prototype with basic UI and initial functionalities.
Backend Completion	10/17/2024	Flask backend connected to MySQL, supporting core app functionalities.
Frontend Completion	10/24/2024	Responsive-based UI, with profile and matching components implemented.
Product Testing Completion	10/31/2024	Functional testing of matching algorithm, login, and profile features.
Poster Board Completion	11/13/2024	Visual project summary for presentation purposes.
Final Report	12/02/2024	Comprehensive final report detailing the entire development process and results.

### Version Control Plan:

**Repository:** GitHub (<https://github.com/KSUMentorshipMatching/INDY-7-Mentorship-App>)

- **Source Code Management:** All project code is stored and managed on GitHub, ensuring version control and collaborative work.
- **GitHub Desktop:** Used for pushing and pulling updates, allowing team members to synchronize changes quickly and prevent conflicts.
- **Backup and Recovery:** GitHub's versioning maintains a record of all updates, allowing for recovery of previous versions if issues arise.

## Task Planning and Responsibilities

**Development Timeline (Gantt Chart):** *(For full Gantt chart and timeline details, refer to the attached project work plan file)*

Task	Assignee(s)	Expected Duration	Description
Initial Research	All members	1 week	Gather requirements and plan software functionalities.
Backend Setup	Khang Nguyen, John Huynh	2 weeks	Set up Flask, MySQL integration, and authentication.
Frontend Setup	Khang Nguyen, John Huynh	2 weeks	Design React-based user interface with basic page navigation.
Matching Algorithm	John Huynh	2 weeks	Implement Gale-Shapely matching algorithm for mentor-mentee connections.
UI and UX Enhancements	Trisha Nguyen	2 weeks	Refine interface design, add profile settings, and improve navigation.
Testing and Debugging	All members	2 weeks	Conduct testing, fix issues, and validate functionality and performance.
Documentation	Trisha Nguyen	Throughout	Maintain project documentation, including design, testing, and user guides.

## Collaboration and Communication

The team prioritizes active communication, maintaining transparency on progress and promptly addressing issues. GitHub serves as the main platform for collaboration, while phone and in-person meetings ensure that all team members stay aligned on project goals and updates.

## 11. Appendix 2 – Challenges and Future Plans

Throughout development, several challenges prompted adjustments:

- **Database / Web host Migration:** AWS costs led to PythonAnywhere adoption, a free alternative for Flask and MySQL hosting.
- **Web hosting:** PythonAnywhere can be inconsistent in its hosting and can show errors on non-buggy websites. In the future moving from PythonAnywhere to a more reliable web host will be necessary.
- **Language Migration:** Difficulties implementing PHP resulted in a change to Flask and Python to better suit the team's strengths. In the future, an implementation of PHP would allow for easier hosting.
- **Frontend Refactoring:** HTML front end is planned to be transitioned to React for modularity and performance.
- **Forgot Password Implementation:** Currently in progress, expected to be part of the final release.
- **Matching Algorithm:** Gale-Shapely implementation is ongoing, aiming to optimize match stability. Thoughts of pivoting towards calculating Jaccard similarity for matching users instead.

Future additions include enhancing email/password validation and expanding user feedback options for better app reliability.