Architecture and Design Document

for SMP Project

Developed by: Aishwary Sharma, Sarthak Daksh, Mohit Sharma, Vishesh Jain and Pankaj Jalote

## Architecture Design

Background

The main goal of this project is to make the Student Mentorship Program (SMP) at the Institute easier to administer, manage, and track. Currently, SMP uses various tools like spreadsheets, forms, and paper for different administrative tasks like selecting mentors, assigning mentors to mentees, tracking mentor-mentee interactions, etc. The aim is to develop a web application/portal that can be used to manage most of the key SMP administrative activities. The requirements for the portal have been specified in the SRS document.

Architectural Style

As the application is web-based, it is natural to design it as a 3-tier application. However, the back-end of this application has some components for specific services that are needed, e.g. mapping mentor to mentee, managing forms, managing meetings, etc. The frontend uses google's authentication service for user authentication (e.g. frontend typically makes the call to verify, and after verification backend uses it), and Django's ORM for connecting with the database. The architecture diagram is given below:
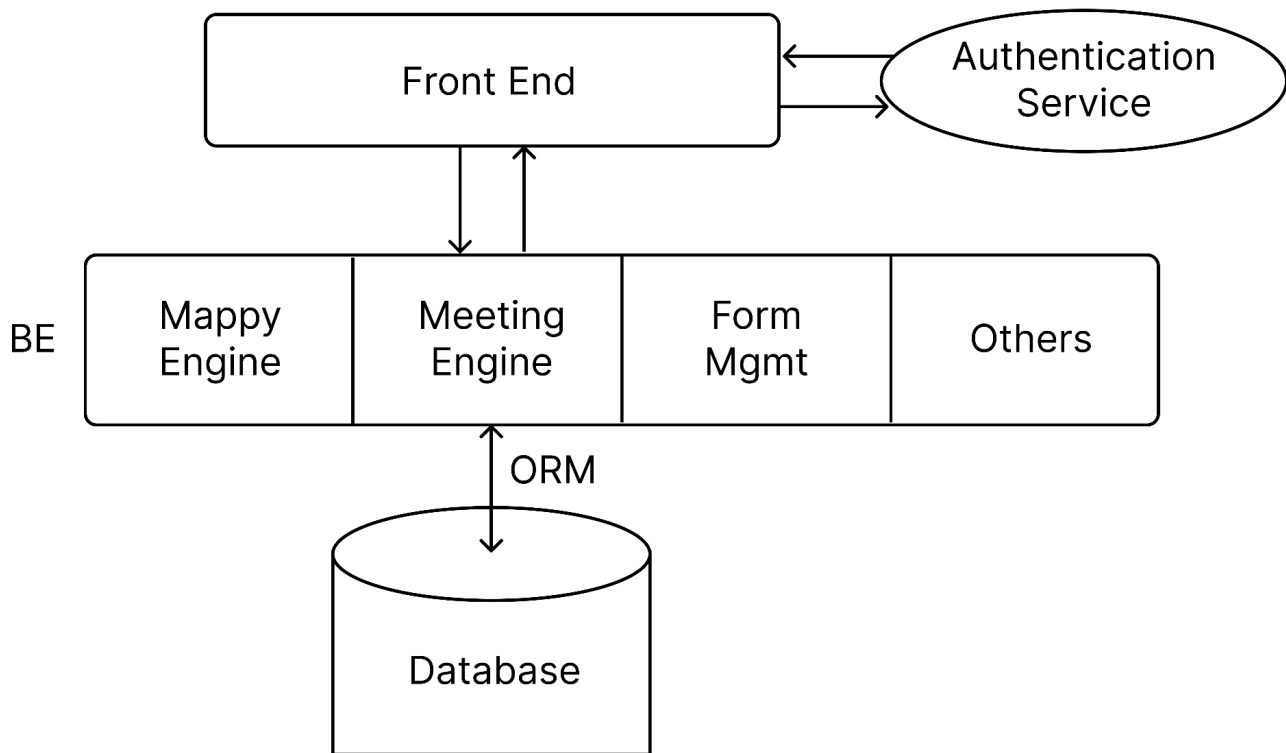
Brief description of the components is given below:

Front End. It is a set of HTML+ Javascript files (React code) for rendering various components and for calling the APIs, including google auth's APIs. These files together represent the screens the users see.

Back End. Back end has some main sub-components. As Django is being used, the routing of requests to the proper backend code (based on the URLs) is handled by Django. (For this, suitable configuration files have to be provided.)

- **Mapping Engine.** Provides the APIs for mentor-mentee mapping, downloading the mapping, editing the mapping, etc.
- **Meeting Engine.** Provides APIs for scheduling, updating, managing, etc. for meetings.
- **Form Management**. There are multiple forms. This component is for enabling/disabling the forms, getting the responses, sending notifications, etc.

- **(Models layer.** No separate model's layer/component is needed, as Python provides rich and easy to use functions to access / update the database.)

Note: Django requires all APIs to be in the views folder. For these components, their APIs are files in views. Therefore, these components are not directly represented as folders.

Note: Components are a set of related APIs, which are grouped under this component. But APIs are finally implemented individually as functions.

APIs in the different components:

- **Mapping Engine**: createMentorMenteePair, uploadCSV, addCandidate
- **Meeting Engine:** addMeeting, editMeetingById, deleteMeetingById, getMeetings, getAttendance, updateAttendance
- **Form Management**: submitConsentForm, getFormResponse, getFormStatus, updateFormStatus, sendConsentForm, getExcellenceAward, updateExcellenceAward, getMailSubjectAndBody
- **Others(CRUD operations)**: getAllMentors, getMentorById, getAllMentees, getMenteeById, deleteAllMentors, deleteMentorById, deleteAllMentees, deleteMenteeById, addMentor, addMentee, editAdminById, editMentorById, editMenteeById

Technology Stack and Other Choices

The technology selected for implementation is mentioned here, along with a brief rationale for the selection.
- Frontend:
    - HTML
    - CSS with Bootstrap: Bootstrap is used for UI component styling and layout. It provides pre-designed UI elements, grids, and responsive components, accelerating front-end development and ensuring a consistent user experience.
    - React with JavaScript language. We use React with JavaScript for efficient and dynamic user interface development.
- Backend:
    - Django. Django's built-in features, including authentication and its robust ORM, align well with our project's requirements for user management and database operations.
    - Nginx. It is a commonly used webserver with Django which can efficiently run Django based applications.
    - Python library Pandas for handling tabular data, including data import and export from spreadsheets.

- Database
    - PostgreSQL. PostgreSQL is chosen for its reliability and high performance, ensuring data integrity and scalability in our project. As most of the data is tabular, Postgres will work well.
- Source Code Control:
    - Git for version control
    - Github to host the code repository
- Security.
    - It was decided to use google authentication. This provides a reliable and off-the-shelf method for authenticating users. HTTPs is used for interaction between the frontend and the backend.
- Hosting.
    - The application will be hosted on a virtual host provided by the Institute. This will make it independent of other applications.

Evaluation

- The architecture style allows new features to be added in the front-end easily and providing new APIs, if needed, making it extensible.
- Security and access control is ensured by google's authentication. In addition, https will be used.
- Performance constraints are minimal - they will be met easily with a reasonable server to host the application.

High-Level Design

Design Summary

Summary of Frontend Screens

Note: Most of the screen names will be folders (or .js files) - names are similar to the screen names in the table.

Screen Name	Main Component	API it calls
User Authentication Screen	Login Options, User Authentication Fields	getFormStatus, getIdByEmail
Welcome Screen (Mentor)	New User Application, Application Status Messages	submitConsentForm, getFormStatus, addCandidate
Dashboard (Mentor)	Mentor's Profile, Assigned Mentees, Navigation Links	getMentorById
Meeting Schedule (Mentor)	List of Previous and Upcoming Meetings, Add Meeting Button	getMeetings, deleteMeetingById, editMeetingById
Meeting Details (Mentor)	Meeting Details, Update Meeting Details, Take View Attendance	N.A.
Schedule Meeting (Mentor)	Date and Time Selection, Meeting Details, Attendees, Schedule Button	addMeeting, getAttendance, updateAttendance
Dashboard (Mentee)	Mentee's Profile, Assigned Mentor, Navigation Bar	N.A.
Meeting Schedule (Mentee)	Previous and Upcoming Meetings	getMeetings,
Feedback Form Page (Mentee)	Feedback Form with Questions, Submit Button, Form Status Message	getFormStatus, menteeFilledFeedback
Dashboard (Admin)	Admin's Profile	N.A.
Mentees List Page (Admin)	Search Bar, Add/Remove Mentees, Upload CSV, Download CSV, Change Mentor, Filters	getAllMentees, editMenteeById, addMentee, deleteMenteeById, uploadCSV
Mentee Details (Admin)	Mentee's and their Mentor's Details	N.A.
Mentor List Page (Admin)	Search Bar, Add/Remove Mentors, Download CSV, Filters, Download Images	deleteMentorById, getAllMentors, addMentor

Mentor Details (Admin)	Mentor's and their Mentees' Details	N.A.
Meeting Schedule (Admin)	List of Previous and Upcoming Meetings of Admin and Mentor, Add Meeting Button	deleteMeetingById, getMeetings, editMeetingById
Meeting Details (Admin)	Meeting Details, Mark/View Attendance	N.A.
Schedule Meeting (Admin)	Date and Time Selection, Meeting Details, Participants Selection	addMeeting, getAttendance, updateAttendance
Form Page (Admin)	List of Forms, Enable/Disable Forms	getFormStatus, updateFormStatus,
Form Response Page (Admin)	Form Responses, Search and Filters options, Checkbox selection, Mentor-Mapping Button(Consent Form), Send Consent Mail button(Enrollment Form), Excellence Award	getFormResponse, createMentorMenteePair, sendConsentEmail

Summary of the Backend APIs

Note: Most of these APIs will show up as functions with the API name.

API Name	Input	Responses
getAllMentors		List of all mentors
getMentorById	id	Mentor details
getAllMentees		List of all mentees
getMenteeById	id	Mentee details
getIdByEmail	email	User ID
deleteAllMentors		Success/Error Message
deleteMentorById	id	Success/Error Message
deleteAllMentees		Success/Error Message
deleteMenteeById	id	Success/Error Message
addMentor	Mentor Details (Name, Roll no, department, year etc)	Success/Error Message
addMentee	Mentee Details (Name, Roll no, department,	Success/Error Message

	year etc)	
addCandidate	Candidate Details (Name, Roll no, department, year etc)	Success/Error Message
editAdminById	Updated Admin Details (Name, email etc)	Success/Error Message
editMentorById	Updated Mentor Details (Name, Roll no, department, year etc)	Success/Error Message
editMenteeById	Updated Mentee Details (Name, Roll no, department, year etc)	Success/Error Message
uploadCSV	CSV File containing mentee Details	Success/Error Message
addMeeting	Meeting Details (title, schedulerId, date, time, attendee, description, mentorBranches, menteeBranches, menteeList)	Success/Error Message
editMeetingById	Updated Meeting Details (title, schedulerId, date, time, attendee, description, mentorBranches, menteeBranches, menteeList)	Success/Error Message
deleteMeetingById	meetingId	Success/Error Message
getMeetings	role, id	List of previous and Upcoming meetings
getAttendance	meetingId	Dictionary of attendees and their attendance
updateAttendance	(attendees, meetingId)	Success/Error Message
createMentorMenteePair	Selected students list	Success/Error Message
submitConsentForm	A json containing user responses	Success/Error Message
getFormResponse	formType	Form Responses of the Given form type
getFormStatus		On/off status of the form
updateFormStatus	formId, formStatus	Success/Error Message
sendConsentForm	A list of students and emails subject and body	Success/Error Message
getExcellenceAward		List of Mentors, Their Scores and their status
updateExcellenceAward	List of selected students	Success/Error Message
getMailSubjectAndBody	type	Subject and body of the mail

The database design is quite straightforward - the schemas are given later. Djanjo ORM is used to interact with the database.

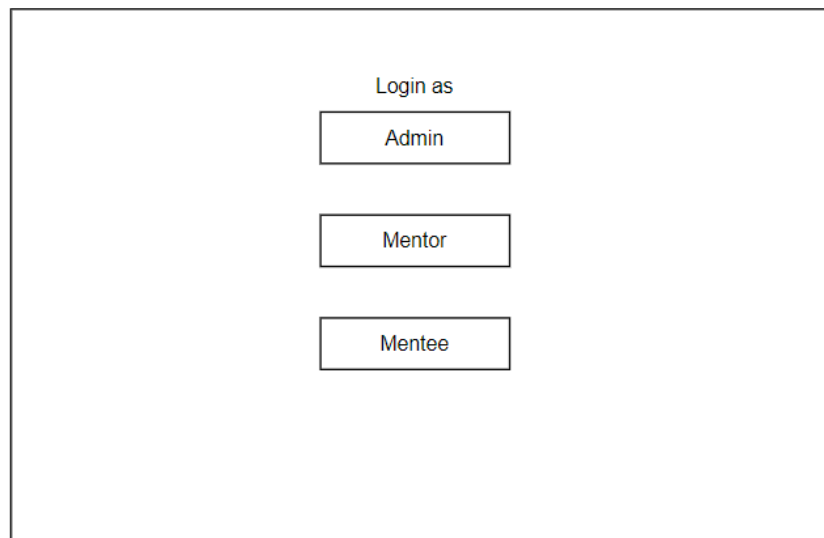## Front End Design Details

Overall technology:
-   HTML, CSS with Bootstrap, React with JavaScript

Component Library and State Management:
1. **Component Library:** Bootstrap for UI components. Bootstrap provides a set of styles and interactive components (buttons, forms, navigation bars, modals, alerts, and more) that can be easily customised and seamlessly integrated into our web application.
2. **State Management Library:** React's built-in state management.

—---

Common Screen

Login as

Admin

Mentor

Mentee

-   **Screen Name:** User Authentication Screen
-   **Purpose:** The Login Page is the initial access point for users to authenticate themselves into the SMP portal. It provides three distinct login options, allowing users to access the portal based on their roles: Mentor, Mentee, or Admin.
-   **Main Components:**
    1. **Login Options:**
       - Three buttons or links labeled as "Mentor," "Mentee," and "Admin," each representing the different user roles.
       - Clicking on one of these options will direct users to their respective login or sign-up interfaces.
    2. **User Authentication:**
       - Fields for entering authentication credentials.
       - Use the "Login" or "Sign In" button to submit the authentication details.
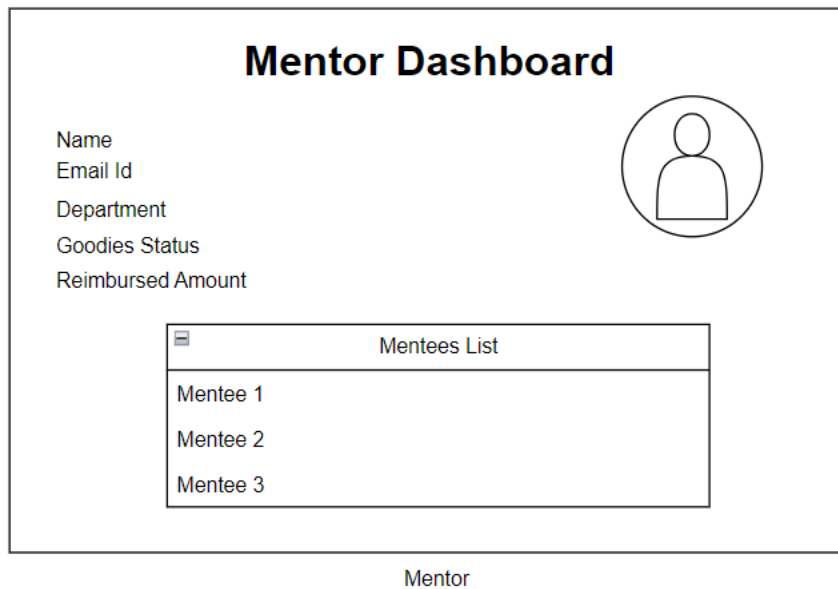
—---

For Mentor

1.
- **Screen Name:** Welcome Screen (Meteor)
- **Purpose:** This screen in Meteor serves as the initial point of interaction for new users, specifically mentors who are applying for the mentorship program. This screen provides essential information and options based on the user's current status within the application process.
- **Main Components:**
    1. **New User (Before Deadline):**
       - New users who have not yet applied to become mentors will see the mentorship application form.
       - They can fill out the application form for the mentor role.
    2. **Already Applied (Pending Approval):**
       - Users who have previously applied but are waiting for approval will be informed that their application is pending review.
    3. **After the Deadline (Deadline Over):**
       - If the mentorship application deadline has passed, new users will see the deadline over on the screen.
       - They will not be able to submit new applications.
    4. **Rejected (Not Approved):**
       - Users who have been rejected will receive a notification explaining that their mentorship application has not been approved.
       - They will be informed that they can try again in the future.
    5. **Admin Approved (RSVP):**
       - Users who have been approved as mentors will be asked to confirm their participation by filling out the Consent Form.
       - If they choose to reject any of the consent, they will be shown a screen confirming their rejection.
       - If they choose to accept all the consents, they will be directed to the Mentor Dashboard for further interactions and tasks.

2.
- **Screen Name:** Dashboard (Mentor)
- **Purpose:** Provides mentors with an overview of their mentorship activities and responsibilities.
- **Main Components:**
    1. **Mentor's profile information:**
       - Mentor's name and other relevant personal details if mentee allocation is done.
       - If not then message that mentee allocation has not been done yet.
    2. **List of assigned mentees**
    3. **Navigation links to various sections (meetings):**
       - Navigation links(in the Navigation bar) allow mentors to access various sections within the portal.

Mentor Dashboard

Name
Email Id
Department
Goodies Status
Reimbursed Amount

☐ Mentees List
Mentee 1
Mentee 2
Mentee 3

Mentor

3.
- **Screen Name:** Meeting Schedule (Mentor: from navigation bar)
- **Purpose:** This screen is designed for the Mentor to oversee and manage the meeting schedules.
- **Main Components:**
  1. **List of Previous and Upcoming Meetings:**
     - A list of previous and upcoming meetings displayed chronologically, with the most recent meeting at the top.
     - Each meeting entry is clickable, allowing the Mentor to access details and perform actions related to that meeting.
  2. **Add Meeting Button:** A button to initiate scheduling a new meeting. Clicking this button opens the "Add Meeting" screen.
  3. **Edit/Delete Meeting:** To change/delete a meeting.

4.
- **Screen Name:** Meeting Details (Mentor)
- **Purpose:** This screen provides mentors with a comprehensive view of a specific meeting, allowing them to access meeting details and mark attendance (if applicable).
- **Main Components:**
  1. **Meeting Details:** Information about the meeting, including date, time, location, and any additional notes or agenda.
  2. **Update the existing details.**
  3. **Mark Attendance (if scheduled by them):** Attendees list for marking attendance.

5.
- **Screen Name:** Schedule Meeting (Mentor)
- **Purpose:** Allows mentors to schedule a new meeting (only for the assigned mentees) and fill in the necessary information.
- **Main Components:**
  1. **Date and Time Selection**

2. **Meeting Details:** Additional details about the meeting, such as location, agenda, or topics to be discussed.
3. **Participants Selection:** Tick boxes to select participants for the meeting.
4. **Schedule Button:** A button to confirm and schedule the meeting. Clicking this button saves the meeting details and adds them to the meeting schedule.

▬---

For Mentee:

1.
- **Screen Name:** Dashboard (Mentee)
- **Purpose:** Provides mentees with an overview of their mentorship experience and basic information.
- **Main Components:**
    1. **Mentee's profile information:**
       - Display the mentee's personal information, including their name, department and email.
    2. **Assigned Mentor Details:**
       - Information about the mentor assigned to the mentee, including the mentor's name and contact information.
    3. **Navigation Bar:**
       - A navigation bar or menu providing direct access to the meeting schedule section and feedback form.
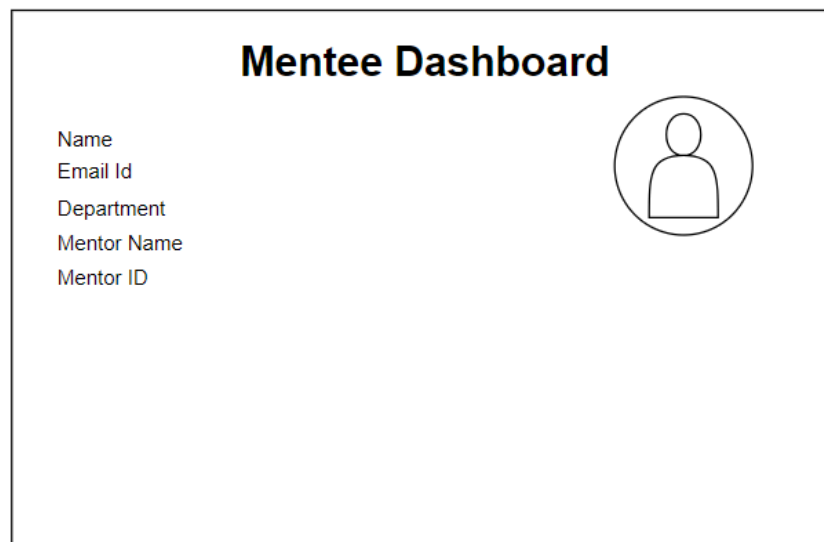
Mentee

2.
- **Screen Name:** Meeting Schedule (Mentee)
- **Purpose:** The "Meeting Schedule" screen is designed to provide mentees with a clear overview of their upcoming mentor-mentee meetings, facilitate attendance tracking, and enable them to manage their meeting-related activities.
- **Main Components:**
    1. **Previous and Upcoming Meetings:**
       - A section displaying a mentee's scheduled meetings list.

- Each meeting entry includes the meeting date, time, location, and the mentor's/Admin name.
- Meetings are typically organised chronologically, with upcoming meetings displayed first.

2. **View attendance.** Mentee can view attendance of individual meetings.
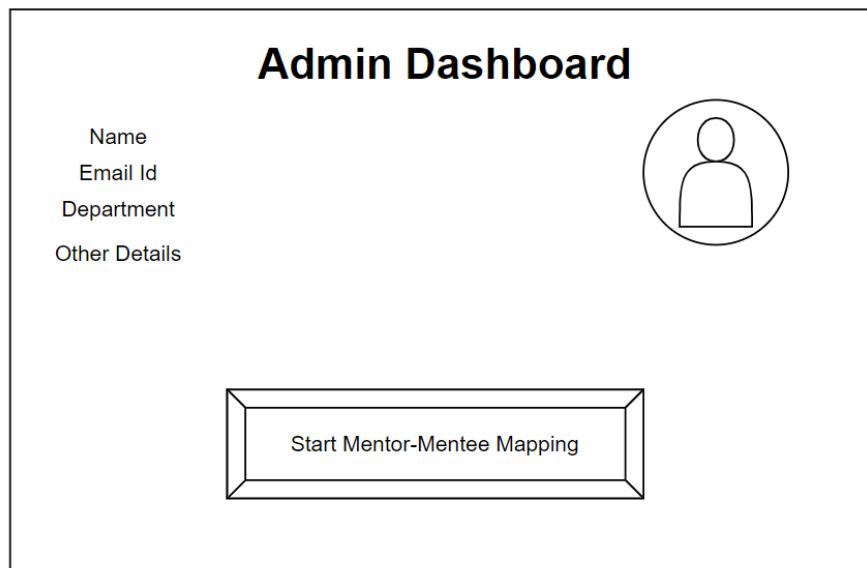
3.
- **Screen Name:** Feedback Form Page (Mentee)
- **Purpose:** Mentees can provide feedback on their mentorship experience.
- **Main Components:**
    1. Feedback form with questions and comment fields
    2. Submit button

▬--

For Admin:

1.
- **Screen Name:** Dashboard (Admin)
- **Purpose:** The Admin Dashboard serves as the central control panel for program administrators, offering a suite of tools to manage and oversee the mentorship program's operations efficiently.
- **Main Components:**
    1. **Admin's profile information:**
        - Display the administrator's essential information, including their name, email, and department.
        - Any additional details relevant to the administrator's role in the program.
    2. **Navigation Bar:**
        - A comprehensive navigation bar or menu providing quick access to essential program management sections.
        - Navigation options typically include
            a. Mentee List
            b. Mentor List
            c. Meetings
            d. Form and Responses.

Admin Dashboard

Name
Email Id
Department

Other Details

Start Mentor-Mentee Mapping

Admin

2.

- **Screen Name:** Mentee List (Admin: from navigation bar)
- **Purpose:** This screen is designed to manage the Mentees list and provide actions such as adding new mentees, removing existing mentees, changing their mentor, uploading new CSV, downloading CSV and accessing individual mentee details.
- **Main Components:**
  1. Search Bar
  2. Filter Buttons
  3. Add new mentees
  4. Remove/Delete mentees
  5. Download CSV
  6. **Upload New Mentees List:**
     - A feature that allows administrators to upload a CSV file containing a list of new mentees in bulk. This simplifies the process of adding multiple mentees at once.
  7. Mentee Profile(Clickable button - open mentee's details)

Mentors/Mentees List

Search 🔍 | Add | Remove | Upload csv |

Name 1 ⟩
Name 2 ⟩
Name 3 ⟩
Name 4 ⟩
Name 5 ⟩
Name 6 ⟩

Admin

3.

- **Screen Name:** Mentee Details Page (Admin)
- **Purpose:** Allows admin to see all the details related to mentees on a single page and monitor them.
- **Main Components:**
  1. **Details:** Mentee's Basic Information, Assigned Mentor's Details

Mentee Details

Name
Email Id
Department
Mentor Name
Mentor ID

Edit Details

Send Mail

Remove Mentee

Admin

4.

- **Screen Name:** Mentor List (Admin: from navigation bar)
- **Purpose:** This screen is designed to manage the mentors list and provide actions such as adding new mentors, deleting existing mentors, downloading CSV, and accessing individual mentor details.
- **Main Components:**
  1. Search Bar
  2. Filter Buttons
  3. Add new Mentors

4. Remove/Delete Mentors
5. Download CSV
6. Mentor Profile(Clickable button - opens mentor's details)

5.

- **Screen Name:** Mentor Details Page (Admin)
- **Purpose:** Allows admin to see all the details related to a mentor on a single page and monitor them.
- **Main Components:**
  1. **Details:** Mentor's Basic Information, Assigned Mentees details

Admin

6.

- **Screen Name:** Meetings (Admin: from navigation bar)
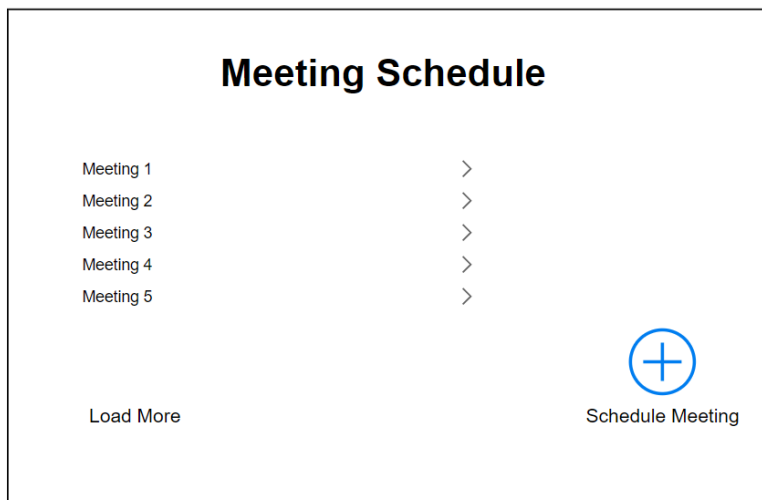- **Purpose:** This screen is designed for administrators to oversee and manage the meeting schedules.
- **Main Components:**
  1. **List of Meetings:**
     - A list of upcoming meetings displayed chronologically, with the most recent meeting at the top.
     - Each meeting entry is clickable, allowing administrators to access details and perform actions related to that specific meeting.
  2. **Add Meeting Button:** A button to initiate the process of scheduling a new meeting. Clicking this button opens the "Add Meeting" screen.
  3. **Edit/Delete Meetings**.

Meeting Schedule

Meeting 1   >
Meeting 2   >
Meeting 3   >
Meeting 4   >
Meeting 5   >

Load More   ⊕ Schedule Meeting

Admin and Mentor

7.

- **Screen Name:** Meeting Details (Admin)
- **Purpose:** This screen provides administrators with a comprehensive view of a specific meeting, allowing them to access meeting details, and mark attendance (if applicable).
- **Main Components:**
    1. **Meeting Details:** Information about the meeting, including date, time, location, and any additional notes or agenda.
    2. **Mark Attendance (if scheduled by them):** Attendees list for marking attendance.

8.

- **Screen Name:** Schedule Meeting (Admin)
- **Purpose:** Allows admin to schedule a new meeting and fill in the necessary information.
- **Main Components:**
    1. Date and Time Selection
    2. **Meeting Details:** Additional details about the meeting, such as location, agenda, or topics to be discussed.
    3. **Participants Selection:** Tick boxes to select participants for the meeting. Administrators can choose whether the mentor, mentee, or both will be part of the meeting.
    4. **Schedule Button:** A button to confirm and schedule the meeting. Clicking this button saves the meeting details and adds them to the meeting schedule.

9.

- **Screen Name:** Form Page (Admin: From navigation bar)
- **Purpose:** This page is designed for administrators to manage various forms within the mentorship program. It allows administrators to view, enable, or disable forms and access form details for review.
- **Main Components:**
    1. **List of Forms:**
    - A list of available forms, each represented with a title.
    - For each form, an on-off button is displayed in front, indicating whether the form accepts responses.

- Administrators can click on the View Responses button to access its details.

Forms and Responses

Form 1	>
Form 2	>
Form 3	>
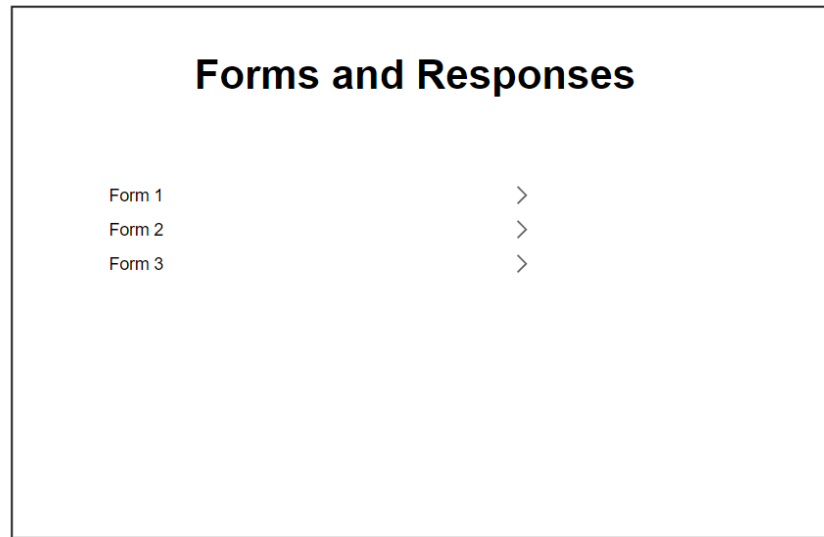
Admin

10.
- **Screen Name:** Form Response Page (Admin)
- **Purpose:** This page allows Admin to view and analyse student responses on the mentorship program's forms.
- **Main Components:**
    1. **Responses** of forms with questions as columns and unique responses as rows
    2. **Search Bar, Filter and Sort Buttons**
    3. **Checkboxes** to select form submitters
    4. **Send Consent Form Button** (only in Enrollment Form) to send consent to selected students,
    5. **Mentor-Mentee Mapping Button** (only in Consent Form) to initialise the mentor-mentee mapping process with the selected mentors.

Technology for connecting with the backend:

- **HTTP Communication:** We will connect the React frontend with the Django backend using HTTP requests. React sends HTTP requests to Django's RESTful API endpoints.

- **Request Methods:** We will utilise libraries like Axios or the built-in fetch API in React (as per specific coding requirements and needs) to send HTTP requests to Django's RESTful API endpoints.

- **Data Serialization:** These requests will facilitate data exchange in JSON format between the frontend and backend components of our Student Mentorship Portal

Back End Design Details

Backend framework:

- Django

Components/libraries:

- **Database ORM:** Django's Object-Relational Mapping for database operations.
- **Data Serialization:** Serialization of data in JSON format.
- **External Libraries:** Integration of the built-in fetch API for HTTP requests.

Details of Backend APIs

1. **API Name:** `getAllMentors`
   - **Brief Description:** Retrieves a list of all mentors from the candidate table.
   - **Inputs:** None
   - **Responses:** List of all mentor details.

2. **API Name:** `getMentorById`
   - **Brief Description:** Retrieves details of a specific mentor by their ID.
   - **Inputs:** Mentor ID
   - **Responses:**
        Successful: Mentor details for the specified ID.
        Error: An error message

3. **API Name:** `getAllMentees`
   - **Brief Description:** Retrieves a list of all mentees
   - **Inputs:** None
   - **Responses:** List of all mentee details.

4. **API Name:** `getMenteeById`
   - **Brief Description:** Retrieves details of a specific mentee by their ID.
   - **Inputs:** Mentee ID
   - **Responses:**
        Successful: Mentee details for the specified ID.
        Error: An error message

5. **API Name:** `getIdByEmail`
   - **Brief Description:** Retrieves the user ID associated with a given email based on the role (Admin, mentee, mentor)
   - **Inputs:** User email and role
   - **Responses:**
        Successful: User ID if the user email exists
        Error: An error message

.

6. **API Name:** `deleteAllMentors`
   - **Brief Description:** Deletes all mentors from the Candidate table
   - **Inputs:** None
   - **Responses:** Success/Error Message.

7. **API Name:** `deleteMentorById`
   - **Brief Description:** Deletes a specific mentor by their ID.
   - **Inputs:** Mentor ID
   - **Responses:** Success/Error Message.

8. **API Name:** `deleteAllMentees`
    - **Brief Description:** Deletes all mentees.
   - **Inputs:** None
   - **Responses:** Success/Error Message.

9. **API Name:** `deleteMenteeById`
   - **Brief Description:** Deletes a specific mentee by their ID.
   - **Inputs:** Mentee ID
   - **Responses:** Success/Error Message.

10. **API Name:** `addMentor`
    - **Brief Description:** Adds a new mentor in the Candidate table based on the Input JSON.
    - **Inputs:** A JSON containing Mentor Details (Name, Roll no, department, year, etc.)
    - **Responses:** Success/Error Message.

11. A**PI Name:** `addMentee`
    - **Brief Description:** Adds a new mentee in the Mentee table based on the Input JSON.
    - **Inputs:** A JSON containing Mentee Details (Name, Roll no, department, year, etc.)
    - **Responses:** Success/Error Message.

12. **API Name:** `addCandidate`
    - **Brief Description:** Adds a new candidate in the Candidate table based on the Input JSON.
    - **Inputs:** A JSON containing Candidate Details (Name, Roll no, department, year, etc.)
    - **Responses:** Success/Error Message.

13. **API Name:** `editAdminById`
    - **Brief Description:** Edits details of a specific admin by their ID.
    - **Inputs:** A JSON containing Updated Admin Details (Name, email, etc.)
    - **Responses:** Success/Error Message.

14. **API Name:** `editMentorById`
    - **Brief Description:** Edits details of a specific mentor by their ID.
    - **Inputs:** A JSON containing Updated Mentor Details (Name, Roll no, department, year, etc.)

- **Responses:** Success/Error Message.

15. **API Name:** `editMenteeById`
    - **Brief Description:** Edits details of a specific mentee by their ID.
    - **Inputs:** A JSON containing Updated Mentee Details (Name, Roll no, department, year, etc.)
    - **Responses:** Success/Error Message.

16. **API Name:** `uploadCSV`
    - **Brief Description:** create entries in the mentee table to add new mentees form the given CSV file.
    - **Inputs:** CSV File containing mentee details.
    - **Responses:** Success/Error Message.

17. **API Name:** `addMeeting`
    - **Brief Description:** Adds a new meeting with details such as title, schedulerId, date, time, attendees, etc.
    - **Inputs:** Meeting Details (title, schedulerId, date, time, attendee, description, mentorBranches, menteeBranches, menteeList)
    - **Responses:** Success/Error Message.

18. **API Name:** `editMeetingById`
    - **Brief Description:** Edits details of a specific meeting by its ID.
    - **Inputs:** Updated Meeting Details (title, schedulerId, date, time, attendee, description, mentorBranches, menteeBranches, menteeList)
    - **Responses:** Success/Error Message.

19. **API Name:** `deleteMeetingById`
    - **Brief Description:** Deletes a specific meeting by its ID.
    - **Inputs:** Meeting ID
    - **Responses:** Success/Error Message.

20. **API Name:** `getMeetings`
    - **Brief Description:** Retrieves a list of previous and upcoming meetings based on the user's role, department and ID.
    - **Inputs:** Role, ID
    - **Responses:**
        Success: List of previous and upcoming meetings.
        Failure: Error Message

21. **API Name:** `getAttendance`
    - **Brief Description:** Retrieves the attendees list and their attendance details for a specific meeting.
    - **Inputs:** Meeting ID
    - **Responses:**
        Success: Dictionary of attendees and their attendance.
        Failure: Error Message

22. **API Name:** `updateAttendance`
   - Brief Description: Updates the attendance of the attendees for a specific meeting. (Can both mark and unmark the attendance)
      - Inputs: Dictionary of attendees and their attendance and meetingId
      - Responses: Success/Error Message.

23. **API Name:** `createMentorMenteePair`
   - Brief Description: Create the pair of mentor-mentees from the list of selected students based on their departments. Sends emails to selected students for mentor-mentee pairing.
      - Inputs: Selected students list
      - Responses: Success/Error Message.

24. **API Name:** `submitConsentForm`
   - Brief Description: Submits the consent form with user responses. Also, update the candidate status based on the user response (consent given/rejected) and send them mail as a confirmation.
      - Inputs: JSON containing user responses
      - Responses: Success/Error Message.

25. **API Name:** `getFormResponse`
      - Brief Description: Retrieves form responses of the users based on the given form type.
      - Inputs: Form Type
      - Responses:
          Success: Form Responses of the given form type.
          Failure: Error Message

26. **API Name:** `getFormStatus`
      - Brief Description: Retrieves the on/off status of the form.
      - Inputs: None
      - Responses: List of form types and their form statuses.

27. **API Name:** `updateFormStatus`
   - Brief Description: Updates the status of a specific form and sends an email to the students regarding the same.
      - Inputs: (formId, formStatus)
      - Responses: Success/Error Message.

28. **API Name:** `sendConsentForm`
   - Brief Description: Sends consent forms to selected students via email, asking them to fillout the consent form.
      - Inputs: A list of students and emails, subject, and body
      - Responses: Success/Error Message.

29. **API Name:** `getExcellenceAward`
   - Brief Description: Calculates scores of the mentors based on mentee feedback, number of meetings etc and returns the list of mentors, their scores and the status (selected for Excellence award or not)

- Inputs: None
- Responses: List of Mentors, Their Scores and their status

30. **API Name:** `updateExcellenceAward`
    - Brief Description: Update the Exellence award status of the given mentors
    - Inputs: List of selected students
    - Responses: Success/Error Message.

31. **API Name:** `getMailSubjectAndBody`
    - Brief Description: Retrieves the subject and body of the mail based on the given type ('consent', 'pairing' etc)
    - Inputs: Type
    - Responses: Subject and body of the mail.

DB Design

- **Database System:** PostgreSQL
- **Brief justification:** We believe that PostgreSQL is a suitable choice for our SMP project's database system due to its open-source nature, Security, reliability, data integrity, and strong community support. It will provide us with a solid foundation for storing and managing the data required for our mentorship program.

Database Schemas. There are a few schemas that are to be created in the database. The main ones along with their fields are:

1. **Candidate:** id (PK), email, name, department, year, status, contact, size, score, remarks, imgSrc
2. **Mentee:** id (PK), email, name, department, contact, imgSrc, mentorId
3. **Admin:** id (PK), email, name, department, phone, address, imgSrc
4. **Meetings:** meetingId (PK), schedulerId, title, date, time, attendee, mentorBranches, menteeBranches, menteeList, description
5. **Attendance:** id (PK), attendeeId, meetingId
6. **FormResponses:** SubmissionId (PK), submitterId, FormType, responses
7. **FormStatus:** formId (PK), formStatus
8. **ExcellenceAward:** id (PK), candidateId