Software Requirement Specification for Bus Registration Portal

Name	GOKUL S
Roll no	7376221CS152
Seat no	331
Project ID	25
Problem Statement	GRIEVANCE PORTAL FOR ANONYMOUS AND PERSONAL GRIEVANCES
Stack	MERN STACK

Technical Components:

Components	Tech Stack
Backend	NODE.JS WITH EXPRESS.JS
Frontend	REACT
Database	MONGO DB
API	OPEN API

1. Landing Page

- **Start:**
    - Users arrive at the landing page, which is a React component (LandingPage).
    - The page includes options for "Admin Login" and "Teacher Login/Registration" as buttons or links.
- **Decision Point:**
    - Users decide whether they are an Admin or Teacher.
    - Depending on the choice, the user is routed to either the Admin Login or Teacher Login/Registration page using React Router.

2. Admin Flow

- **Admin Login:**
    - **Frontend:**
        - AdminLogin component handles input fields for username and password.
        - On form submission, a POST request is made to the backend API using axios or fetch.
    - **Backend API:**
        - **Endpoint**: POST /api/auth/admin-login
        - **Process**:
            - Validate the credentials against stored data in MongoDB.
            - If valid, generate a JWT and send it back to the frontend.
            - If invalid, return an error message.
    - **Frontend Response:**
        - On success: Store JWT in localStorage or sessionStorage.
        - Redirect to the AdminDashboard component.
        - On failure: Display an error message in the AdminLogin component.

- **Admin Dashboard:**
    - **Frontend:**
        - AdminDashboard component is protected by a higher-order component (HOC) or custom hook (useAuth) that checks if a valid JWT exists.
        - Dashboard displays key metrics like the number of grievances, categories, and user details.
    - **Backend API:**
        - **Endpoint**: GET /api/admin/dashboard
        - **Process**:
            - Fetch relevant admin data from the database.
            - Return the data to the frontend.
    - **Frontend Response:**
        - Data is fetched on component mount (useEffect).
        - Data is displayed using React state management (e.g., useState).
- **Manage Grievances:**
    - **Frontend:**
        - Admin can see a list of all grievances using the ManageGrievances component.
        - List is rendered using a table or list, with options to view, categorize, or reject each grievance.
    - **Backend API:**
        - **Endpoint**: GET /api/admin/grievances
        - **Process**:
            - Query the database for all grievances.
            - Return the list to the frontend.

- **Frontend Response:**
    - Data is fetched and stored in a state variable, then rendered in the ManageGrievances component.

- **View All Grievances:**
  - **Frontend:**
    - The ViewGrievances component displays detailed information about each grievance.
    - Admin can click on a grievance to view details.
  - **Backend API:**
    - **Endpoint**: GET /api/admin/grievances/:id
    - **Process**:
      - Fetch detailed data for a specific grievance from the database.
      - Return the data to the frontend.
  - **Frontend Response:**
    - Data is displayed on a modal or a new page, depending on the UI flow.
- **Categorize Grievance:**
  - **Frontend:**
    - In the CategorizeGrievance component, the admin can assign a category to each grievance.
    - This is typically done through a dropdown or radio buttons.
  - **Backend API:**
    - **Endpoint**: PUT /api/admin/grievances/:id/categorize
    - **Process**:
      - Update the grievance record in the database with the selected category.
      - Return a success message or the updated grievance.

   - **Frontend Response:**
     - The frontend updates the UI to reflect the new category, possibly with a confirmation message.

- **Query Database:**
   - **Frontend:**
     - Queries are made through various components like ViewGrievances, GenerateReports, etc**.**
   - **Backend API:**
     - **Endpoint**: Varies based on the query (e.g., GET /api/admin/reports)
     - **Process**:
       - Specific queries are processed based on the API endpoints and the type of data requested.
       - Complex queries might involve filtering, sorting, and aggregations.
   - **Frontend Response:**
     - Data is rendered in tables, charts, or other UI components based on the query results.

- **Update Status:**
   - **Frontend:**
     - In the UpdateStatus component, the admin can change the status of a grievance (e.g., "Pending", "In Progress", "Resolved").
     - This is often done via a dropdown or buttons.
   - **Backend API:**
     - **Endpoint**: PUT /api/admin/grievances/:id/status
     - **Process**:
       - The status is updated in the database.
       - Return the updated grievance data.

    - **Frontend Response:**
        - The UI updates to show the new status immediately or after confirmation.

- **Generate Reports:**
    - **Frontend:**
        - The GenerateReports component allows the admin to create and download reports based on grievance data.
        - Reports can be displayed on the screen or exported as a file (e.g., CSV, PDF).
    - **Backend API:**
        - **Endpoint**: GET /api/admin/reports
        - **Process**:
            - Backend queries the database and formats the data as requested (e.g., summary, detailed, by category).
            - Returns the report data.
    - **Frontend Response:**
        - Reports are rendered on the frontend and can be downloaded if needed.

- **Manage Users:**
    - **Frontend:**
        - ManageUsers component displays a list of teachers with options to edit or delete accounts.
    - **Backend API:**
        - **Endpoint**: GET /api/admin/users
        - **Process**:
            - Fetch all teacher user details from the database.
            - Return the data to the frontend.
    - **Frontend Response:**
        - User data is displayed, with options to perform actions (e.g., edit, delete) using

additional API calls like PUT or DELETE.

- **Admin Notifications:**
  - **Frontend:**
    - AdminNotifications component displays alerts or notifications for new grievances or updates.
    - Notifications can be in the form of a list, badge, or toast messages**.**
  - **Backend API:**
    - **Endpoint**: GET /api/admin/notifications
    - **Process**:
      - Fetch notifications from the database.
      - Return them to the frontend.
  - **Frontend Response:**
    - Notifications are displayed in real-time or on page load.
- **Logout:**
  - **Frontend:**
    - The Logout button clears the JWT from localStorage or sessionStorage.
    - Redirects to the LandingPage.
  - **Backend API:**
    - No API call is needed for client-side JWT handling.
    - If sessions are managed server-side, an endpoint like POST /api/auth/logout might be needed to invalidate the session.

3. Teacher Flow

- **Teacher Login/Registration:**
  - **Frontend:**
    - TeacherLogin and TeacherRegister components handle input for login or registration.
    - On submission, a POST request is made to the backend.

- **Backend API:**
  - **Endpoint**: POST /api/auth/teacher-login and POST /api/auth/teacher-register
  - **Process**:
    - Validate credentials for login.
    - For registration, save new teacher data in the database.
    - Return a JWT on successful login or registration.
- **Frontend Response:**
  - On success: Store JWT and redirect to the TeacherDashboard.
  - On failure: Display an error message.

- **Teacher Dashboard:**
  - **Frontend:**
    - TeacherDashboard component is protected by authentication checks.
    - Displays options like "Submit Grievance", "View Grievances", "Check Status", etc.
  - **Backend API:**
    - **Endpoint**: GET /api/teacher/dashboard
    - **Process**:
      - Fetch data specific to the teacher, such as recent grievances, status updates, etc.
      - Return data to the frontend.
  - **Frontend Response:**
    - Data is fetched on component mount and rendered accordingly.
- **Submit Grievance:**
  - **Frontend:**
    - The SubmitGrievance component provides a form for teachers to fill out.
    - On submission, the data is sent to the backend.

  - **Backend API:**
    - **Endpoint**: POST /api/grievances
    - **Process**:
      - Save the grievance in the database with the teacher's ID.
      - Return a success message or the created grievance.
  - **Frontend Response:**
    - The grievance is added to the teacher's list, and a success message is displayed.
- **View Grievances:**
  - **Frontend:**
    - The ViewGrievances component displays a list of all grievances submitted by the teacher.
  - **Backend API:**
    - **Endpoint**: GET /api/teacher/grievance.
    - **Process**:
      - Fetch all grievances associated with the teacher's ID.
      - Return the data to the frontend.
  - **Frontend Response:**
    - The list of grievances is displayed, with each item clickable for more details.
- **Grievance Status:**
  - **Frontend:**
    - The GrievanceStatus component shows the current status of each grievance (e.g., "Pending", "In Progress", "Resolved").
  - **Backend API:**
    - **Endpoint**: GET /api/teacher/grievances/:id
    - **Process**:
      - Fetch the status and details of a specific grievance.
      - Return the data to the frontend.

- **Frontend Response:**
    - Status is displayed in a user-friendly format (e.g., color-coded badges, progress bars).
- **Notifications:**
  - **Frontend:**
    - The Notifications component alerts the teacher to updates on their grievances.
  - **Backend API:**
    - **Endpoint**: GET /api/teacher/notifications
    - **Process**:
      - Fetch notifications related to the teacher's grievances.
      - Return them to the frontend.
  - **Frontend Response:**
    - Notifications are displayed in real-time or on page load.
- **Logout:**
  - **Frontend:**
    - The Logout button clears the JWT and redirects to the LandingPage.
  - **Backend API:**
    - If server-side session management is used, an API endpoint like POST /api/auth/logout might be implemented.

4. Database Interaction

- **Central Database (MongoDB):**
  - **Grievances Collection**: Stores all grievances with fields such as title, description, status, category, teacher_id, and timestamps.
  - **Users Collection**: Stores user data, differentiating between Admins and Teachers.
  - **Notifications Collection**: Stores notifications for both teachers and admins.

- **Data Flow:**
    - **CRUD Operations**: Admin and Teacher actions result in Create, Read, Update, and Delete (CRUD) operations on the database.
    - **Real-Time Updates**: Using WebSockets (e.g., Socket.IO) for real-time notifications and updates.
    - **Indexing and Optimization**: Implementing indexing on frequently queried fields like status, teacher_id, and created_at for faster database operations.

5. End

- **Session Termination:**
    - After logout, the session is terminated, JWT is cleared, and the user is redirected to the landing page.
- **Post-Logout:**
    - Ensure no residual data is accessible by clearing React state, cookies, and storage.

Tools & Technologies

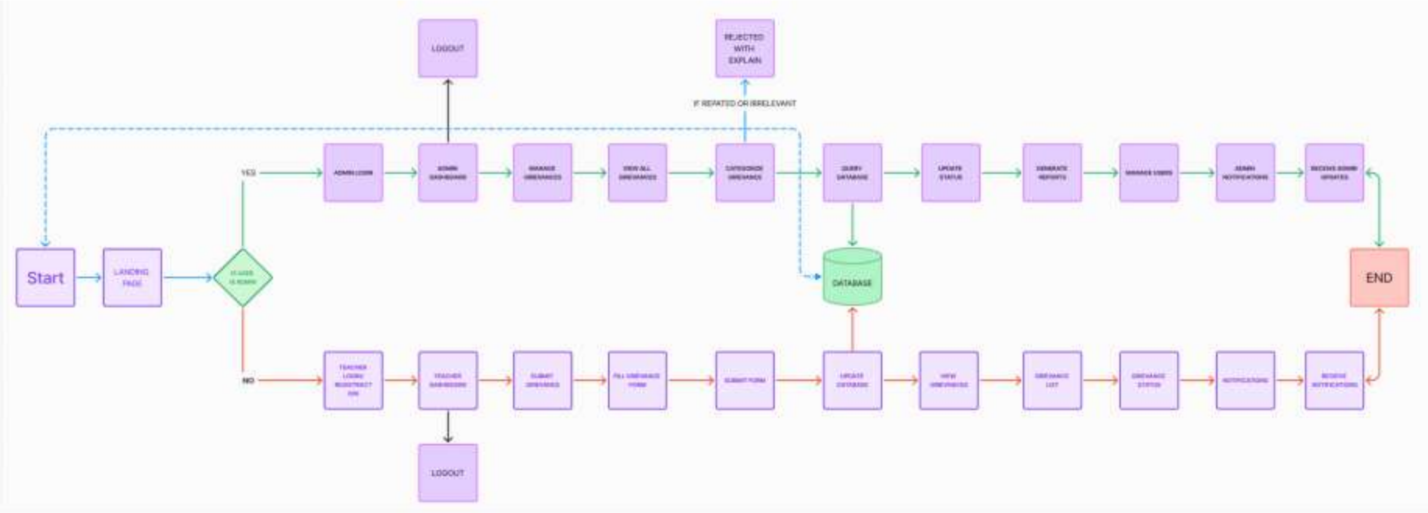
- **Frontend**:
    - **React.js**: Primary library for building the user interface.
    - **React Router**: For handling routing between different components/pages.
    - **Axios/Fetch**: For making API requests.
    - **State Management**: Using React's built-in useState and useEffect hooks or Context API for managing global state.
    - **Styling**:
        - **CSS/SCSS**: Custom styling for components.
        - **Bootstrap or TailwindCSS**: For rapid UI development with predefined classes.
- **Backend**:
    - **Node.js with Express.js**: For creating RESTful API endpoints.
    - **MongoDB**: For storing all application data.

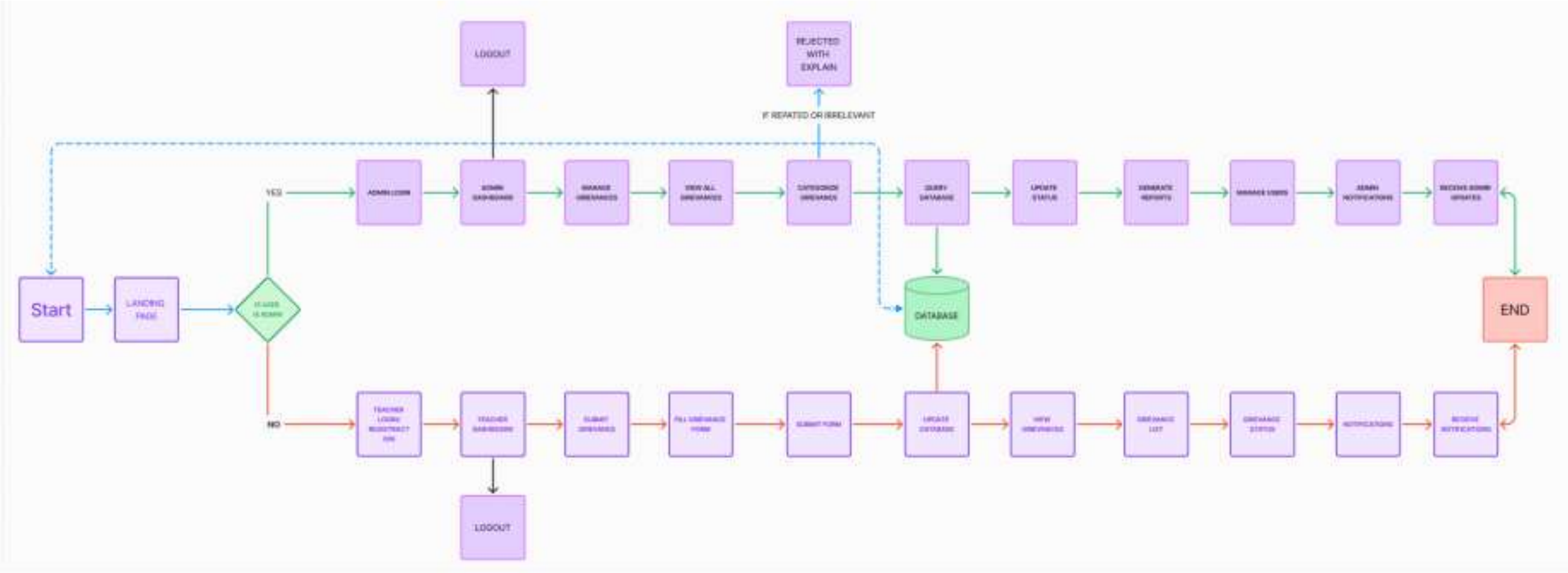
- **Authentication**:
  - **JWT**: JSON Web Token for securing API endpoints.
- **Security**:
  - **Validation**: Use libraries like express-validator for backend input validation.
  - **Sanitization**: Protect against XSS and other injection attacks by sanitizing inputs.

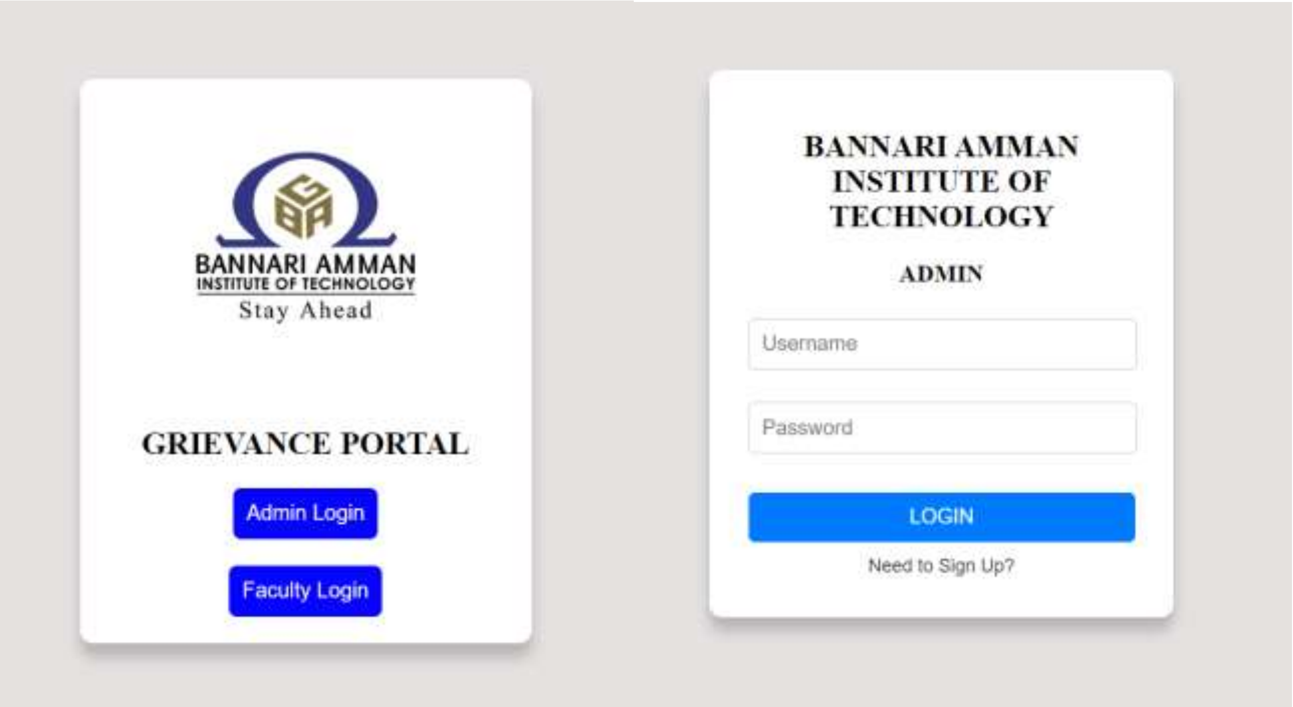
Development Process

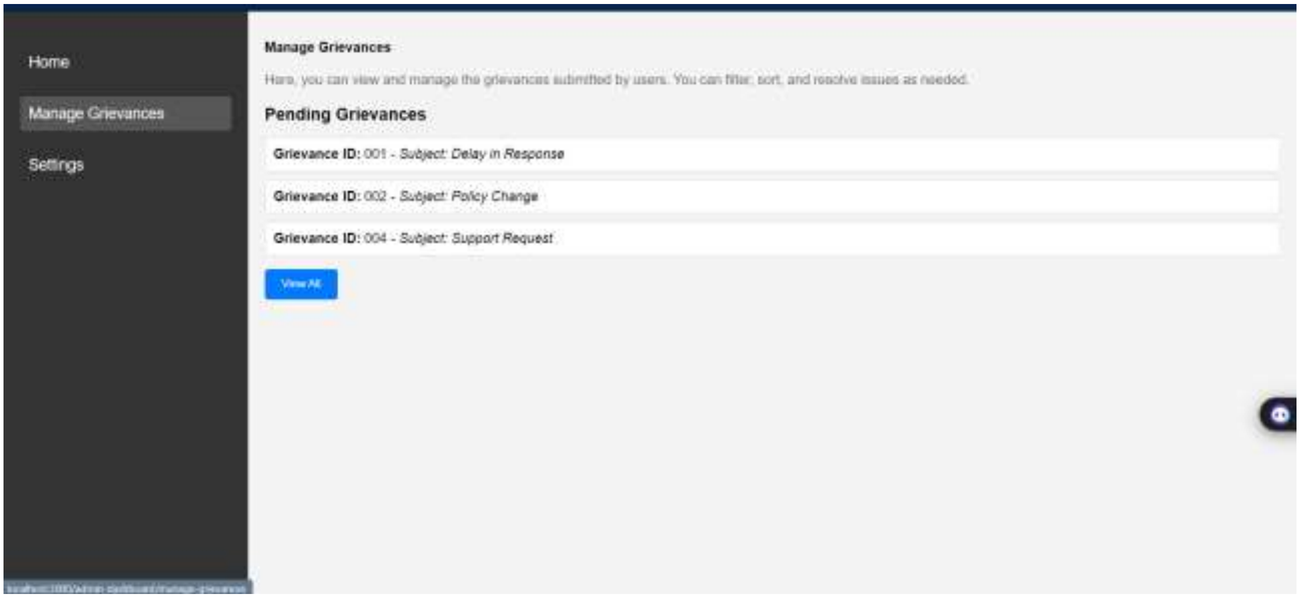
- **Component Breakdown**: Break the project into small, reusable React components (e.g., AdminLogin, TeacherDashboard, GrievanceForm).
- **API Integration**: Gradually integrate API calls with React components, ensuring data flows correctly from the backend to the frontend.
- **Testing**: Use tools like Jest and React Testing Library for unit testing components and integration tests.
- **Deployment**:
  - **Frontend**: Deploy on platforms like Netlify or Vercel.
  - **Backend**: Deploy on cloud services like Heroku, AWS, or DigitalOcean.
  - **Database**: Host on MongoDB Atlas for scalability and easy management.

This enhanced flow should provide a more detailed roadmap for building and implementing the grievance portal with React, covering all aspects from frontend to backend, including database interactions and API integrations.


LOGOUT
REJECTED WITH EXPLAIN
IF REPATED OR IRRELEVANT
YES
ADMIN LOGIN
ADMIN DASHBOARD
MANAGE GRIEVANCES
VIEW ALL GRIEVANCES
CATEGORIZE GRIEVANCES
QUERY DATABASE
UPDATE STATUS
GENERATE REPORTS
MANAGE USERS
ADMIN NOTIFICATIONS
RECEIVE ADMIN UPDATES
Start
LANDING PAGE
IS USER IS ADMIN
DATABASE
END
NO
TEACHER LOGIN/ REGISTRATION
TEACHER DASHBOARD
SUBMIT GRIEVANCE
FILL GRIEVANCE FORM
SUBMIT FORM
UPDATE DATABASE
VIEW GRIEVANCES
GRIEVANCE LIST
GRIEVANCE STATUS
NOTIFICATIONS
RECEIVE NOTIFICATIONS
LOGOUT


Start
LANDING PAGE
YES
NO
ADMIN LOGIN
ADMIN DASHBOARD
LOGOUT
MANAGE GRIEVANCES
VIEW ALL GRIEVANCES
CATEGORIZE GRIEVANCES
IF REPATED OR IRRELEVANT
REJECTED WITH EXPLAIN
QUERY DATABASE
DATABASE
UPDATE STATUS
GENERATE REPORTS
MANAGE USERS
ADMIN NOTIFICATIONS
RECEIVE ADMIN UPDATES
END
TEACHER DASHBOARD
LOGOUT
SUBMIT GRIEVANCE
FILL GRIEVANCE FORM
SUBMIT FORM
UPDATE DATABASE
VIEW GRIEVANCES
GRIEVANCE LIST
GRIEVANCE STATUS
NOTIFICATIONS
RECEIVE NOTIFICATIONS





Manage Grievances

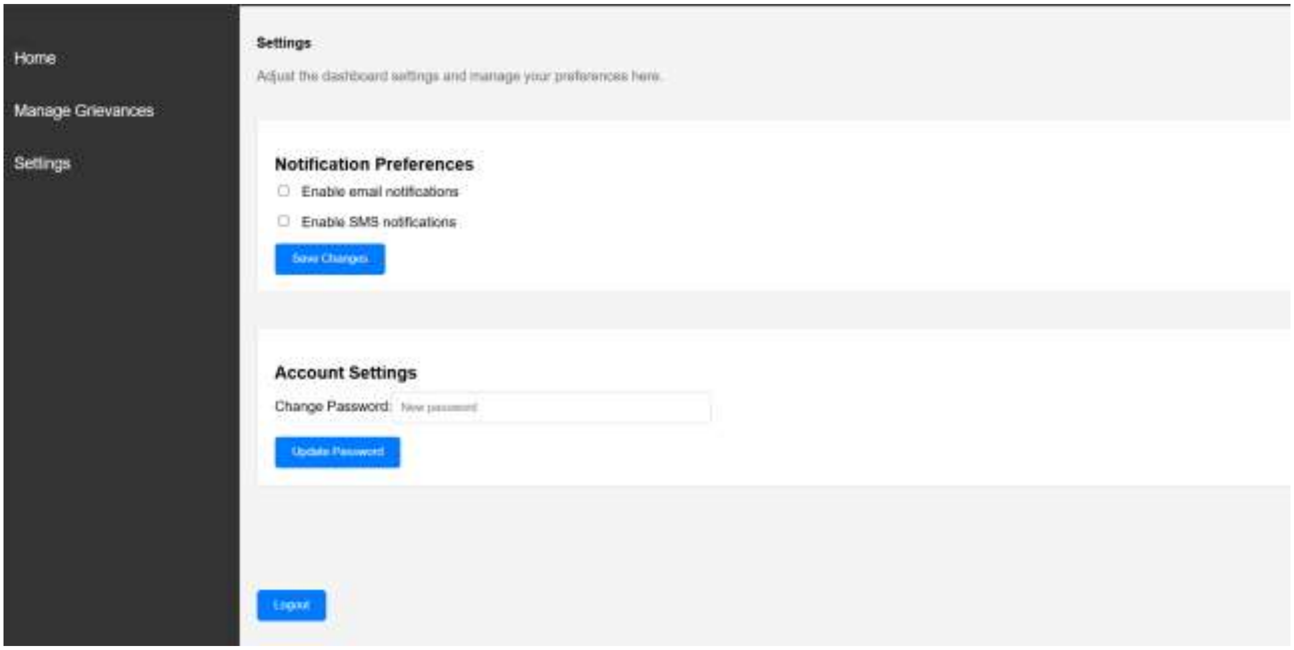
Here, you can view and manage the grievances submitted by users. You can filter, sort, and resolve issues as needed.

Pending Grievances

Grievance ID: 001 - *Subject: Delay in Response*

Grievance ID: 002 - *Subject: Policy Change*

Grievance ID: 004 - *Subject: Support Request*



Settings

Adjust the dashboard settings and manage your preferences here.

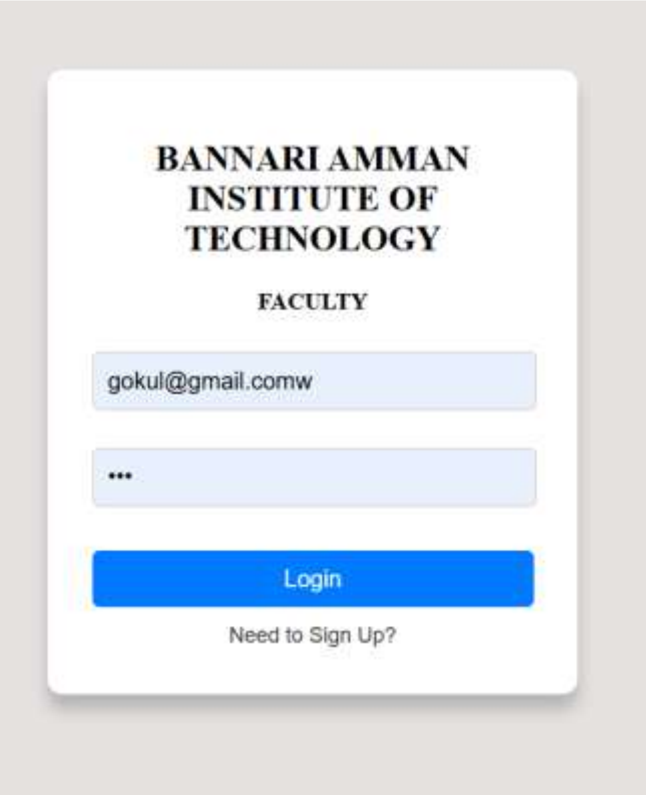
Notification Preferences

□ Enable email notifications

□ Enable SMS notifications

Account Settings

Change Password: New password

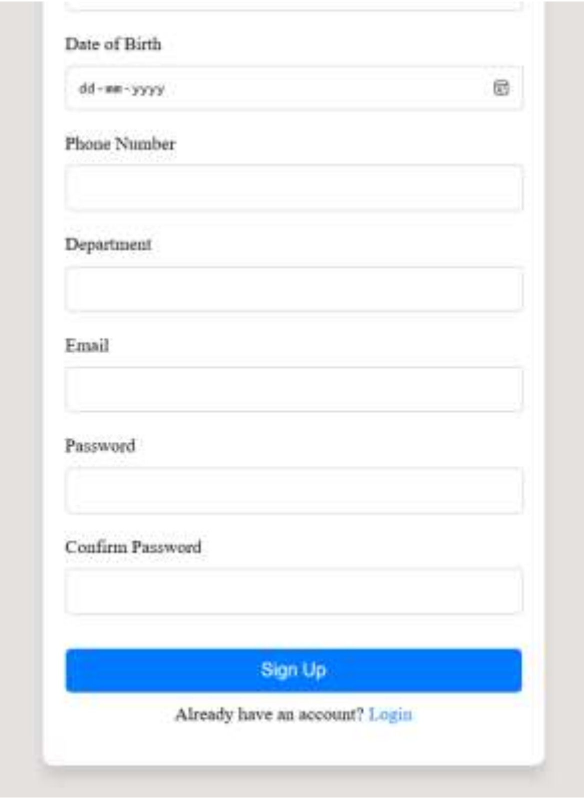


BANNARI AMMAN INSTITUTE OF TECHNOLOGY

FACULTY

gokul@gmail.comw

•••



Date of Birth

dd-mm-yyyy

Phone Number

Department

Email

Password

Confirm Password