Clearance Project Scope Statement

PROJECT TITLE: Student Clearance System

DATE PREPARED: 20th May 2025

PREPARED BY: Linnet Wahome

Project Overview

A **student clearance system** aimed at reducing processing time and streamlining clearance workflows. **Students** *place their requests, upload relevant documents, and submit them* to the respective **staff** *authorized to handle the approval or rejection* of that particular level of clearance, namely, *project, lab, or library clearance*. The **admin** user *oversees the activities of both parties* on the system.

Project Objectives and Goals

1. Reduce Clearance processing time
2. Streamline clearance workflows
3. Make finding and rectifying issues from both ends, student or staff, easier
4. Enhance transparency in the clearance process through admin monitoring

Core Features

There are three users that can use the system, the core features are based on how each can interact with the system. Namely:

1. The student
2. The staff
3. The admin

Student

Activate account/Sign In

- Activation step: Students use their *name, university email* and *enrollment number* to activate their account (not the regular sign up).
- After activation, set & confirm password and details stored in User table in the database, for sign in, university email and password
- Each student has access only to their account and requests

Communicate with Staff

- Staff may leave comments if additional info or corrections are needed.
- Students can reply to these comments within each request's detail view.
- No separate messaging, communication happens only in the context of requests.

Upload documents

- Upload documents in supported file formats: PDF, DOCX; depending on clearance type: *Project, Lab or Library*
- System validates file type and size for security

Check Status

- Requests display real-time status: *In-progress, Approved, Rejected, Needs Changes*
- Possible addition: Students can track progress without direct staff contact.

Submit Request

- After attaching documents, click submit.
- Request is sent to the assigned staff for review.
- Students cannot edit submitted requests unless staff request changes.

View History

- Dashboard or history page lists all requests (past and current).
- Filter by type or status for better organization.

Staff

Activate account/Sign In

- Staff accounts are created by the admin (HoD).
- Activation required: staff enters name, university email and staff ID to activate, once validated, set and confirm password next.
- After that, sign in with universty email and password.

Response

- Review the request and supporting documents. Options to:
  - i.) Approve (mark request as complete)
  - ii.) Reject (with reason)
  - iii.) Request resubmission (with comments).
- All actions are logged for accountability.

List View of Requests

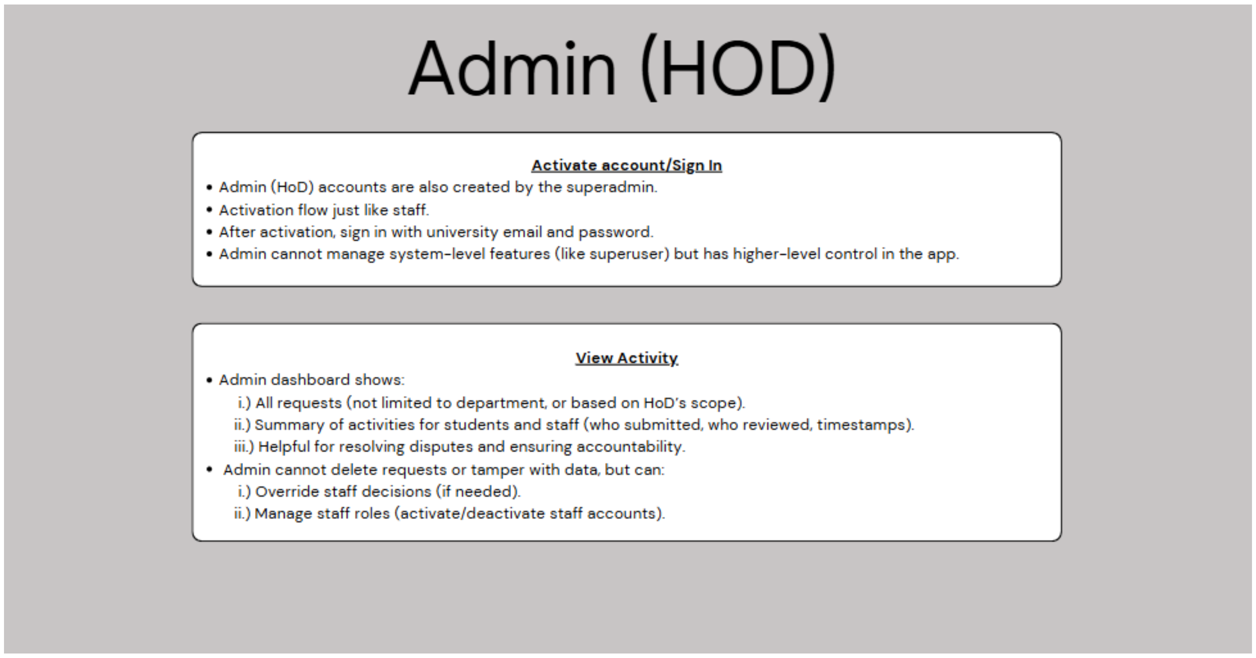
- Staff dashboard shows assigned requests in a list (summary of key details: student name, course, department, type, date, view request button).
- Easy to scan and prioritize

View History

- Staff can view past and present requests they handled.
- Filter by status or date for efficient management.

Request Access

- Click on a request to see:
  - i.) Uploaded documents
  - ii.) Any existing comments



Admin (HOD)

Activate account/Sign In

- Admin (HoD) accounts are also created by the superadmin.
- Activation flow just like staff.
- After activation, sign in with university email and password.
- Admin cannot manage system-level features (like superuser) but has higher-level control in the app.

View Activity

- Admin dashboard shows:
  - i.) All requests (not limited to department, or based on HoD's scope).
  - ii.) Summary of activities for students and staff (who submitted, who reviewed, timestamps).
  - iii.) Helpful for resolving disputes and ensuring accountability.
- Admin cannot delete requests or tamper with data, but can:
  - i.) Override staff decisions (if needed).
  - ii.) Manage staff roles (activate/deactivate staff accounts).

Bonus Features

AI-powered chatbot - Integrate AI with the website using a free AI API for personalized assistance, e.g Zapier or IBM Watson Assistant

Tech Stack

1. Front-end – React JS, Bootstrap/traditional CSS
2. Back-end – Django
3. Database – MySQL

Project Inclusions and Exclusions

❖ Inclusion

- Development and execution of a comprehensive clearance platform.
- Optimization for the current user operations, like enhancing authentication.
- Database management.
- Testing the application against dummy data at each development stage.
- Integration with a free AI API.

❖ Exclusion

- Real-time notifications via SMS or email.

- Mobile app development (web-only version planned).
- Handling financial clearance or student fee reconciliation.

Project Timeline

Week 1 (26th - 30th May): Project Setup & Planning

- Finalize requirements and user roles.
- Define core features and system boundaries.
- Set up project repo on GitHub.
- Initialize Django (backend) and React (frontend) projects.
- Create project folder structure.
- Document project scope and plan in README.md.
- **Push to GitHub:** Initial setup.

Week 2 (2nd - 6th June): Authentication & Account Activation

- Implement user roles (Student, Staff, Admin) in Django.
- Set up custom User model and ValidStudent / ValidStaff tables.
- Develop account activation flow
- Match email + enrollment/staff ID
- Set password securely
- Design activation and sign-in forms in React.
- Connect to backend APIs (activation, login, logout).
- **Push to GitHub:** Working user auth + activation.

Week 3 (9th - 13th June): Student Dashboard & Request Workflow

- Backend:
  - Create models for clearance requests.
  - Create APIs to handle student submissions.
- Frontend:
  - Build student dashboard.
  - Build upload form (project, lab, library).

    - Display real-time status.
- **Push to GitHub:** Student dashboard module.

Week 4 (16th - 20th June): Staff Dashboard & Request Review

- Backend:
    - Implement staff views & permissions.
    - Add logic to update request status + comments.
- Frontend:
    - Staff dashboard to list requests.
    - View individual requests & student documents.
    - Add comments or resubmission requests.
- **Push to GitHub:** Staff-side functionality.

Week 5 (23rd - 27th June): Admin (HoD) Monitoring Tools

- Admin dashboard to:
    - View all requests and activity logs.
    - Override staff decisions.
    - View staff/student request summaries.
    - Implement permission controls (admin vs. staff vs. student).
- **Push to GitHub:** Admin (HoD) dashboard core.

Week 6 (30th June - 4th July): Internal Testing - Student & Staff Flows

- Test:
    - Student request flow
    - Staff review and feedback flow
    - Use dummy data and simulate real-world use.
    - Frontend: Validate forms, UI states, errors.
    - Backend: Validate logic, data handling, and edge cases.
- **Push to GitHub:** Polished Student-Staff workflow.

Week 7 (7th - 11th July): AI Chatbot (Bonus Feature)

- Research and integrate a free AI assistant (e.g. OpenAI GPT API or IBM Watson).
- Build lightweight chatbot UI in frontend.
- Scope-limited use:
- FAQ responses
- Navigation help
- **Push to GitHub:** Optional AI chatbot branch.

Week 8 (14th - 18th July): Admin Testing & Permissions

- Test admin workflows:
    - Activity tracking
    - Overrides
    - Permission boundaries
    - Simulate misuse cases (e.g., staff accessing unauthorized data).
    - Finalize RBAC (Role-Based Access Control) enforcement.
- **Push to GitHub:** Stable Admin logic.

Week 9 (21st - 25th July): Final Bug Fixes & UI Polish

- Refactor repetitive code.
- Fix any minor bugs or inconsistencies.
- Improve UI/UX responsiveness and accessibility.
- Add loading indicators, error boundaries, tooltips where needed.
- **Push to GitHub:** Pre-final stable version.

Week 10 (28th July – 1st August): Documentation

- Write final documentation:
    - Setup guide (dev + deployment)
    - Usage instructions for each user role
    - Postman collection for APIs

    - Add screenshots (or a live demo link) in README.
- **Final Push to GitHub:** Complete documentation.

Project Assumptions and Constraints

Assumptions

- Users (students and staff) have basic digital literacy and internet access.
- Dummy data will be used for development and testing purposes.
- Only clearance for **project**, **lab**, and **library** is in scope.
- The admin user will have full visibility and control over all clearance activities.
- All users will be using modern browsers (Chrome, Firefox, Edge).

Constraints

- Limited development timeline (as defined in the milestone/timeline section).
- Hosting and deployment may be restricted to free-tier services (e.g., Heroku, Netlify).
- No use of paid APIs, services, or software licenses.
- AI chatbot feature is optional and only to be implemented if time allows.