Project Allotment Portal

An Open Ended Lab Project

Semester 6
Bachelor of Technology

by

Abhirami R Iyer
112201001
Department of Computer Science Engineering

and

Samyuktha K
102201003
Department of Civil Engineering

Under the Supervision of

Dr. Garima Shakya

Department of Data Science Engineering

IIT PALAKKAD

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

Kanjikode, Palakkad, Kerala - 678623, India

January, 2025 - May, 2025

Contents

Abstract

Academic project allocation is often managed manually or through inefficient processes, leading to confusion, delays, and dissatisfaction among students and faculty. This Project Allotment Portal addresses the need for a transparent, structured, and automated system to manage project applications and assignments in academic institutions. The portal allows students to apply for projects and rank their preferences, while faculty can post projects, evaluate and rank applicants, and define selection criteria based on CGPA, department, year, and prerequisites. An admin module oversees timelines and initiates phases of the allocation process. Several allocation algorithms have been implemented for fair project allocation like the Boston Mechanism and Gale-Shapley. Built using Angular, Flask/Node.js, and PostgreSQL, the system ensures a seamless, equitable, and scalable approach to academic project distribution.

Chapter 1

Introduction

1.1 Problem Statement

At IIT Palakkad, Open Ended Lab Projects (OELPs) play a crucial role in enabling students to gain research exposure and hands-on learning under faculty mentorship. However, the current process for allotting OELPs is unstructured and heavily dependent on informal communication. There is no centralized system in place — students must email faculty members individually to request project opportunities. This method is highly inefficient and inequitable.

For students, especially those who do not already know faculty members personally, this becomes a frustrating and uncertain process. Emails often go unanswered, especially when faculty receive large volumes of requests or have already committed to students they are familiar with. As a result, many capable students are denied equal access to opportunities simply because they lack prior visibility or connections, not because of merit.

From the faculty's perspective, manually tracking student emails, verifying eligibility, and allocating projects becomes tedious and unmanageable, particularly when receiving repetitive or poorly formatted messages. The absence of any filtering mechanism or standard application format further complicates the process.

This leads to unnecessary delays, lack of transparency, and an overall inefficient experience for both students and faculty. There is a strong need for an automated, centralized web portal tailored for OELP allotment at IIT Palakkad. Such a system should facilitate structured student applications, allow faculty to define selection criteria, and ensure a fair, merit-based, and easily manageable allocation process.

1.2 Objectives

The primary objective of this project is to design and develop a centralized **Project Allotment Portal** for the allocation of **Open Ended Lab Projects (OELPs)** at **IIT Palakkad**, addressing the current lack of structure and fairness in the selection process. The specific goals of the system are:

1. To **eliminate the need for informal email-based communication** between students and faculty by providing a structured platform for OELP applications.
2. To ensure **equal access to project opportunities** for all students, regardless of personal familiarity with faculty members.
3. To enable faculty to **post project details, define eligibility criteria, view applications, and manage project allotment** efficiently.
4. To allow students to **view available projects, apply with a single, unified form, and rank their preferences** based on interest.

5. To implement fair and transparent **allotment algorithms** (e.g., Boston Mechanism or Gale-Shapley) based on factors such as CGPA, department, year, and prerequisite completion.

6. To provide an **admin interface** for managing timelines (phases), initiating allotment rounds, and monitoring overall activity across the platform.

7. To build a scalable, user-friendly, and responsive **web application** using Angular (frontend), Node.js (backend), and PostgreSQL (database).

8. To reduce workload and confusion for both faculty and students by **automating repetitive tasks and maintaining centralized records of applications and allotments.**

Chapter 2

Overview

2.1 Overall Structure

The platform is structured into three dedicated interfaces: Student Side, Faculty Side, and Admin Side, each tailored to the specific roles and functionalities of the respective users. All three interfaces are accessible from a central landing page, which serves as the entry point to the portal. From this main screen, users can navigate to their designated section and log in through a role-based authentication system.

Additionally, the landing page provides access to an Algorithms Visualizer — a component designed to enhance transparency and understanding of the allocation process. This visualizer demonstrates the working of the underlying allocation algorithms (such as the Boston Mechanism or Gale-Shapley), and includes clear explanations of how the system assigns projects based on various criteria. It also guides users on how to effectively navigate and use the portal.

2.2 User Roles

2.2.1 Student

- View available OELPs and faculty details
- Apply to multiple projects with a ranked preference order
- View application status and final allotment
- Sign up if new, or log in to access their dashboard

2.2.2 Faculty

- Create and manage OELP listings
- View applicants and their profiles (CGPA, year, department, prerequisites completed)
- Set eligibility criteria and priorities (e.g., branch match, prerequisites, CGPA)
- View final allotment of students to their projects

2.2.3 Admin

- Set application and allotment timelines (phases)
- Monitor platform activity
- Initiate and manage allotment algorithms
- Edit or override allotments if necessary

2.3 Phases

2.3.1 Phase I. Project Proposal Phase

In the Proposal Phase, faculty members submit their project proposals through the portal. Each project includes a detailed title, description, expected outcomes, the number of available slots, and eligibility criteria such as required CGPA, department, year of study, and prerequisite courses. This phase ensures that all project offerings are finalized before students begin applying. Faculty can edit or delete their proposals during this phase, and the system ensures that only valid and complete entries are published for student view.

2.3.2 Phase II. Student Application Phase

Once the Proposal Phase ends, the Application Phase begins. During this period, students can browse through the list of available OELPs and apply to the projects they are eligible for. The portal automatically filters projects based on each student's academic profile, including CGPA, department, semester, and completed courses. Students are required to submit a short bio, a statement of purpose explaining why they are interested in the project, and can optionally upload supporting documents like resumes or project portfolios. Students may apply to multiple projects during this phase.

2.3.3 Phase III. Preferences Setting Phase

After applications have been submitted, both students and faculty enter the Preference Setting Phase. Here, students must rank the projects they have applied to in order of preference. Similarly, faculty members rank the students who have applied to their projects based on academic credentials and the submitted application. The preference data from both sides plays a critical role in the subsequent allocation process. This phase ensures mutual prioritization and gives both students and faculty a degree of control in the matching.

2.3.4 Phase IV. Allocation Phase

In the final Allocation Phase, the system uses a backend algorithm (e.g., Boston Mechanism or Gale-Shapley Stable Matching) to assign students to projects. The algorithm takes into account the preferences from both students and faculty, along with eligibility and slot constraints. Once the allocations are finalized, the results are published on the portal. Students and faculty can view their respective matches, and admins may intervene only in special cases where manual overrides are necessary. This phase concludes the allotment cycle.

2.4 Student-Side Screen

- **Login**

  Students, faculty members, and administrators can securely log in to the portal using their unique credentials. Each user type is registered in the system's database with associated usernames and hashed passwords. Upon successful login, the student's academic data—such as CGPA and list of completed courses—is automatically retrieved from the database. This information is used to personalize the student's experience and determine eligibility for various projects. Once authenticated, the student is redirected to their dedicated dashboard.

- **Student Dashboard**

  The student dashboard features a clean and intuitive layout with a navigation bar containing multiple tabs. These include: "All Projects," showing all projects proposed by faculty, "Available Projects" showing projects which are still available; "Eligible Projects," which filters and displays only those projects for which the student qualifies; and "Profile," where students can view their academic information and previously submitted applications. Project listings contain detailed information including the project title, description, faculty mentor, preferred departments, eligible semesters, required prerequisite courses, and number of available slots. Eligibility is determined based on criteria set by faculty, such as CGPA thresholds, department and year restrictions, and completion of specific prerequisites.

- **Project Application**

  To apply for a project, the student must fill out a structured application form. This includes a personal bio where the student introduces themselves, especially useful in cases where the faculty member is unfamiliar with the applicant. The student is also required to write a short statement explaining why they are interested in the project and what makes them a suitable candidate. Additionally, the portal allows for optional document uploads, enabling students to attach resumes, project portfolios, or any other relevant materials to strengthen their application.

- **Application Tracker**

A key feature of the student interface is the Application Tracker, which is accessible via a side panel. This component allows students to monitor all their submitted applications and view the current status of each—such as pending, shortlisted, or accepted. Students are also required to rank the projects they apply to in order of preference. This preference data is later used during the allocation process. Importantly, students can also view their position in a faculty member's preference list for a given project. This visibility helps students gauge their chances of being selected and plan accordingly.

2.5 Faculty-Side Screen

- **Login**

  Faculty members can log in securely using their institutional credentials. Their data is stored in the system's database, with each faculty member uniquely identified. Upon login, their associated projects and received student applications are retrieved and displayed on their dashboard. This ensures that faculty members can efficiently manage project postings and applicant tracking in a centralized and streamlined interface.

- **Faculty Dashboard**

  The faculty dashboard provides a user-friendly interface with dedicated sections for managing project proposals, reviewing student applications, and setting preferences. Faculty can create and manage multiple projects, each with specific details such as title, description, number of available slots, preferred departments, eligible semesters, minimum CGPA, and required prerequisite courses.

- **Application Review**

  Faculty can access a list of students who have applied for each of their posted projects. For each applicant, relevant academic data such as CGPA, department, year of study, and completed courses are displayed, along with the student's written bio and application statement. Faculty members can also view any uploaded documents, such as resumes or project portfolios. This centralized view allows for quick comparison of applicants and informed decision-making.

- **Preference Setting and Shortlisting**

  To facilitate the allocation process, faculty are required to rank the students who have applied to each project in order of preference. The interface provides easy drag-and-drop or button-based controls to reorder applicants. These preferences are then used by the backend matching algorithm to allocate students fairly. Faculty can also mark applicants as shortlisted or rejected, based on initial screening.

- **Final Allocation**

  Once the final allocation is computed (based on the selected algorithm, such as the Boston Mechanism or Gale-Shapley), faculty are shown the list of students who have been allotted their projects. They are also notified of any conflicts or oversubscriptions. The system ensures that allocations align with both student and faculty preferences, while respecting capacity constraints.

2.6 Admin Screen

The responsibility of the admin is to manage the different phases of the project allotment cycle. The admin can set and update the start and end dates for each of the four predefined phases of the process— project proposal phase, student application phase, preference setting phase, and allocation phase. The system ensures that each phase starts the day after the previous one ends, maintaining logical continuity and automating the scheduling. These dates are stored in the PostgreSQL database and are used to dynamically control user permissions and page visibility during each phase.

2.7 Common Screen

The common screen comprises of the faculty login, student login, admin screen and the help/visualizer window navigation. From the common screen, clicking on faculty login would navigate to the faculty login screen in the same window, whereas clicking on student login would take us to student login screen. clicking on admin login would take us to a window where we can change the phase timelines. Help would take us to the visualizer window.

2.7.1 Visualizer

The visualizer component consists of tabs called overview, spa-lecturer, spa-student, Gale-Shapley student proposing, Gale-Shapley faculty proposing. The overview tab displays the details of each phase. The specific algorithm tabs display the the work-flow of algorithm with each student name encoded in a specific format, so that privacy is ensured.

Chapter 3

Algorithms

3.1 Gale Shapley - student proposing

In this the famous matching algorithm the gale-shapley which is used to solve marriage problems, producing stable marriages is used. Usually gale-shapley is used in one to one mapping, but here since a many-to-one mapping is needed, we tried to modify the already existing algorithm. In student proposing, the students propose and projects accept or reject according to their preference list.

- The students start as free
- All projects are initially available(that is they have free slots)
- The iterative process :
  - The process continues as long as there is atleast one unassigned student who still has projects in their preference list, they have not yet proposed to.

- Pick any student say s_i, traverse its preference list say P_{s_i}, pick the topmost project p_j from it (the one which they have not yet proposed) and check if the project is available.
- p_j evaluates
  * If p_j has a free slot, p_j tentatively accepts s_i.
  * If p_j is tentatively filled, we walk through the already assigned students and pick the student with the worst rank among them, let it be s_{worst}. if $rank(s_i) < rank(s_{worst})$, then remove s_{worst} from the assignment and add s_i. Add s_{worst} back to the list of free students. But if $rank(s_i) > rank(s_{worst})$, then p_j rejects s_i
- This loop continues until all the students are tentatively assigned to some project or till all the students have traversed all their preferences.

This gives a **Student-optimal** matching. The algorithm ensures **Guaranteed Termination**.

3.2 Gale Shapley - lecturer proposing

- The students start as free
- All projects are initially available(that is they have free slots)
- The iterative process :
  - The process continues as long as there is atleast one project with students in their preference list not yet proposed to. i.e., there are free slots in the projects which have not yet traversed their preference lists.

- Pick a free slot say P_{i_0}, it consults it preference list (i.e., nothing but the preference list of project P_i. From the list, pick the top-most preferred student say s_i which it has not yet proposed to.
- s_i evaluates
  - If s_i is not yet assigned to any project, it tentatively accepts p_{i_0}.
  - If s_i is tentatively assigned, we compare the ranks of the projects, let the currently assigned project be p_{curr_n}. if $rank(p_{i_0}) < rank(p_{curr_n})$, then s_i accepts the slot else rejects.
- This loop continues until all the slots are tentatively assigned to some student or till all the slots have traversed all their preferences.

This gives a **Project-optimal** matching. The algorithm ensures **Guaranteed Termination**.

3.3 SPA

This algorithm is adapted from the paper Student-Project allocation. Here, they have modified the usual Gale-Shapley algorithm, to facilitate many-to-one matching. According to the paper, they have a set of students $\{s_i, s_{i+1}, \ldots\}$ and sets of lecturers $\{l_i, l_{i+1}, ...\}$ each lecturer would have a set of projects, say P_i, where P_i is the set of projects given by l_i. They have also considered capacities for each project and also a maximum capacity constraint for the faculty. In the paper, another notable point is that they have only a preference list per lecturer(this is not per project). In our case, there is no specific capacity constraint for the faculty as it depends on the number of projects he/she gives and their available slots. Also, one of the major changes on our side is that we have project-specific preferences, unlike in the paper, where there are faculty-specific preferences. Let us see how we have implemented each of the sides of the SPA algorithm.

3.3.1 SPA - student side

Similar to the gale-shapley student side, here also students propose projects.

- The students start as free
- All projects are initially available
- The proposal round :
  - Pick any free student, say s_i, and get the highest ranked project among the student's preference list which he/she has not yet proposed to and lets say its p_j.
  - p_j evaluates
    - If p_j has a free slot, it provisionally accepts s_i.
    - If p_j is full, it compares s_i to the least preferred student currently in the project. If s_i is ranked higher than the least preferred student, it accepts s_i(then the least preferred student gets added back to free list), else rejects(s_i would still be free).
    - Deletion rule : if s_i is accepted by p_j, all the students ranked below s_i in p_j's preference list are removed. This is one of the ways to prune the search space of potential matches, thereby getting a convergible solution.
  - This loop continues until all the slots are tentatively assigned to some student or till all the slots have traversed all their preferences.

This gives a **Student-optimal** matching. The algorithm ensures **Guaranteed Termination**.

3.3.2 SPA - lecturer side

Here implementation is a bit different from the paper, in the paper there is only one preference list per faculty, but in our implementation we consider a preference list per project. So we have modified our algorithm in such a way that we assume that each project is provided by dummy lecturer (whose id would be same as the project id) and hence in such a way each lecturer would have a preference list and they would be offering only one project whose id is same as the lecturer id.

- The students start as free
- All projects are initially available
- The proposal round :
  - Here we use a lecturer queue, and the loop continues till the queue is empty.
  - dequeue a lecturer, say L_k from the queue, while the lecturer is not full and has students in their preference list not yet proposed to, pick the top-most ranked student (which was not yet proposed) from the lecturers preference list say s_i.
    - Check if s_i has p_k in its preference list(i.e., acceptable or not).
    - if acceptable and s_i, is not yet assigned it accepts.
    - if acceptable, and s_i is already assigned to p_{old}, s_i rejects p_{old}, and tentatively accepts p_k. Add L_{old} back to the lecturer queue.

    * Deletion rule : if p_k was accepted by the student, all the projects ranked lower than p_k must be removed from it.
    * if L_k still has space, it is added back to lecturer queue.
  - This loop continues until all the slots are tentatively assigned to some student or till all the slots have traversed all their preferences.

This gives a **Lecturer-optimal** matching. The algorithm ensures **Guaranteed Termination**.

3.4 Boston Algorithm

In this project, we implemented a customized version of the **Boston Algorithm (Immediate Acceptance Mechanism)** to allocate students to projects based on a priority scoring system. The algorithm processes student applications round-by-round, assigning students to their preferred projects based on weighted criteria.]

Algorithm Overview-
In our implementation, we **score** all student-project pairs using a weighted formula and directly assign students to the highest-ranking projects based on available slots. This is effectively a one-pass, priority-based allocation using the Boston logic.

Scoring System-
Each student's application to a project is assigned a **score** based on the following weighted criteria:

- **CGPA**: Normalized (out of 10) and multiplied by a user-defined weight.
- **Year**: Bonus score if the student's year meets or exceeds the project's minimum required year.

- **Department Match**: Additional points if the student belongs to the same department as the project.
- **Prerequisites Match**: Percentage of completed prerequisite courses for the project, scaled by a weight.

These weights are dynamic and can be configured through the UI or passed as an object to the back-end.

Customizable Priority System by Faculty-

One of the key innovations in our system is that **faculty members can dynamically choose, reorder, or remove** the evaluation criteria that determine student-project allocations. Faculty can drag and reorder the available criteria in the front-end interface. They can also add and remove criteria based on what is important to them. The order of these criteria determines their **relative weight** in the scoring process.

Dynamic Weighting Logic-

The weights are automatically assigned based on the number of selected criteria and their order, as shown below:

Number of Criteria	**Priority 1**	**Priority 2**	**Priority 3**	**Priority 4**
4 (All selected)	50	30	20	10
3 Selected	60	30	10	—
2 Selected	70	30	—	—

These weights are passed to the backend, where the allocation algorithm uses them in the student-project scoring formula.

Chapter 4

Architecture

4.1 Front End

Fronted was made with Angular framework and bootstrap styling. The student screen, faculty screen, admin screen and algorithms visualizer were made as separate components and linked to the common main screen

4.2 Back End

Backend was made with Node.js. All the frontend screens are connected to the backend and the backend is connected to the database.

4.3 Database

Database was made with postgresql