CS 682 Project

Project 1 - A Portal for Managing

Students Capstone Projects

TECHNICAL DOCUMENTATION

Submitted by:

Lohith Reddy Mudipalli

Sri Charan Tadiparthi

Bharath Karumanchi

Online JSDoc Documentation:

https://sricharan0912.github.io/A-Portal-for-Managing-Students-Capstone-Projects-/

Contents

1. Introduction

Capstone projects play a vital role in preparing students for professional practice by bridging theoretical coursework with real-world problem solving. Through these projects, students gain hands-on experience while developing technical proficiency, teamwork, and critical thinking skills. However, managing capstone initiatives effectively—especially when multiple stakeholders such as students, instructors, and external industry clients are involved—can be operationally challenging. Poor coordination, manual workflows, and fragmented communication often reduce efficiency and detract from the educational objectives of such programs.

To address these challenges, we designed and implemented Capstone Hub, a web-based capstone project management platform tailored for software engineering and development courses. The platform introduces a structured, role-based environment that supports seamless collaboration among clients, students, and instructors. External clients can submit project proposals through an intuitive interface, clearly specifying requirements, deliverables, and timelines. Approved projects are then made available to students, who can explore project details and submit ranked preferences using an interactive and user-friendly system.

A key strength of Capstone Hub lies in its automated group formation capability. Using an instructor-defined algorithm, the system assigns students to projects based on their preferences, promoting fairness and transparency while significantly reducing instructor workload. At the same time, the platform allows instructors to manually adjust assignments when necessary, ensuring flexibility in exceptional or pedagogically motivated cases.

Instructors are further supported through a comprehensive dashboard that enables project approval, student monitoring, announcement management, and evaluation scheduling from a single centralized interface. Each user role is provided with a tailored dashboard, ensuring clarity, ease of navigation, and an efficient workflow. The frontend is developed using React and TailwindCSS, delivering a responsive and visually consistent experience.

Security and access control are handled through Firebase Authentication, ensuring reliable role-based authorization and session management. The backend architecture, built with Node.js and Express, exposes RESTful APIs that enable real-time synchronization between the frontend and database. By centralizing project workflows and automating preference handling and group assignment, Capstone Hub enhances the overall learning experience while minimizing administrative overhead. Ultimately, the platform offers a scalable, professional solution that enables institutions to manage capstone programs with the efficiency and robustness of an enterprise-grade system.

2. Problem Statement

Traditional capstone management often suffers from email-based project submissions that lead to lost proposals, manual spreadsheet tracking that is prone to errors, disconnected preference collection via forms, time-intensive manual group formation, and limited transparency in approval status.

Capstone Hub addresses these issues through a centralized project repository with structured submission forms, real-time preference tracking with drag-and-drop ranking, an automated group formation algorithm that optimizes student satisfaction, role-based dashboards that show approval

status and feedback, and a complete audit trail of decisions.

The platform transforms capstone management from manual chaos into an organized, automated workflow.

3. Architecture Overview

Frontend (React + Vite + TailwindCSS)

Component-based React application with role-specific views. Firebase handles authentication, React Router manages navigation, and TailwindCSS provides styling. All user actions trigger REST API calls to the backend.

Key frontend technologies include React 19.0.0 for UI components, Vite 6.0.5 as the build tool, TailwindCSS 4.0.0 for styling, React Router DOM 7.1.1 for routing, and Firebase for authentication.

Backend (Node.js + Express)

RESTful API handling business logic and database operations. Routes are organized by feature (projects, preferences, groups, evaluations). Firebase middleware verifies tokens on protected endpoints.

Route files are organized by feature: `clientRoutes.js` (project submission), `projectRoutes.js` (CRUD operations), `studentRoutes.js` (preferences and assignments), `instructorRoutes.js` (approvals and settings), `groupAlgorithmRoutes.js` (group formation), and `evaluationRoutes.js` (evaluation scheduling).

Database (MySQL)

Relational database using raw SQL queries via mysql2 connection pool. Normalized schema with foreign key constraints.

Core tables include `users` (all users with role column), `user_profiles` (extended user information), `projects` (project proposals with approval status), `student_preferences` (ranked preferences), `student_groups` (formed groups), `group_members` (group membership mapping), `evaluations` (scheduled evaluations), and `app_settings` (system configuration).

System Architecture Diagram

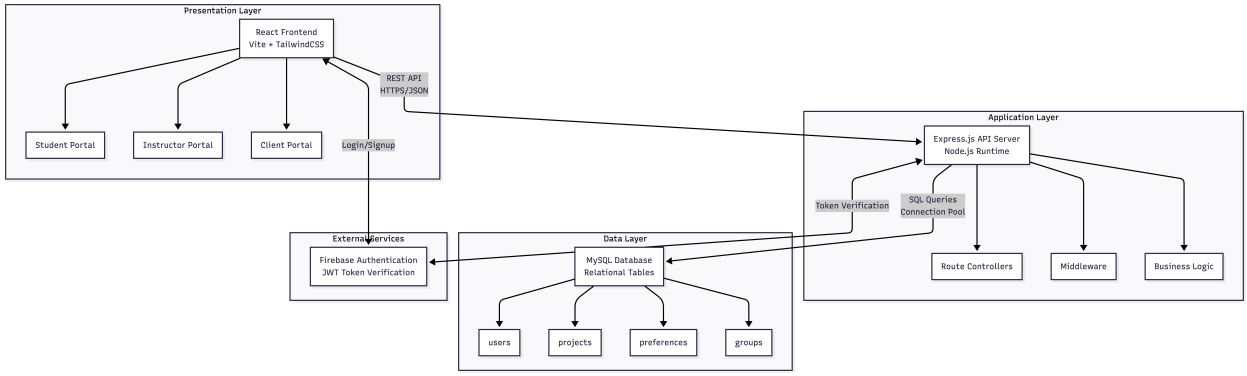

Figure 1: System Architecture Diagram (Presentation, Application, Data Layers, and External Services).

The system follows a three-tier architecture:

Presentation Layer:

- Role-specific React portals (Student, Instructor, Client) plus Public Portal
- Built with React components managing dynamic state and user interactions
- TailwindCSS provides responsive styling across all portals

Application Layer:

- Node.js/Express backend handling business logic and API requests
- Firebase Authentication manages user credentials and JWT token generation
- Middleware verifies tokens and enforces role-based access control

Data Layer:

- MySQL database with normalized schema stores all application data
- Core tables: users, projects, preferences, groups, evaluations
- Foreign key constraints ensure referential integrity and data consistency

External Services:

- Firebase operates independently for authentication
- Backend coordinates all data flow between frontend portals and database

Database ER Diagram

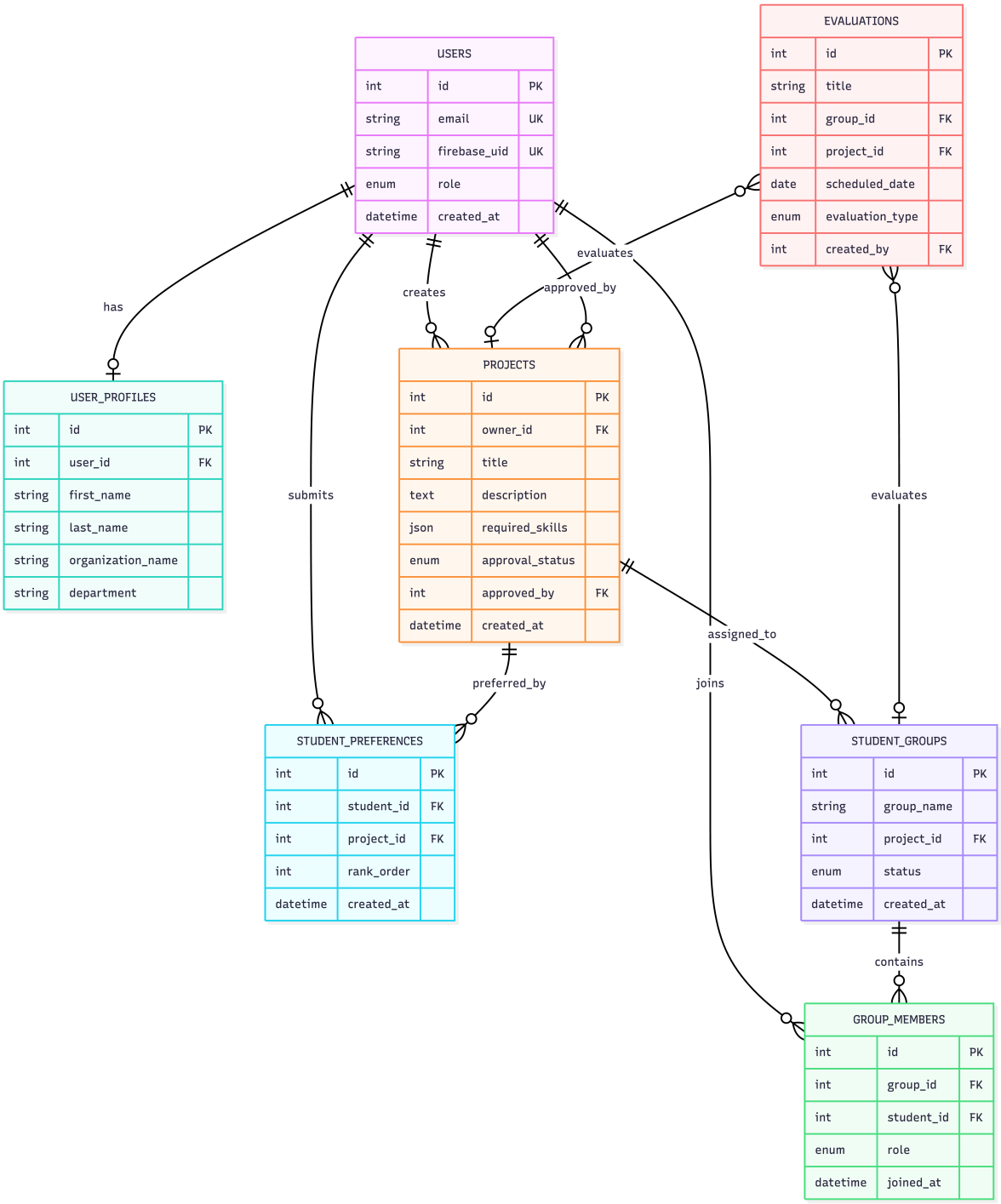

Figure 2: Database ER Diagram showing relationships among Users, Projects, Preferences, Groups, and Evaluations.

Interaction Flow

Interaction flow (high level): a client submits a project via `POST /api/projects` (status: pending), an instructor approves via `PUT /api/projects/:id/approval`, students submit preferences via `POST /api/preferences`, the instructor runs group formation via `POST /api/groups/generate`, students view assignments via `GET /api/student/assigned-project`, and clients view assigned teams via `GET /api/clients/:id/teams`.

Sequence Diagram - Complete Project Lifecycle

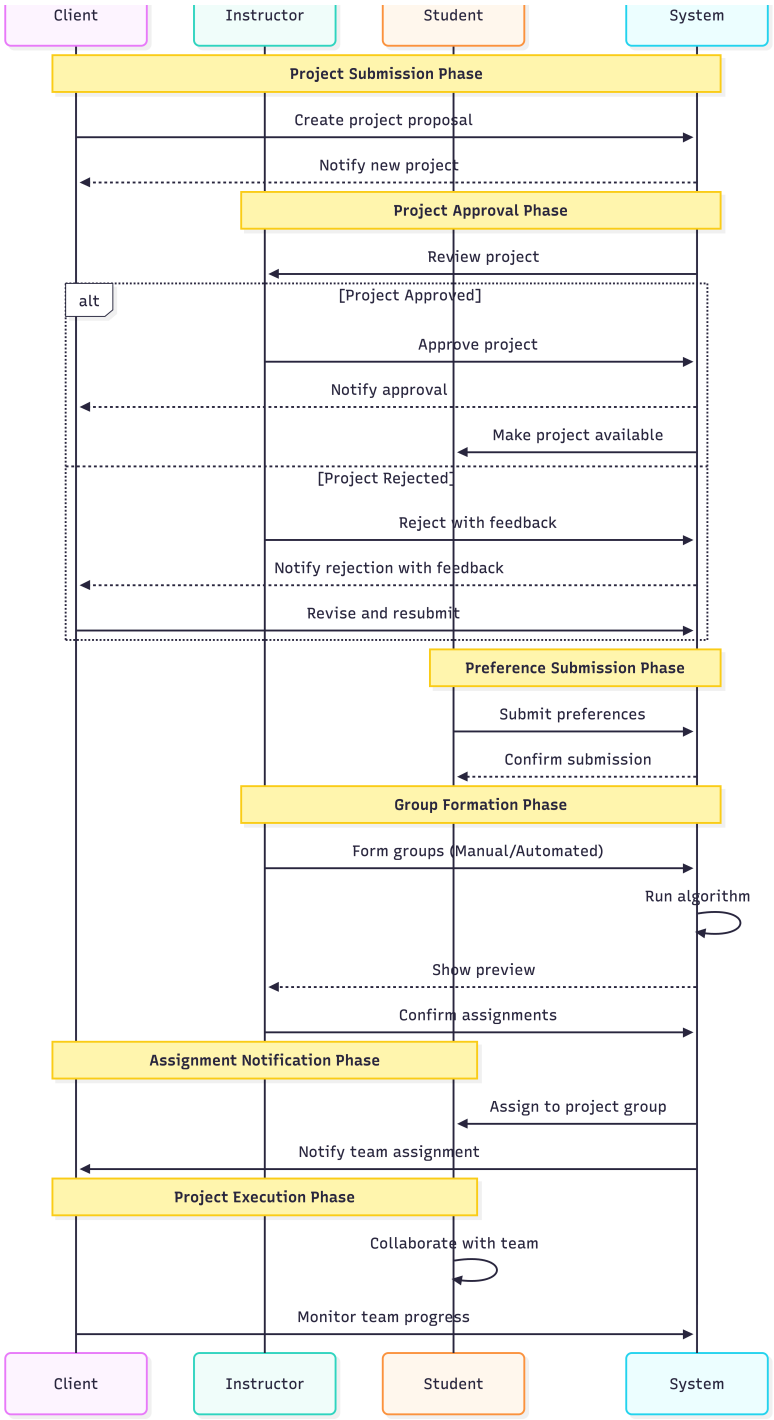

Figure 3: Sequence Diagram for the complete project lifecycle: submission, approval, preferences, group formation, and execution.

The complete workflow involves five distinct phases:

Project Submission Phase:

- Client creates proposal through frontend form
- System stores project with "pending" status
- Instructor receives notification of new submission

Approval Phase:

- Instructor reviews project and makes decision
- Approved projects become available to students
- Rejected projects receive feedback for client revision

Preference Submission Phase:

- Students browse approved projects with full details
- Submit ranked preferences (up to three choices)
- System confirms and stores preference rankings

Group Formation Phase:

- Instructor triggers automated algorithm
- System generates optimal team assignments
- Instructor reviews preview and confirms assignments

Assignment Notification & Execution Phase:

- Students and clients receive team assignment notifications
- Teams begin collaboration on assigned projects
- Instructor monitors progress throughout lifecycle

Security & Authentication

Firebase Authentication provides secure login. JWT tokens stored in localStorage are sent in Authorization headers. Backend middleware verifies tokens and enforces role-based access control.

4. API Endpoints

The backend exposes a RESTful API with role-based access control. All endpoints require Firebase authentication via JWT tokens in Authorization headers.

Authentication and User Management

POST /api/users/register registers new users (Student, Client, Instructor), creates a Firebase account and database entry, and returns the user ID and role.

POST /api/user/verify verifies the Firebase token and returns the user role and database ID for session validation.

GET /api/users/instructors returns a list of all instructors, used by clients when selecting project mentors.

Project Management (Client + Instructor)

GET /api/projects fetches all projects (instructors see all, students see approved only) including title, description, skills, and approval status.

POST /api/projects allows clients to submit new project proposals; status defaults to pending. Required fields include title, description, required_skills, and max_team_size.

PUT /api/projects/:id updates project details (clients can edit before approval only).

DELETE /api/projects/:id deletes projects not yet assigned to groups.

PUT /api/projects/:id/approval allows instructors to approve/reject projects with optional feedback; updates approval status and records approving instructor.

GET /api/projects/client returns projects created by the logged-in client with status and instructor feedback.

GET /api/projects/public returns only approved projects for students during preference selection.

GET /api/project/:id fetches a single project with full details including assigned team if available.

Student Preferences

GET /api/preferences/student returns the current student's preferences with project details ordered by rank.

POST /api/preferences submits student project preferences (up to three project IDs) and validates that projects exist and are approved.

DELETE /api/preferences removes a project from the preference list using projectId as a query parameter.

POST /api/preferences/rank updates the preference ranking order using an ordered array of project IDs.

Group Formation

POST /api/groups/generate triggers the automated group formation algorithm and returns formation statistics.

GET /api/groups/all retrieves all formed groups.

GET /api/groups/:projectId returns the group assigned to a specific project including student

names and contact details.

GET /api/group-members/:groupId returns members of a specific group for the assigned project view.

Evaluations (Instructor)

GET /api/evaluations returns evaluations filtered by role.

POST /api/evaluations creates evaluation events and supports group-specific or project-wide scope.

PUT /api/evaluations/:id updates evaluation details.

DELETE /api/evaluations/:id removes an evaluation (instructors only).

Course Settings (Instructor)

GET /api/instructors/settings/:key retrieves system configuration values (e.g., preference_deadline).

PUT /api/instructors/settings/:key updates system settings and logs changes for audit.

GET /api/instructors/:id/stats returns dashboard statistics (students, projects by status, total groups).

5. Component Modules

React components are organized by user role with shared utilities and routing.

Student Components

StudentDashboard.jsx, ViewProjects.jsx, ProvidePreferences.jsx, AssignedProjects.jsx, StudentSidebar.jsx.

Instructor Components

InstructorDashboard.jsx, InstructorProjects.jsx, ManageGroups.jsx, ManageStudents.jsx, CourseSettings.jsx, InstructorSidebar.jsx.

Client Components

ClientDashboard.jsx, CreateProject.jsx, MyProjects.jsx, ViewTeams.jsx, ClientSidebar.jsx.

Shared & Routing Components

TopbarWithSidebar.jsx, App.jsx, Login.jsx, Signup.jsx, ProtectedRoute.jsx.

6. Role-Based Features

Students

Students can browse approved projects with detailed descriptions and requirements. They submit ranked project preferences through a drag-and-drop interface and later view their assigned project along with team member information and evaluation schedules.

Clients

Clients submit project proposals with structured requirements, track approval status (pending/approved/rejected), review instructor feedback for rejected proposals, and view the assigned student teams once projects are approved.

Instructors

Instructors approve or reject client submissions with feedback, manage preference submissions, run automated group formation, manually adjust group assignments when needed, schedule evaluations, and configure system settings such as deadlines.

Role-Based Feature Flow Diagram

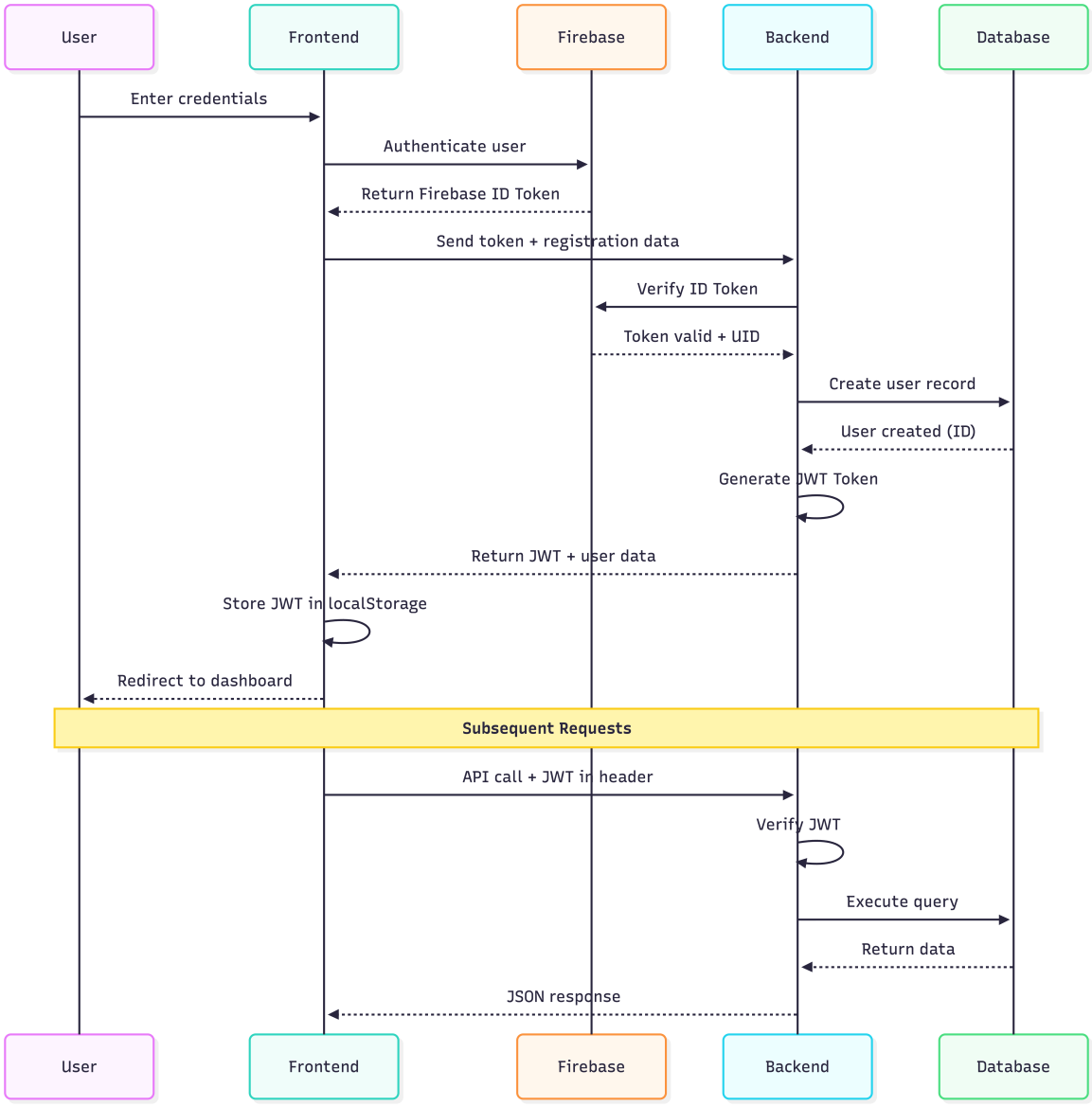

Figure 4: Role-based flow: Student, Instructor, and Client dashboards with supported features.

Authentication and role-specific access flow:

Authentication Process:

- User enters credentials through frontend login
- Firebase authenticates and returns ID token
- Backend verifies token and creates user record in MySQL
- System generates JWT for session management
- JWT stored in localStorage for subsequent requests

Student Access:

- Browse Projects: View all approved project proposals
- Submit Preferences: Rank up to three projects via drag-and-drop
- View Assignments: See assigned project and team members
- Manage Deadlines: Track preference and evaluation schedules

Instructor Access:

- Approve Projects: Review and approve/reject with feedback
- Assign Groups: Run automated algorithm or manual assignment
- Manage Students: Monitor profiles and preference submissions
- Schedule Evaluations: Set up midterm and final reviews
- Configure Settings: Manage deadlines and system parameters

Client Access:

- Submit Projects: Create proposals with requirements
- View Approval Status: Track pending/approved/rejected status
- Monitor Teams: See assigned student groups with contact info
- Edit Projects: Update details before approval

7. Group Formation Algorithm

The group formation algorithm assigns students to projects based on ranked preferences, maximizing overall student satisfaction while respecting each project's maximum team size. A first-choice preference receives three points, a second-choice preference receives two points, and a third-choice preference receives one point. Projects are processed in order of total preference points, and students are assigned by priority of ranking.

Group Formation Algorithm Flowchart

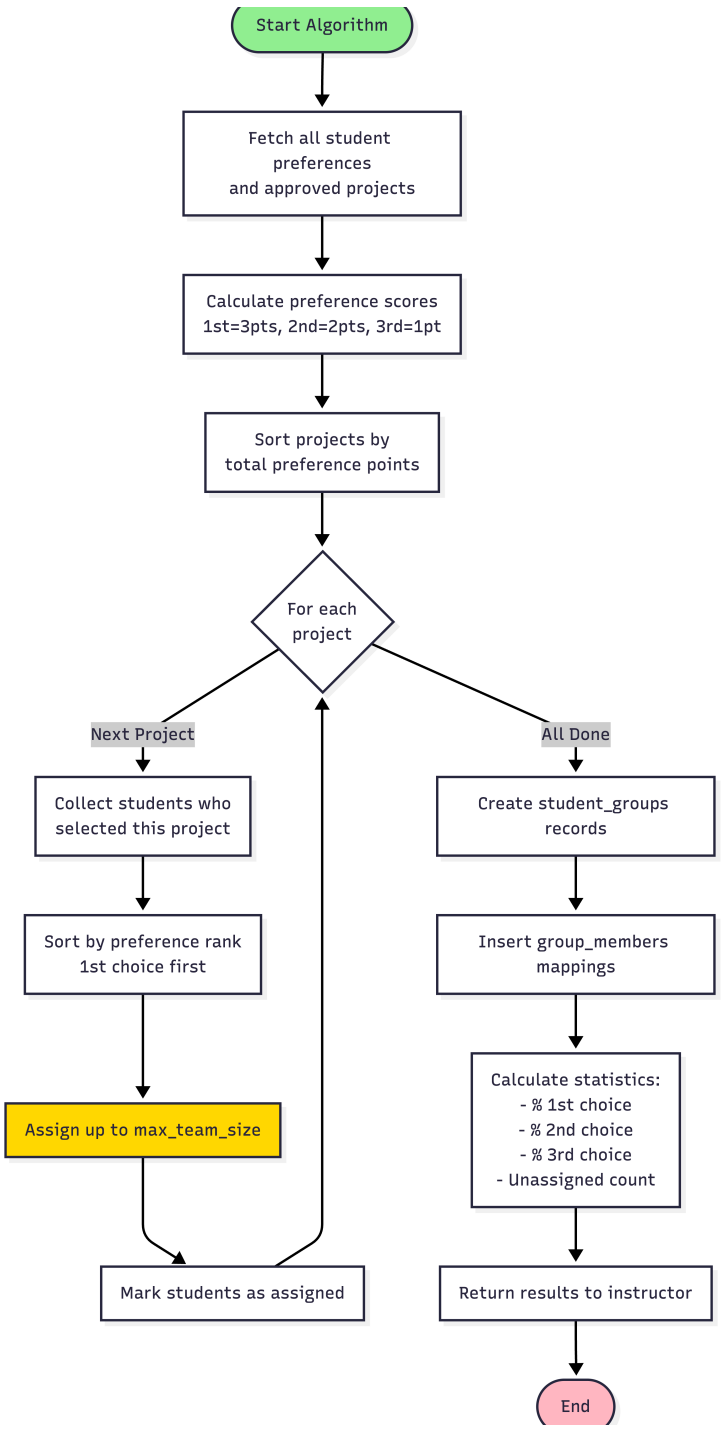

Figure 5: Group formation algorithm flowchart: scoring, sorting, assignment, and statistics reporting.

Algorithm optimizes assignments using weighted preference scoring:

Step 1 - Data Collection:

- Fetch all student preferences from database
- Retrieve all approved projects with capacity limits

Step 2 - Preference Scoring:

- Calculate weighted points: 1st choice = 3pts, 2nd = 2pts, 3rd = 1pt
- Aggregate total preference points for each project

Step 3 - Project Prioritization:

- Sort projects by total preference points (descending)
- High-demand projects processed first

Step 4 - Student Assignment:

- For each project, collect interested students
- Sort students by preference rank (1st choice first)
- Assign up to max_team_size students
- Mark assigned students to prevent duplicates

Step 5 - Database Updates:

- Create student_groups records for each project
- Insert group_members mappings

Step 6 - Statistics Generation:

- Calculate percentage receiving 1st choice
- Calculate percentage receiving 2nd choice
- Calculate percentage receiving 3rd choice
- Count unassigned students
- Return results to instructor for review

8. Authentication Flow

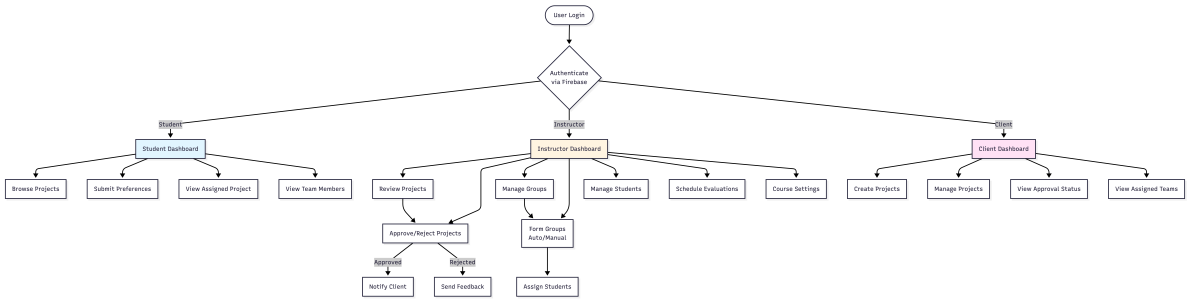

Figure 6: Authentication flow using Firebase authentication and backend JWT session management.

Secure multi-role authentication using Firebase and JWT:

Initial Authentication:

- Users enter credentials at unified sign-in interface
- Frontend sends credentials to Firebase for verification
- Firebase validates and returns ID token upon success

User Registration:

- Frontend sends token with registration data to backend
- Backend verifies token authenticity with Firebase
- System creates user record in MySQL with role information
- Backend generates JWT token for session management

Session Management:

- JWT stored in localStorage on frontend
- All subsequent API requests include JWT in Authorization header
- Backend middleware verifies JWT before processing requests

Role-Specific Routing:

- Students: Dashboard with preferences, assignments, deadlines
- Instructors: Management tools for approvals, groups, settings
- Clients: Project submission, status tracking, team monitoring
- System enforces role-based access at both frontend and backend levels

9. Future Scope

AI-Powered Project Recommendations: Machine learning algorithms might examine student academic records, talent profiles, and career interests to recommend the best project matches, going beyond human preference selection. This clever system will guarantee a fair allocation of skills among teams while assisting students in finding pertinent assignments they might otherwise miss. Additionally, by using past data to forecast team success rates, the program could assist instructors in identifying possibly mismatched assignments prior to finalization. Portfolio System and Client Reputation: Accountability and transparency would be established by putting in place a rating system where finished student teams assess their client experience. Top student preferences would be drawn to clients who exhibit good communication, reasonable expectations, and beneficial mentoring. Future cohorts would be better able to make educated choices when submitting their preferences if they had access to a searchable client portfolio that included student testimonials and previous successful projects. Dynamic Team Rebalancing and Risk Detection: The platform may automatically recommend the best team reassignments based on remaining preferences and project needs if students abandon classes or circumstances change during the semester. Furthermore, by using milestone monitoring with early warning alerts, at-risk projects with little activity would be identified, allowing instructors to take prompt action before problems worsen.

10. Accessing Documentation

Online JSDoc Documentation (Recommended)

The complete API documentation for both frontend and backend is available online via GitHub Pages:

Main Documentation Portal:
https://sricharan0912.github.io/A-Portal-for-Managing-Students-Capstone-Projects-/

This interactive documentation portal provides:

- **Backend API Documentation**: Complete REST API endpoints, controllers, middleware, and database operations with detailed parameter descriptions and return types
- **Frontend Component Documentation**: React components, custom hooks, utilities, and UI implementation details
- **Search Functionality**: Quick search across all documented functions and modules
- **Syntax Highlighting**: Color-coded code examples and function signatures

Accessing from Repository

Alternatively, the documentation can be viewed locally from the GitHub repository:

1. Clone the repository and checkout the `documentation` branch:

```
git clone https://github.com/sricharan0912/
  A-Portal-for-Managing-Students-Capstone-Projects.git
cd A-Portal-for-Managing-Students-Capstone-Projects
git checkout documentation
```

2. Open `docs/index.html` in any modern web browser
3. Navigate to Backend docs at `docs/backend/index.html` or Frontend docs at `docs/frontend/index.html`

Documentation Structure

The documentation follows JSDoc standards with comprehensive coverage:

- **Function Documentation**: All functions include parameter types, return values, and usage examples
- **Module Organization**: Documentation is organized by module (controllers, routes, middleware, utilities)
- **Cross-References**: Linked references between related functions and modules
- **Code Examples**: Real-world usage examples for complex functions

11. Conclusion

Capstone Hub transforms capstone project management from a fragmented, manual process into a streamlined, automated system. By centralizing project submissions, preference collection, and group formation, the platform reduces administrative burden while improving transparency and fairness.

The modular architecture built on React, Node.js, and MySQL provides a solid foundation for future enhancements. Firebase authentication ensures secure access control, while RESTful APIs enable seamless communication between frontend and backend.

Key achievements include a centralized project repository that replaces email-based submissions, automated group formation that saves instructor time and improves student satisfaction, role-based dashboards that ensure relevant access, and an audit trail that supports transparent decision-making.