

Abstract

Overview

The rapid growth of competitive programming and problem-solving platforms has made coding practice an essential part of students' academic journey. While universities teach fundamental subjects like Data Structures and Algorithms, faculty often face problems in direct visibility into students' performance on external coding platforms. This creates a gap between structured classroom learning and independent practice. To address this gap, we propose **MilestoneMe**, a monitoring platform that enables faculty to track, analyze, and support students' coding progress effectively.

Built on the MERN stack (MongoDB, Express.js, React.js, and Node.js), MilestoneMe integrates with coding platforms such as Codeforces to fetch and visualize real-time student activity. The system provides a centralized faculty dashboard where instructors can view student progress, detect inactivity, and send reminders to ensure consistent engagement. By combining automation with manual control, MilestoneMe bridges the gap between student practice and faculty supervision, ultimately strengthening learning outcomes.

Key Features of MilestoneMe

- **Periodically Email Notifications** – Automated system to send motivational emails to inactive students.
- **Inactive Student Detection** – Identifies students with prolonged inactivity (e.g., 7 days) on coding platforms.
- **Automated & Manual Notifications** – Faculty can either rely on automated reminders or send personalized notifications manually.
- **Regular Data Fetching** – The platform fetches coding activity data daily from Codeforces to maintain real-time accuracy.
- **Multiple Visual Representations** – Provides insights through rating distribution graphs, submission frequency charts, heatmaps, and activity summaries to track student consistency and progress.

This project demonstrates how digital monitoring, combined with intelligent notifications and data-driven insights, can foster accountability and strengthen the faculty–student academic relationship.

Contents

Abstract	1
1 Introduction	8
1.1 Problem Identification	8
1.2 Objectives of the Project	9
1.3 Technologies Used	9
1.3.1 MERN Stack Components	9
1.3.2 Supporting Libraries and Tools	10
1.4 Key Features of the System	10
1.4.1 Timely Email Notifications	10
1.4.2 Inactive Student Detection	10
1.4.3 Automated and Manual Communication	10
1.4.4 Real-Time Data Fetching	11
1.4.5 Multiple Visual Representations	11
1.5 Scope of the Project	11
1.5.1 Main Purpose	11
1.5.2 Target Audience	11
1.5.3 Scope Details	11
2 Literature Review	13
2.1 Existing Student Academic Tracking Systems	13
2.1.1 ProgressIQ	13
2.1.2 CodeBuddy	13
2.1.3 GitHub Projects for Student Tracking	14
2.2 Gamified and Interactive Coding Platforms	14
2.2.1 CodeCombat	14
2.2.2 CodeGrade	14
2.2.3 Codio	14
2.2.4 Codesters	14

2.3	Performance Tracking and Analytics Tools	15
2.3.1	Kitaboo	15
2.3.2	BMSCE IEEE Tool	15
2.3.3	Programming Activity Tracking (O'Connor)	15
2.3.4	Watcher (Tanaka)	15
2.3.5	Student Performance Tracking (Sharma)	15
2.3.6	AI-based Monitoring (Gupta)	16
2.3.7	Learning Analytics (Verma)	16
2.4	Summary	16
3	Methodology	17
3.1	Introduction	17
3.2	Pre-Programming Preparation	17
3.2.1	Research of Project	17
3.2.2	Design and Planning	18
3.2.3	Wireframing and Prototyping	18
3.3	Development Process	18
3.3.1	Technology Selection	18
3.3.2	Use of Libraries and Tools	19
3.3.3	UI and Usability Considerations	19
3.4	Development Strategy	19
3.4.1	Exploring Possible Strategies	19
3.4.2	Adoption of Agile Model	19
3.5	Testing Phase	20
3.5.1	Unit Testing	20
3.5.2	Integration Testing	20
3.5.3	System Testing	20
3.6	Faculty and Student Modules	20
3.6.1	Faculty Module	20
3.6.2	Student-Side Application	20
3.7	API Integration	21
3.8	Tools and Platforms Used	21
3.9	Summary	21
4	System Design and Diagrams	22
4.1	Entity-Relationship Diagram	24
4.2	System Architecture	25
4.3	Use Case Diagram	26

4.4	Sequence Diagram	27
4.5	Activity Diagram	28
5	Implementation	29
5.1	Introduction	29
5.2	Project File Structure	30
5.2.1	Frontend Structure	30
5.2.2	Backend Structure	31
5.3	Frontend Implementation	32
5.3.1	Phase 1: Student Section	32
5.3.2	Phase 2: Student Profile Section	32
5.3.3	Phase 3: Dashboard Section	32
5.3.4	Phase 4: Settings Section	33
5.4	Backend Implementation	34
5.4.1	Phase 1: Student Data Management	34
5.4.2	Phase 2: Data Fetching and Contest Tracking	34
5.4.3	Phase 3: Email Notifications	35
5.4.4	Phase 4: Cron Configuration	35
5.5	Database Implementation	35
5.5.1	Students Collection	36
5.5.2	Contests Collection	36
5.5.3	ProblemStat Collection	36
5.5.4	CronConfig Collection	37
5.6	Tools and Technology	37
5.6.1	Frontend Tools and Libraries	37
5.6.2	Backend Tools and Libraries	38
5.6.3	Database	38
5.6.4	Other Tools	38
5.7	Summary	38
6	Final Product screenshots	40
6.1	Dashboard Page	40
6.1.1	Details	40
6.1.2	Dashboard	41
6.2	Students Page	42
6.2.1	Details	42
6.2.2	Students	43
6.3	Student Profile Page	44

6.3.1	Details	44
6.3.2	student profile (1)	45
6.3.3	student profile (2)	46
6.4	Settings Page	47
6.4.1	Details	47
6.4.2	Settings	48
6.5	Special Features Screenshots	49
6.5.1	Automated/Manual Email Screenshot	49
6.5.2	Database Entry Screenshot	50
6.5.3	Terminal Sync Process Screenshot	51
6.5.4	CSV Export Screenshot	52
7	Conclusion	53
7.1	Project Achievements	53
7.2	Technical Learning and Innovations	54
7.3	Challenges and Solutions	55
7.4	Impact and Future Prospects	55
7.5	Project Repository	56
7.6	Conclusion	56
8	Future Plans and Enhancements	58
8.1	Integration with Additional Competitive Programming Platforms	58
8.2	Predictive Analytics and Machine Learning Models	59
8.3	Personalized Problem Recommendation System	59
8.4	Gamification and Leaderboards	59
8.5	Enhanced Visualization and Reporting Tools	60
8.6	Scalability and Multi-Class / Multi-Institution Support	60
8.7	Mobile Application and Cross-Platform Accessibility	60
8.8	Artificial Intelligence for Automated Mentoring	61
8.9	Enhanced Security and Data Privacy Measures	61
8.10	Integration with Collaborative Learning Tools	61
8.11	Conclusion on Future Plans	62

List of Figures

4.1	Entity-Relationship Diagram of the System	24
4.2	System Architecture of the Project	25
4.3	Use Case Diagram of the System	26
4.4	Sequence Diagram of the System	27
4.5	Activity Diagram of the Workflow	28
5.1	Frontend Folder Structure (Rotated)	30
5.2	Backend Folder Structure	31
6.1	Backend Folder Structure	41
6.2	Backend Folder Structure	43
6.3	Backend Folder Structure	45
6.4	Backend Folder Structure	46
6.5	Backend Folder Structure	48
6.6	Screenshot of Automated/Manual Email Sent to Student . . .	49
6.7	Screenshot of Student Data Entry in MongoDB	50
6.8	Screenshot Showing Data Syncing in Terminal	51
6.9	Screenshot of CSV File Download by Faculty	52

Chapter 1

Introduction

1.1 Problem Identification

In the modern academic landscape, faculty members face increasing challenges in monitoring students' learning progress, particularly in competitive programming and problem-solving activities conducted outside the classroom. Traditional methods of performance evaluation rely heavily on periodic examinations, manual observations, or students' self-reports. These methods are insufficient to capture the true consistency and engagement levels of students over time.

Historically, this gap has widened with the rise of online coding platforms such as Codeforces, LeetCode, and CodeChef. While these platforms have empowered students to practice independently, they have simultaneously created difficulties for faculty in tracking real-time engagement and identifying inactive or struggling students. As a result, many students lose momentum due to inactivity or irregular practice, and faculty remain unaware until performance drops significantly.

The exact problem lies in the absence of a centralized, automated, and intelligent monitoring system that can provide faculty with timely insights into student activity, consistency, and progress. This issue not only reduces accountability but also hinders the ability of educators to provide proactive guidance, ultimately affecting students' placement readiness and overall academic growth.

1.2 Objectives of the Project

The primary objective of this project is to design and implement a centralized monitoring platform that enables faculty to:

- Track students' coding activity in real time.
- Identify inactive students and provide timely interventions.
- Streamline the process of sending automated and manual notifications.
- Visualize students' performance trends for effective supervision.
- Foster a collaborative and accountable learning environment.

Through these objectives, the project aims to ensure greater transparency, accountability, and consistency in students' coding practices while reducing the faculty's workload of manual tracking.

1.3 Technologies Used

The project is built using the MERN stack architecture, complemented by several supporting libraries and tools.

1.3.1 MERN Stack Components

- **MongoDB (Database Layer):** Stores student records, coding activity, and faculty dashboards.
- **Express.js (Backend Framework):** Handles API routing, user authentication, and notification triggers.
- **React.js (Frontend Framework):** Builds an interactive and dynamic user interface with modular components.
- **Node.js (Runtime Environment):** Provides asynchronous processing and scheduling of recurring tasks.

1.3.2 Supporting Libraries and Tools

- **Recharts:** For visualizing data in interactive charts and graphs.
- **node-fetch:** For fetching student performance data from coding platform APIs.
- **node-cron:** For scheduling background tasks such as daily synchronization and notifications.
- **nodemailer:** For sending automated and manual notification emails to inactive students.
- **json2csv:** For exporting performance data into structured CSV/Excel formats.

1.4 Key Features of the System

The project incorporates several essential features designed to maximize efficiency and effectiveness:

1.4.1 Timely Email Notifications

Automated emails are sent to inactive students, reminding them to resume coding activity. Faculty also have the option to send manual notifications when necessary.

1.4.2 Inactive Student Detection

The system identifies students who have been inactive for a defined period (e.g., 7 days), ensuring early intervention to prevent disengagement.

1.4.3 Automated and Manual Communication

Faculty members can rely on automated alerts while still retaining control to send customized manual notifications based on specific needs.

1.4.4 Real-Time Data Fetching

Student activity is regularly synchronized with coding platforms such as Codeforces, maintaining a real-time reflection of their progress.

1.4.5 Multiple Visual Representations

The system provides various graphical insights, including submission frequency charts, rating distributions, and activity heatmaps, helping faculty assess consistency and growth patterns.

1.5 Scope of the Project

The scope of this project outlines its purpose, target audience, and implementation boundaries:

1.5.1 Main Purpose

The system aims to bridge the gap between students' self-learning efforts and faculty supervision, ensuring accountability and continuous improvement in coding practices.

1.5.2 Target Audience

- **Faculty Members:** For effective monitoring and timely intervention.
- **Students:** For tracking their learning journey and identifying weak areas.
- **Institutions:** For standardizing student progress monitoring at scale.

1.5.3 Scope Details

- Integration with Codeforces initially, with future support for LeetCode, CodeChef, HackerRank, and AtCoder.
- Automated and manual notifications for balanced supervision.
- Visual analytics and data export features for institutional use.

- Scalability to include AI-driven recommendations, gamification, and predictive analytics.

In summary, the project provides a scalable, adaptable, and practical solution for academic monitoring in coding education, meeting the evolving needs of modern learning environments.

Chapter 2

Literature Review

In this chapter, we review existing tools, platforms, and research work related to student performance tracking, academic monitoring, and programming activity analysis. The purpose of this review is to understand the strengths, limitations, and contributions of different systems so that our project can build upon them and fill the identified gaps.

2.1 Existing Student Academic Tracking Systems

2.1.1 ProgressIQ

ProgressIQ provides an academic tracking system widely used by educational institutions. It enables faculty to monitor student progress, identify struggling learners, and provide timely interventions. Its strength lies in presenting academic data in a simplified and actionable way, though it primarily focuses on academic records rather than practical coding activity.

2.1.2 CodeBuddy

CodeBuddy, as described by Bogomolov et al., is a programming assignment management system. It is designed to streamline assignment submissions, evaluation, and feedback. The platform emphasizes automation and reduces the burden on instructors. However, it is more aligned with assignment workflows rather than holistic student progress across multiple platforms.

2.1.3 GitHub Projects for Student Tracking

Two notable GitHub projects by Zaidi and Chakraborty demonstrate open-source approaches to student performance monitoring. These projects experiment with tracking performance, though they are limited in scalability and are better suited for learning and experimentation rather than institutional adoption.

2.2 Gamified and Interactive Coding Platforms

2.2.1 CodeCombat

CodeCombat offers a gamified learning environment where students learn coding by playing games. It includes a teacher dashboard for progress tracking. While engaging, it is primarily aimed at K-12 learners, making it less suitable for advanced programming education.

2.2.2 CodeGrade

CodeGrade focuses on creating an engaging code learning platform that integrates with learning management systems. It automates assessment and provides detailed student feedback. Its value lies in its strong integration features and feedback mechanisms.

2.2.3 Codio

Codio provides a hands-on coding environment combined with a learning management system. It allows teachers to design learning modules, track coding progress, and assess student work. The platform is particularly useful in structured course delivery.

2.2.4 Codesters

Codesters brings coding into classrooms by providing a user-friendly platform for teachers and students. While it is more suited for school-level education, its simplicity and interactive approach make it an accessible way to introduce coding concepts.

2.3 Performance Tracking and Analytics Tools

2.3.1 Kitaboo

Kitaboo focuses on performance tracking for personalized and individualized instruction. By leveraging analytics, it helps educators tailor teaching strategies to different learners. However, it is more geared toward digital publishing and e-learning rather than programming education specifically.

2.3.2 BMSCE IEEE Tool

The BMSCE IEEE project introduces a tool that tracks student performance across online platforms. This work highlights the importance of consolidating multi-platform data, which is highly relevant to our project objective of monitoring diverse coding activities.

2.3.3 Programming Activity Tracking (O'Connor)

O'Connor's research emphasizes tracking student programming activity to assist instructors during exercises. It highlights the importance of monitoring not only outcomes but also the process, providing insights into how students approach problem-solving.

2.3.4 Watcher (Tanaka)

Watcher is a cloud-based activity tracker for coding education. It aims at fairness and convenience in monitoring online programming tasks. Its design shows the potential of cloud solutions in scalable education technologies, which inspires part of our work.

2.3.5 Student Performance Tracking (Sharma)

Sharma's system focuses on building an academic and performance monitoring platform, integrating multiple student metrics. This contributes to a holistic view of progress beyond just coding or assignments.

2.3.6 AI-based Monitoring (Gupta)

Gupta introduces an AI-driven approach to monitor student performance and apply preventive interventions. This shows the scope of intelligent analytics in academic monitoring, moving beyond static reports into actionable insights.

2.3.7 Learning Analytics (Verma)

Verma’s work applies learning analytics in LMS to track student engagement and progress. This reflects the growing importance of data-driven decision-making in education and underlines how analytics can enhance traditional teaching methods.

2.4 Summary

The reviewed literature demonstrates a wide range of approaches—from academic tracking systems to gamified platforms, assignment managers, and AI-driven tools. While each system contributes valuable features, a common gap is the lack of an integrated, multi-platform solution that consolidates student activity and generates personalized insights. Our proposed system seeks to address this gap by combining academic tracking, coding activity monitoring, and intelligent recommendations.

Chapter 3

Methodology

3.1 Introduction

This chapter describes the methodology adopted for the development of the project **MilestoneMe**. The approach involved multiple stages, starting from project research and requirement analysis to design, development, testing, and deployment. Each stage was carefully planned and executed to ensure that the project was efficient, scalable, and user-friendly.

3.2 Pre-Programming Preparation

3.2.1 Research of Project

Before starting development, significant effort was invested in researching the need for such a platform. Discussions were held with faculty members, mentors, and students to identify challenges faced in monitoring skill-oriented programming activities. The following issues were commonly highlighted:

- Lack of a centralized system for faculty to track student performance across coding platforms.
- Difficulty in identifying inactive or irregular students.
- Limited insight into trends such as consistency, streaks, and submission frequency.

This feedback helped establish the necessity of a system that could integrate student performance data and present it in an accessible way.

3.2.2 Design and Planning

The next step involved outlining the structure of the platform. Wireframes and prototypes were created to visualize the layout of major components such as the dashboard, student profiles, and settings. Planning ensured that the design remained simple and intuitive so faculty members could quickly learn how to use the system.

3.2.3 Wireframing and Prototyping

Initial wireframes were drawn to map the interface. These were later converted into prototypes that simulated how different sections—such as the dashboard, student details tab, and profile view—would look and function. This phase helped refine the user experience before actual development began.

3.3 Development Process

3.3.1 Technology Selection

Multiple options were considered for developing the platform, including Python-based frameworks and PHP solutions. After evaluation, the MERN stack was selected because it provides:

- End-to-end JavaScript support, reducing complexity.
- Scalability for handling large datasets.
- A responsive and modular frontend through React.
- Efficient API handling with Express.js and Node.js.
- Flexibility in handling diverse data using MongoDB.

3.3.2 Use of Libraries and Tools

Several libraries and tools were integrated to extend core functionality:

- **Recharts:** For data visualization and graphs.
- **Node-fetch:** For fetching external API data.
- **Node-cron:** For scheduling background tasks.
- **Nodemailer:** For sending automated and manual email notifications.
- **json2csv:** For exporting performance data in CSV format.

3.3.3 UI and Usability Considerations

The design was intentionally kept lightweight, ensuring faculty members could learn the system within an hour. Simplicity and clarity were prioritized to increase adoption and minimize training requirements.

3.4 Development Strategy

3.4.1 Exploring Possible Strategies

Several development strategies were considered:

- **Waterfall Model:** Too rigid and unsuitable for evolving requirements.
- **Spiral Model:** Complex and resource-intensive for a medium-scale project.
- **V-Model:** Overly focused on testing, with limited flexibility for iterations.

3.4.2 Adoption of Agile Model

The Agile methodology was chosen due to its iterative nature, flexibility, and suitability for team collaboration. Development was divided into phases:

1. Student section: Faculty can view students in a tabular format.

2. Student profile section: Faculty can check individual student activity on coding platforms.
3. Dashboard: Provides an overall view of class performance and progress.
4. Settings: Allows synchronization and configuration of notifications.

Free APIs from Codeforces were integrated and tested using sample student accounts to validate the working system.

3.5 Testing Phase

3.5.1 Unit Testing

Each individual tab (dashboard, student profile, settings) was tested to ensure proper functionality.

3.5.2 Integration Testing

The integration of APIs, database, and UI components was tested to check the system's stability and performance.

3.5.3 System Testing

End-to-end testing was conducted to validate the overall functionality, ensuring that all modules worked together seamlessly.

3.6 Faculty and Student Modules

3.6.1 Faculty Module

The primary functionality of the system lies in the faculty module. Faculty can monitor student activity, visualize progress, and send manual or automated notifications.

3.6.2 Student-Side Application

The student side is minimal and designed primarily to receive notifications. Its functionality is limited compared to the faculty module.

3.7 API Integration

The project relied on Codeforces free APIs to fetch real-time student activity. These APIs were periodically called using background jobs, ensuring data consistency and accuracy across the system.

3.8 Tools and Platforms Used

The following tools and platforms were used during the project:

- **Visual Studio Code (VS Code):** Primary IDE used for coding.
- **Postman:** For API testing and debugging.
- **MongoDB Compass:** For database visualization and management.
- **Git and GitHub:** For version control and collaboration.
- **Vercel:** Deployment platform for hosting the frontend.

These tools were chosen for their wide adoption, compatibility with the MERN stack, and ease of use.

3.9 Summary

This chapter described the complete methodology followed for developing MilestoneMe. Beginning with project research and requirement gathering, the process moved through design, prototyping, technology selection, and development. Agile methodology allowed iterative progress, and thorough testing ensured reliability. Finally, the selection of modern tools and deployment platforms enabled efficient development, testing, and hosting of the project.

Chapter 4

System Design and Diagrams

Introduction

This chapter presents the **system design** of our project in detail, supported by a series of carefully constructed diagrams. System design plays a critical role in bridging the gap between conceptual understanding and actual implementation, ensuring that both technical and non-technical stakeholders can clearly visualize how the platform functions.

The diagrams included in this chapter not only describe the **static structure** of the system, such as entities and their relationships, but also capture the **dynamic workflows** and **process interactions** that define how data flows across different components. By representing multiple perspectives of the system, these diagrams serve as a blueprint for development, validation, and further enhancements.

The following visual representations are included:

1. **Entity–Relationship (ER) Diagram** – Describes the database schema, entities, attributes, and relationships between core modules like faculty, students, activities, and notifications.
2. **System Architecture Diagram** – Explains the high-level interaction between frontend, backend, database, and external services such as Codeforces API and email services.
3. **Use Case Diagram** – Highlights the actions that faculty users can perform, including student management, viewing progress, and sending notifications.

4. **Sequence Diagram** – Illustrates the flow of communication between frontend, backend, and database during student activity tracking and notification delivery.
5. **Activity/Flow Diagram** – Depicts the logical workflow of key processes such as data fetching, inactivity detection, and automated notifications.
6. **Agile Process Cycle Diagram** – Visualizes the methodology adopted for the project, emphasizing iterative development and feedback-based improvements.

Together, these diagrams provide a **comprehensive understanding** of how the system is structured, how it behaves under different scenarios, and how various components collaborate to achieve the project's objectives. They not only validate the correctness of the design but also make the system more approachable for faculty members, mentors, and future developers who may extend or maintain the platform.

4.1 Entity-Relationship Diagram

The Entity-Relationship (ER) Diagram shows the entities in the system and their relationships.

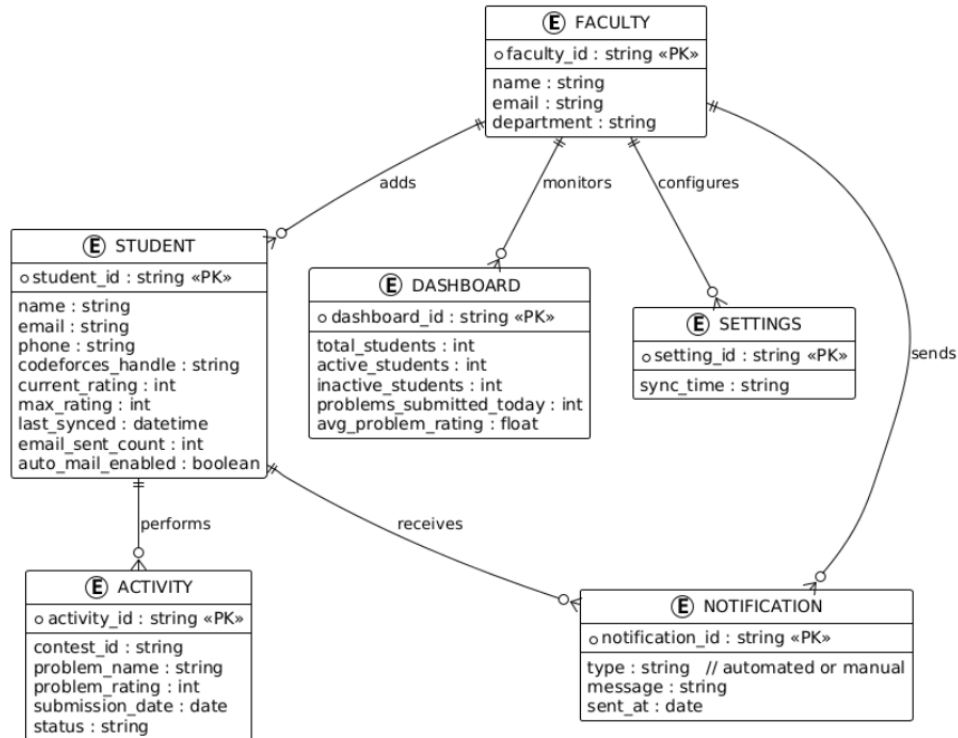


Figure 4.1: Entity-Relationship Diagram of the System

4.2 System Architecture

The system architecture explains the overall structure of the project and the interaction between its components.

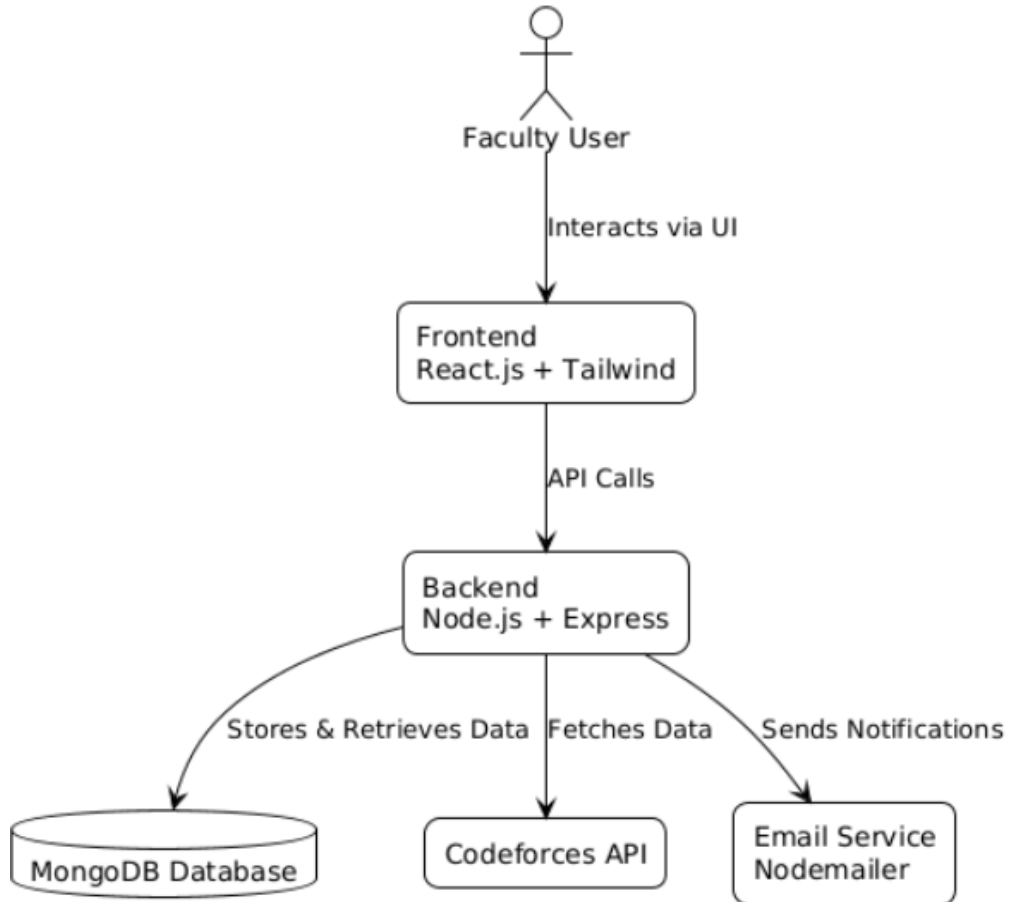


Figure 4.2: System Architecture of the Project

4.3 Use Case Diagram

The Use Case Diagram highlights the interactions between the user (faculty) and the system modules.

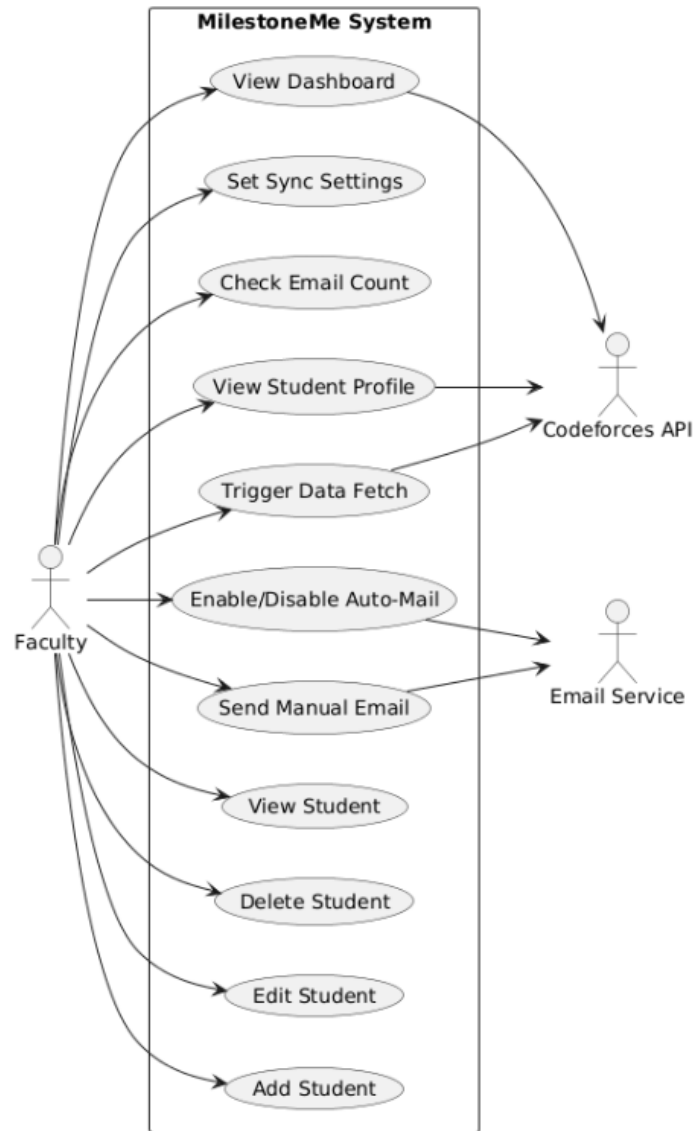


Figure 4.3: Use Case Diagram of the System

4.4 Sequence Diagram

The Sequence Diagram represents the flow of communication between the faculty and the system for specific tasks.

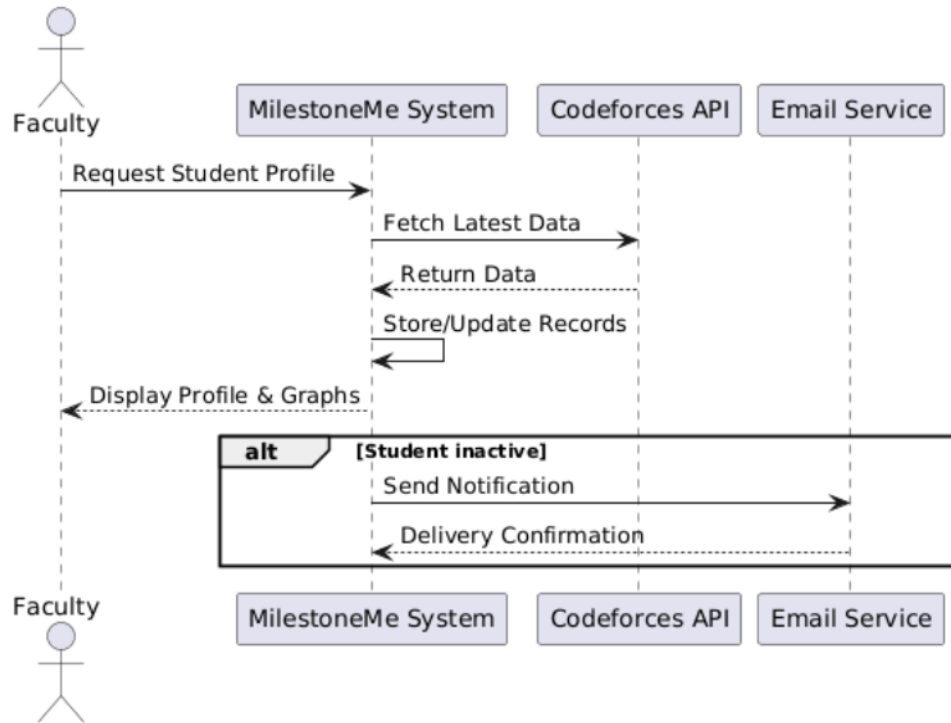


Figure 4.4: Sequence Diagram of the System

4.5 Activity Diagram

The Activity Diagram shows the workflow of data fetching, inactivity detection, and notification sending.

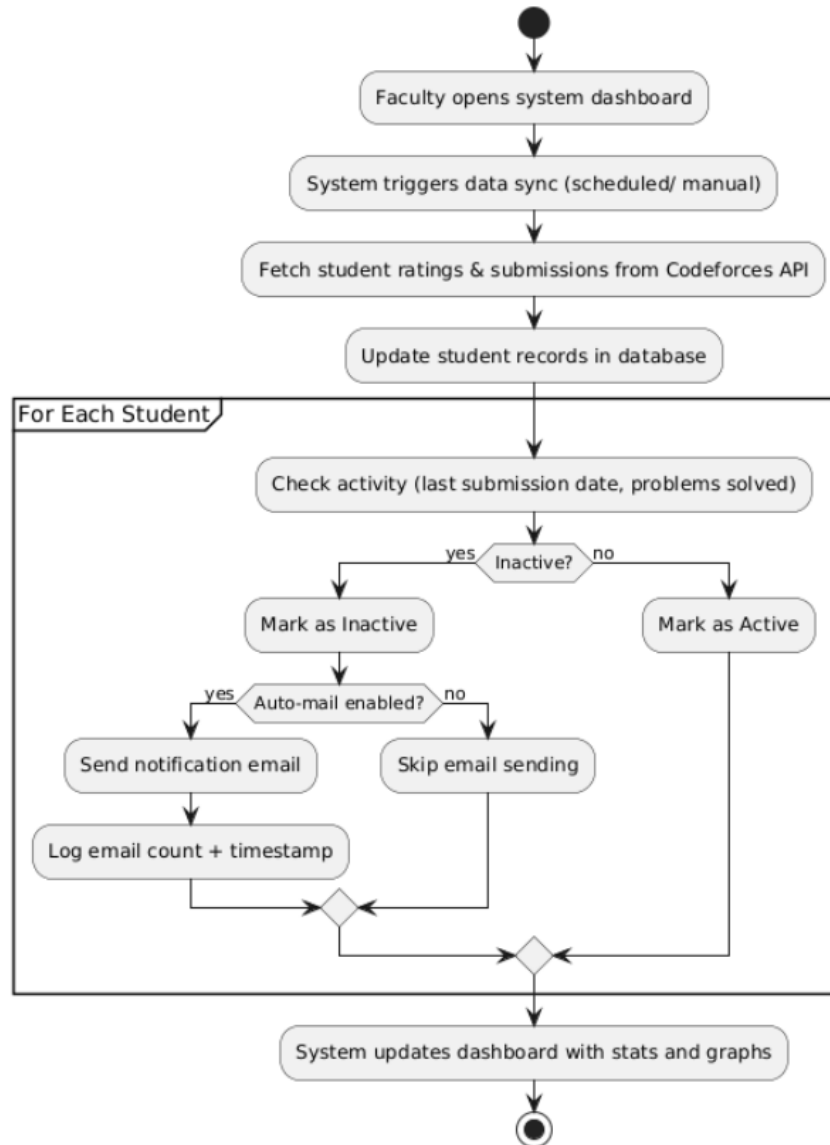


Figure 4.5: Activity Diagram of the Workflow

Chapter 5

Implementation

5.1 Introduction

In this chapter, the focus is on how the **MilestoneMe** project was implemented from start to finish. It provides a detailed description of the **development process**, covering the complete **file structure**, **frontend and backend architecture**, **database design**, and the **tools and technologies** used at each phase of the project.

The goal of this chapter is to give a clear understanding of how the system was built, how different components interact with each other, and the rationale behind choosing specific libraries, frameworks, and services. Each section highlights the functionality and organization of the project, making it easier for anyone reviewing the project to understand its workflow and technical implementation.

5.2 Project File Structure

5.2.1 Frontend Structure

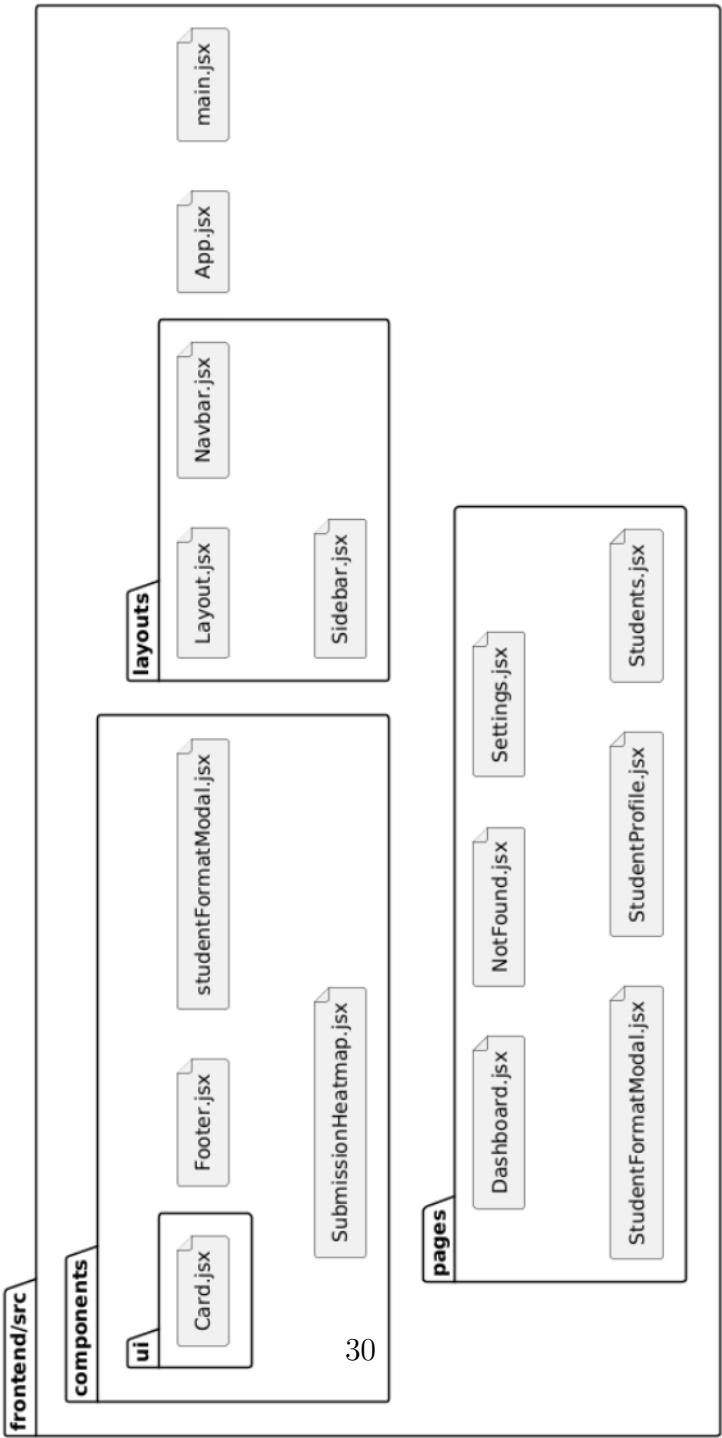


Figure 5.1: Frontend Folder Structure (Rotated)

5.2.2 Backend Structure

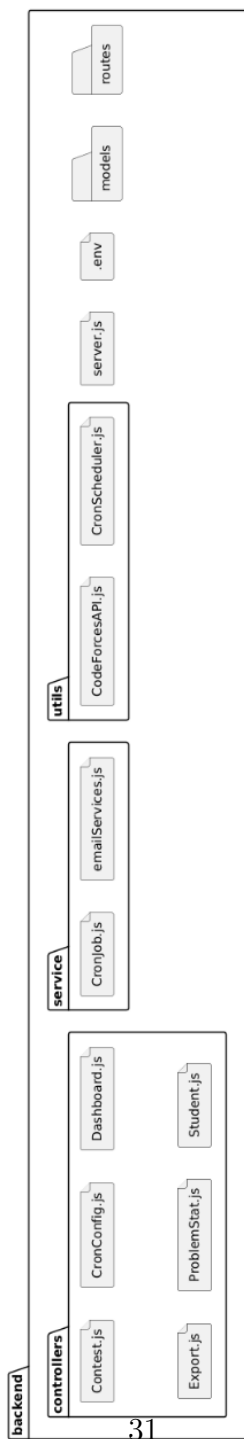


Figure 5.2: Backend Folder Structure

5.3 Frontend Implementation

The frontend of **MilestoneMe** was developed using **React.js**, following a **component-based architecture** to ensure reusability and maintainability. The frontend provides faculty with an intuitive interface to manage students, view their performance, and customize system settings. The development was carried out in **four phases**:

5.3.1 Phase 1: Student Section

In this phase, the **Students section** was implemented, allowing faculty to:

- Add new students to the platform.
- View students' data in a tabular format including Name, Email, Phone, Codeforces ID, Highest Rank, Average Rank, Notifications Sent, and Email Settings.
- Perform actions such as View, Delete, Edit, and Send Mail for each student.

5.3.2 Phase 2: Student Profile Section

This module allows faculty to track individual student activity:

- Consistency graph to visualize activity over time.
- Table of problems solved.
- Performance cards showing total problems solved, most difficult problem rating, average rating, and average problems solved per day.
- Bar graph showing number of problems solved per day.
- Submission heatmap for visualizing student activity trends.

5.3.3 Phase 3: Dashboard Section

The dashboard provides a high-level overview:

- Three summary boxes: Total Students, Active Students, Inactive Students.

- Rating distribution bar graph.
- Line graph showing number of problems solved per day across all students.

5.3.4 Phase 4: Settings Section

Faculty can configure system behavior:

- Syncing time for automatic data fetching and updates.

Technologies Used: React.js, Tailwind CSS, Recharts, React Router DOM.

5.4 Backend Implementation

The backend is developed using **Node.js** and **Express.js** with **MongoDB**. It handles data storage, server-side logic, API endpoints, scheduled tasks, and email notifications.

5.4.1 Phase 1: Student Data Management

The **Student model** stores details including Codeforces handle, ratings, and email preferences:

```
const studentSchema = new Schema({
  name: { type: String, required: true },
  email: { type: String, required: true },
  phone: String,
  codeforcesHandle: { type: String, required: true, unique: true },
  currentRating: Number,
  maxRating: Number,
  lastSynced: Date,
  remindersSent: { type: Number, default: 0 },
  emailRemindersEnabled: { type: Boolean, default: true }
}, { timestamps: true });
```

5.4.2 Phase 2: Data Fetching and Contest Tracking

Scheduled cron jobs fetch contest and problem-solving data from Codeforces:

Contest Model:

```
const contestSchema = new Schema({
  student: { type: Schema.Types.ObjectId, ref: 'Student', required: true },
  contestName: { type: String, required: true },
  contestId: { type: String, required: true },
  rank: { type: Number, required: true },
  ratingChange: Number,
  oldRating: Number,
  newRating: Number,
  ratingUpdateTime: Date,
  problemsUnsolved: Number,
  problems: [String]
```

```
}, { timestamps: true });
```

ProblemStat Model:

```
const problemStatSchema = new Schema({
  student: { type: Schema.Types.ObjectId, ref: 'Student', required: true },
  problemId: String,
  name: String,
  rating: Number,
  tags: [String],
  solvedAt: Date
}, { timestamps: true });
```

5.4.3 Phase 3: Email Notifications

Automated emails are sent to inactive students via **emailServices.js**, and later manual email functionality was added for faculty.

5.4.4 Phase 4: Cron Configuration

Faculty can configure automatic data fetching using the **CronConfig model**:

```
const cronConfigSchema = new Schema({
  frequency: { type: String, default: 'daily' },
  time: { type: String, default: '02:00' },
  timezone: { type: String, default: 'Asia/Kolkata' },
  lastRunAt: Date,
  nextRunAt: Date
}, { timestamps: true });
```

Libraries Used: Node.js, Express.js, Mongoose, dotenv, Node-Cron, Axios, Nodemailer, Cors.

5.5 Database Implementation

The database is implemented using **MongoDB**. It stores student data, contest information, problem statistics, and cron configurations.

5.5.1 Students Collection

```
{
  "_id": "64f9a3b2c8f1a72d5e6f9e01",
  "name": "John Doe",
  "email": "john@example.com",
  "phone": "9876543210",
  "codeforcesHandle": "john123",
  "currentRating": 1800,
  "maxRating": 1850,
  "lastSynced": "2025-09-09T05:00:00Z",
  "remindersSent": 2,
  "emailRemindersEnabled": true,
  "createdAt": "2025-09-01T03:00:00Z",
  "updatedAt": "2025-09-09T05:00:00Z"
}
```

5.5.2 Contests Collection

```
{
  "_id": "64f9a5d3c8f1a72d5e6f9e10",
  "student": "64f9a3b2c8f1a72d5e6f9e01",
  "contestName": "Codeforces Round #900",
  "contestId": "900",
  "rank": 15,
  "ratingChange": 25,
  "oldRating": 1800,
  "newRating": 1825,
  "ratingUpdateTime": "2025-09-08T12:00:00Z",
  "problemsUnsolved": 2,
  "problems": ["A", "B", "C", "D", "E"],
  "createdAt": "2025-09-08T12:30:00Z",
  "updatedAt": "2025-09-08T12:30:00Z"
}
```

5.5.3 ProblemStat Collection

```
{
```

```

    "_id": "64f9a7e1c8f1a72d5e6f9e21",
    "student": "64f9a3b2c8f1a72d5e6f9e01",
    "problemId": "1234A",
    "name": "Two Sum",
    "rating": 1500,
    "tags": ["implementation", "arrays"],
    "solvedAt": "2025-09-07T10:00:00Z",
    "createdAt": "2025-09-07T10:05:00Z",
    "updatedAt": "2025-09-07T10:05:00Z"
  }

```

5.5.4 CronConfig Collection

```

{
  "_id": "64f9a9f4c8f1a72d5e6f9e32",
  "frequency": "daily",
  "time": "02:00",
  "timezone": "Asia/Kolkata",
  "lastRunAt": "2025-09-08T02:00:00Z",
  "nextRunAt": "2025-09-09T02:00:00Z",
  "createdAt": "2025-09-01T03:00:00Z",
  "updatedAt": "2025-09-08T02:00:00Z"
}

```

This database structure ensures efficient storage, easy querying, and a clear mapping between backend models and frontend functionalities.

5.6 Tools and Technology

During the development of **MilestoneMe**, a range of tools, libraries, and frameworks were used to ensure a robust, scalable, and user-friendly application. These are categorized below:

5.6.1 Frontend Tools and Libraries

- **React.js** - For building the dynamic and responsive user interface.
- **Tailwind CSS** - For clean, responsive, and modular styling.

- **Recharts** - For data visualization (graphs, charts, heatmaps).
- **React Router DOM** - For routing and navigation between pages.

5.6.2 Backend Tools and Libraries

- **Node.js** - Server-side JavaScript runtime.
- **Express.js** - Web framework to create API endpoints and server logic.
- **Mongoose** - MongoDB object modeling and schema management.
- **dotenv** - Environment variable management.
- **Node-Cron** - To schedule and manage automated tasks.
- **Axios / node-fetch** - Fetching data from external APIs (Codeforces).
- **Nodemailer** - Sending automated emails to students.
- **Cors** - Handling cross-origin requests.

5.6.3 Database

- **MongoDB** - NoSQL database to store students, contests, problem statistics, and cron configurations.

5.6.4 Other Tools

- **PlantUML** - For creating backend and frontend folder structure diagrams.

—

5.7 Summary

In this chapter, the implementation of **MilestoneMe** was presented in detail, covering the following aspects:

- **File Structure** – A clear and organized structure for frontend and backend components.

- **Frontend Implementation** – Phase-wise development of the Students section, Student Profile, Dashboard, and Settings.
- **Backend Implementation** – Models, APIs, cron jobs, automated/manual email notifications, and data syncing configuration.
- **Database Design** – MongoDB collections for Students, Contests, ProblemStat, and CronConfig with JSON document examples.
- **Tools and Technology** – All libraries, frameworks, and tools used in the development process.

The implementation phase demonstrates a **modular, scalable, and automated system** that allows faculty to manage students, monitor their performance, and configure system behavior effectively. The architecture and database design ensure smooth interaction between the frontend and backend while supporting analytics and notifications.

Chapter 6

Final Product screenshots

6.1 Dashboard Page

6.1.1 Details

The **Dashboard** provides a high-level overview of student performance. Features include:

- Three summary boxes displaying:
 - Total Students
 - Active Students
 - Inactive Students
- Rating Distribution Bar Graph - Shows the spread of student ratings.
- Problems Solved Line Graph - Displays the number of problems solved per day.

6.1.2 Dashboard



Figure 6.1: Backend Folder Structure

6.2 Students Page

6.2.1 Details

The **Students** page allows faculty to manage and view all student information. Features:

- Add new students.
- Tabular view of students with fields:
 - Name, Email, Phone Number, Codeforces ID
 - Highest Rank, Average Rank
 - Notifications Sent, Enable/Disable Automatic Mailing
- Action buttons for each student: View, Edit, Delete, Send Mail

6.2.2 Students

MilestoneMe

Dashboard

Students

Settings

Dashboard

Student Management

Search by name...

Export CSV

+ Add Student

NAME	EMAIL	PHONE	CF HANDLE	CURRENT	MAX	LAST SYNCED	REMINDERS	EMAIL ALERTS	ACTIONS
Uday Gandhi	2203031450007@paruluniversity.ac.in	9313561902	Beng	3588	3833	08 Sept 2025, 06:06 pm	11	Disabled	<div>View</div> <div>Edit</div> <div>Email</div> <div>Delete</div>
Vraj Rathva	2203031450020@paruluniversity.ac.in	9638015118	LJC00118	3401	3401	08 Sept 2025, 06:08 pm	8	Disabled	<div>View</div> <div>Edit</div> <div>Email</div> <div>Delete</div>
Nevil Modi	2203031450013@paruluniversity.ac.in	7861055422	ksun48	3581	3715	08 Sept 2025, 06:14 pm	2	Disabled	<div>View</div> <div>Edit</div> <div>Email</div> <div>Delete</div>
Yash Chauhan	2203031450005@paruluniversity.ac.in	7984303079	orzdevinwang	3669	3844	08 Sept 2025, 06:18 pm	6	Disabled	<div>View</div> <div>Edit</div> <div>Email</div> <div>Delete</div>

Figure 6.2: Backend Folder Structure

6.3 Student Profile Page

6.3.1 Details

The **Student Profile** page provides detailed analytics for individual students:

- Consistency Graph - Shows student's activity over time.
- Problem Solving Table - List of all solved problems.
- Performance Cards:
 - Total Problems Solved
 - Most Difficult Problem Rating
 - Average Rating
 - Average Problems Solved per Day
- Bar Graph - Number of problems solved per day.
- Submission Heatmap - Visual representation of problem-solving activity.

6.3.2 student profile (1)



Figure 6.3: Backend Folder Structure

6.3.3 student profile (2)

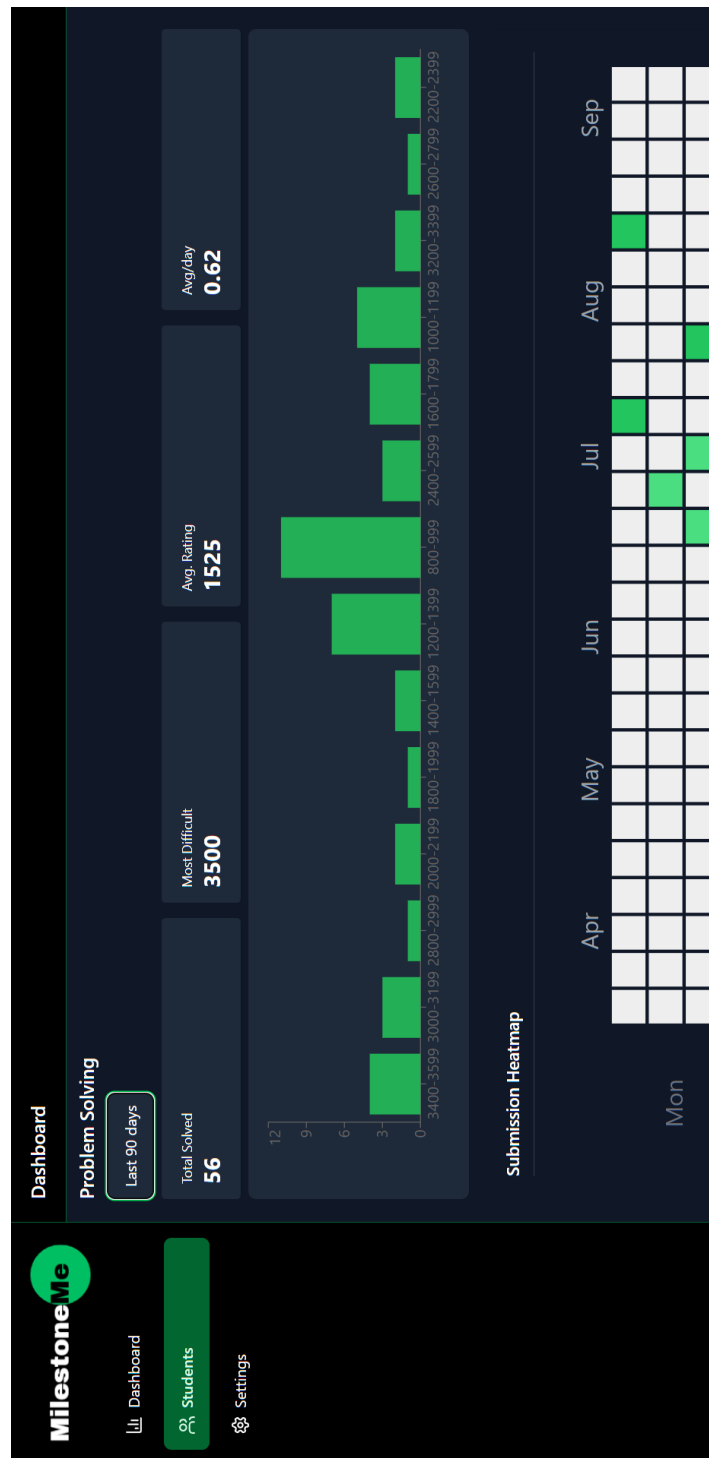


Figure 6.4: Backend Folder Structure

6.4 Settings Page

6.4.1 Details

The **Settings** page allows faculty to configure system behavior:

- Syncing Time Configuration - Faculty can set the automatic data fetching time.

6.4.2 Settings

The screenshot displays the 'Settings' page of the MilestoneMe application. The interface has a dark theme. At the top left, the 'MilestoneMe' logo is visible, followed by a navigation menu with 'Dashboard', 'Students', and 'Settings' (the latter being highlighted with a green background). The main content area is titled 'Dashboard' and 'Sync Configuration'. It contains three input fields: 'Frequency' with a dropdown menu set to 'Daily', 'Time (HH:MM)' with a text input set to '18:00', and 'Timezone' with a dropdown menu set to 'Asia/Kolkata'. A green 'Save Settings' button is located at the bottom right of the configuration area.

MilestoneMe

Dashboard Students Settings

Dashboard

Sync Configuration

Frequency: Daily

Time (HH:MM): 18:00

Timezone: Asia/Kolkata

Save Settings

Figure 6.5: Backend Folder Structure

6.5 Special Features Screenshots

6.5.1 Automated/Manual Email Screenshot

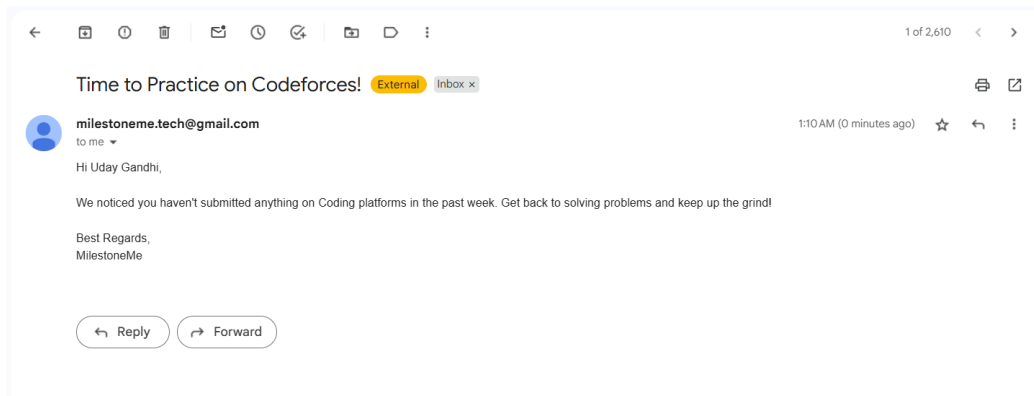


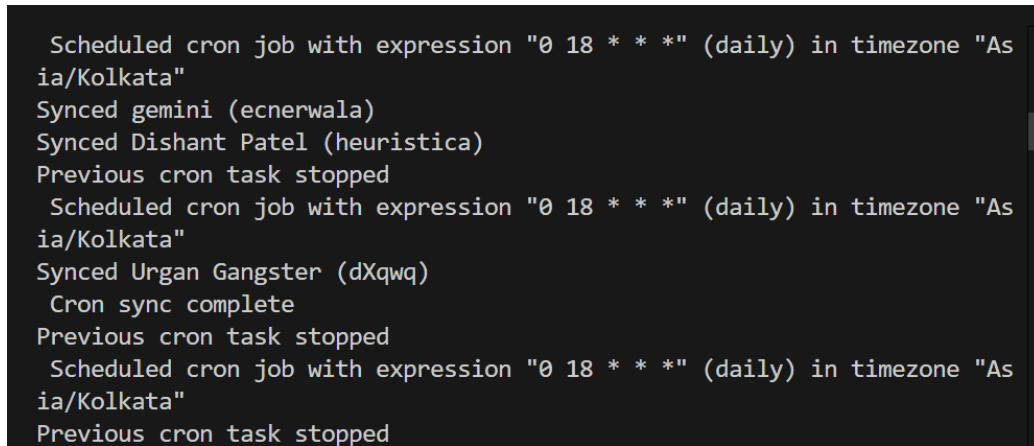
Figure 6.6: Screenshot of Automated/Manual Email Sent to Student

6.5.2 Database Entry Screenshot

```
_id: ObjectId('68bf3106b8de1cd8fa7c8e56')  
name : "name surname"  
email : "example@gmail.com"  
phone : "1234567890"  
codeforcesHandle : "turmax"  
remindersSent : 0  
emailRemindersEnabled : true  
createdAt : 2025-09-08T19:39:50.058+00:00  
updatedAt : 2025-09-08T19:39:50.058+00:00  
__v : 0
```

Figure 6.7: Screenshot of Student Data Entry in MongoDB

6.5.3 Terminal Sync Process Screenshot

A terminal window with a dark background and light-colored text. The text shows a sequence of cron job scheduling and data syncing operations. It includes messages about scheduling cron jobs with specific expressions and timezones, syncing data for users like 'gemini', 'Dishant Patel', and 'Urgan Gangster', and indicating when previous tasks stopped and the sync process was complete.

```
Scheduled cron job with expression "0 18 * * *" (daily) in timezone "Asia/Kolkata"
Synced gemini (ecnerwala)
Synced Dishant Patel (heuristica)
Previous cron task stopped
Scheduled cron job with expression "0 18 * * *" (daily) in timezone "Asia/Kolkata"
Synced Urgan Gangster (dXqwq)
Cron sync complete
Previous cron task stopped
Scheduled cron job with expression "0 18 * * *" (daily) in timezone "Asia/Kolkata"
Previous cron task stopped
```

Figure 6.8: Screenshot Showing Data Syncing in Terminal

6.5.4 CSV Export Screenshot

	A	B	C	D	E	F
1	name	email	codeforcesHandle	lastSynced	remindersSent	
2	Uday Gandhi	example@gmail.com	Benq	2025-09-08T12:36:29.714Z	12	
3	Vraj Rathva	example@gmail.com	LJC00118	2025-09-08T12:38:51.961Z	8	
4	Nevil Modi	example@gmail.com	ksun48	2025-09-08T12:44:32.820Z	2	
5	Yash Chauhan	example@gmail.com	orzdevinwang	2025-09-08T12:48:09.845Z	6	
6	Mohini	example@gmail.com	jiangly	2025-09-08T12:59:50.326Z	1	
7	gemini	example@gmail.com	ecnerwala	2025-09-08T13:05:41.083Z	1	
8	Dishant Patel	example@gmail.com	heuristica	2025-09-08T13:07:37.821Z	2	
9	Urgan Gangster	example@gmail.com	dXqwq	2025-09-08T13:11:29.969Z	1	
10	name surname	example@gmail.com	turmax		0	
11						
12						
13						

Figure 6.9: Screenshot of CSV File Download by Faculty

Chapter 7

Conclusion

The development and implementation of the **MilestoneMe** project marks a significant step forward in creating a structured, efficient, and automated system for managing students' coding activities and performance analytics. Throughout the course of this project, a range of objectives were achieved, each contributing to the overall goal of designing a system that provides meaningful insights, simplifies administrative tasks for faculty, and encourages consistent engagement among students.

7.1 Project Achievements

One of the primary achievements of MilestoneMe is the creation of a robust and modular architecture that separates concerns clearly between the frontend and backend. The frontend was designed with a focus on user experience, ensuring that faculty members can intuitively navigate through student data, performance metrics, and system settings. By breaking the frontend into distinct sections—Dashboard, Students, Student Profile, and Settings—faculty can easily monitor overall trends, individual student performance, and system configurations without confusion. Each section was developed phase-wise, allowing rigorous testing and iterative improvements at every step, which ensured stability and reliability.

On the backend, the project successfully integrates multiple functionalities, including database management, API endpoints, scheduled data fetching, and automated email notifications. The backend structure is designed to be both scalable and maintainable, with separate modules for student data,

contest tracking, problem statistics, cron configurations, and mail services. By using MongoDB as the database, the system efficiently handles hierarchical and dynamic data related to student activities, contests, and problem-solving statistics. Each student's data is uniquely identifiable, allowing precise tracking and aggregation of performance metrics. The implementation of cron jobs and automated notifications adds a dynamic element to the project, keeping students informed and engaged while minimizing manual intervention by faculty.

Another significant achievement is the integration of data visualization in the frontend. Graphical representation of data, including line graphs, bar charts, heatmaps, and performance cards, allows faculty to quickly comprehend large volumes of information and make informed decisions. The project also enables CSV export functionality, providing offline access to critical data and simplifying reporting processes. Such visual tools not only enhance usability but also facilitate better understanding and analysis of students' coding behavior over time.

7.2 Technical Learning and Innovations

Throughout the project, multiple technical skills and knowledge areas were reinforced and expanded. The use of **React.js** for the frontend, combined with Tailwind CSS for responsive and modular design, ensured that the interface is modern, intuitive, and visually appealing. Libraries like Recharts provided interactive graphs that could dynamically reflect changes in the database, enabling real-time insights into student performance. On the backend, technologies such as **Node.js**, **Express.js**, and **Mongoose** allowed the creation of a scalable, RESTful API structure capable of handling complex queries, asynchronous tasks, and scheduled operations effectively. The integration of external APIs, such as Codeforces, provided real-time data, reinforcing the importance of API management and data synchronization in full-stack development.

One of the more innovative aspects of the project is the combination of automated and manual email notifications. By analyzing students' activity and detecting periods of inactivity, the system can automatically send reminders to encourage consistent participation. Additionally, faculty can intervene manually to send customized messages when needed, allowing for both automation and personal oversight. This feature demonstrates a thoughtful

balance between technological automation and human judgment, reflecting an understanding of real-world academic management challenges.

7.3 Challenges and Solutions

Like any complex software project, MilestoneMe faced multiple challenges during development. One of the primary challenges was designing a database structure that could handle multiple types of dynamic data while maintaining efficiency and avoiding redundancy. This was addressed by using a modular approach, with separate collections for students, contests, problem statistics, and cron configurations, all linked through ObjectIds to maintain relational integrity without compromising MongoDB's document-oriented strengths.

Another challenge involved integrating real-time performance tracking with visual analytics. Fetching data from external platforms like Codeforces, processing it, and displaying it in an intuitive manner required careful planning, cron job scheduling, and synchronization of data across multiple modules. This was overcome through rigorous testing, incremental development, and the use of libraries such as Axios for API calls and Node-Cron for scheduled tasks.

Maintaining consistency between frontend displays and backend data also required attention to detail. Features such as heatmaps, line graphs, and performance cards had to accurately reflect the underlying data while being updated automatically during data syncs. Extensive debugging, validation of data at each stage, and efficient state management in React ensured that discrepancies were minimized and the system remained reliable.

7.4 Impact and Future Prospects

The MilestoneMe project has a meaningful impact on both faculty and students. For faculty, it simplifies monitoring student activity, tracking progress, and sending notifications, reducing administrative workload significantly. For students, the system encourages consistent engagement by providing clear visibility into their performance, tracking improvements, and highlighting areas that require focus. This combination of automation, visualization, and analytics makes MilestoneMe a practical tool in educational environments, particularly in programming and coding-intensive courses.

Looking ahead, there are numerous opportunities to enhance and expand MilestoneMe. Potential improvements include integration with additional competitive programming platforms, implementation of machine learning algorithms to predict student performance trends, and personalized recommendation systems for problems based on individual skill levels. Further, adding features such as gamification, leaderboard rankings, and peer comparison could increase motivation and engagement. Scaling the system to handle multiple classes, departments, or even institutions is also feasible, given the modular and scalable architecture already in place.

7.5 Project Repository

The complete source code for the **MilestoneMe** project is available at the following GitHub repository:

https://github.com/uday928/Project_MilestoneMe.git

This repository contains the full frontend and backend code, database configuration files, and instructions to run the project locally.

7.6 Conclusion

In conclusion, the MilestoneMe project demonstrates a successful implementation of a full-stack educational analytics platform. By combining a modular frontend, a robust backend, an efficient database, and real-time visualizations, the project provides a comprehensive solution for managing student performance and engagement in programming courses. The development process strengthened technical knowledge in modern web development, API integration, database management, and automated task scheduling. The system's scalable and maintainable design ensures that it can evolve over time, accommodating new features, increasing user numbers, and integrating additional educational tools.

The project serves as a solid foundation for future work in the field of educational technology, highlighting the importance of data-driven decision-making, automation, and user-centric design. It reflects a balanced approach between technology and pedagogy, creating an environment where both faculty and students benefit from structured, actionable insights. Overall, MilestoneMe is not just a technical achievement but also a meaningful contribu-

tion towards improving educational management and student engagement in coding education.

Chapter 8

Future Plans and Enhancements

The MilestoneMe project, while fully functional in its current form, holds significant potential for future expansion and enhancements. The architecture and design choices made during development ensure that the system is scalable, modular, and adaptable to new features. The following areas have been identified for potential development to further enhance the system's capabilities and impact:

8.1 Integration with Additional Competitive Programming Platforms

Currently, the system fetches data primarily from Codeforces. In the future, MilestoneMe can be expanded to integrate with other competitive programming platforms such as LeetCode, AtCoder, HackerRank, CodeChef, and others. By doing so, the system can provide a more comprehensive view of students' coding activities across multiple platforms. This integration would involve setting up API connections, synchronizing problem-solving statistics, contest participation data, and user rankings, ensuring that faculty can monitor performance holistically.

8.2 Predictive Analytics and Machine Learning Models

One of the most promising future enhancements is the implementation of machine learning algorithms to analyze student performance and predict trends. By leveraging historical data such as problem-solving patterns, contest ratings, submission frequency, and consistency, the system can identify students who may require additional guidance or are likely to achieve exceptional performance. Potential ML models include regression analysis for performance prediction, classification algorithms for categorizing students based on skill levels, and clustering techniques to group students with similar activity patterns. These predictive insights will empower faculty to take proactive measures, provide personalized guidance, and encourage consistent engagement.

8.3 Personalized Problem Recommendation System

A natural extension of predictive analytics is a recommendation engine that suggests problems to students based on their skill level, problem-solving history, and areas of improvement. Such a system can help students focus on challenges that are appropriate for their current ability, thereby maximizing learning efficiency and fostering growth. Recommendations can include difficulty-level adjustments, topic-based suggestions, and highlighting gaps in knowledge. By integrating machine learning with this recommendation system, MilestoneMe can offer personalized learning pathways tailored to each student's strengths and weaknesses.

8.4 Gamification and Leaderboards

To further enhance student engagement, gamification elements can be incorporated into the platform. Features such as leaderboards, badges, achievements, and streak tracking can motivate students to participate more actively in contests and problem-solving activities. This approach not only makes learning more interactive and enjoyable but also encourages healthy competition among students. Combined with detailed analytics, faculty can

track the impact of gamification on performance and participation rates, fine-tuning strategies to maintain motivation over time.

8.5 Enhanced Visualization and Reporting Tools

Future enhancements can also focus on providing more advanced visualizations and reporting tools. Interactive dashboards with drill-down capabilities, real-time updates, and comparative analytics between students or across multiple cohorts can offer deeper insights. Faculty can generate detailed reports for individual students, classes, or departments, supporting performance evaluation, academic planning, and intervention strategies. Exporting these reports in various formats such as PDF, Excel, or CSV can further streamline administrative processes.

8.6 Scalability and Multi-Class / Multi-Institution Support

Currently, MilestoneMe is designed for a single faculty or class. In the future, the platform can be scaled to support multiple classes, departments, or even entire institutions. Features such as multi-tenancy, role-based access control, and centralized analytics dashboards can allow institutions to manage large numbers of students while maintaining data privacy and security. Scalability enhancements would ensure that the system can handle an increased volume of data, more simultaneous users, and more complex queries without compromising performance.

8.7 Mobile Application and Cross-Platform Accessibility

Expanding MilestoneMe into a mobile application or a responsive cross-platform solution can increase accessibility and convenience for both faculty and students. Real-time notifications, mobile-friendly dashboards, and the ability to track performance on-the-go can enhance the user experience and encourage continuous engagement. Integrating push notifications for

reminders, achievements, or contest updates can ensure students remain informed and motivated even outside the desktop environment.

8.8 Artificial Intelligence for Automated Mentoring

In the longer term, the platform could include AI-driven mentorship features. By analyzing student data and performance trends, AI can provide automated suggestions, feedback, or learning resources tailored to individual students. For example, if a student consistently struggles with certain types of problems, the AI system can automatically recommend tutorials, problem sets, or practice contests to improve those skills. This will complement faculty efforts, allowing for a more personalized and adaptive learning experience.

8.9 Enhanced Security and Data Privacy Measures

As the system scales and handles more sensitive student data, implementing robust security and privacy measures becomes crucial. Future improvements can include advanced authentication mechanisms, data encryption, audit logs, and compliance with institutional or governmental data protection regulations. Ensuring a secure and trustworthy system will not only protect users' information but also foster confidence among faculty and students in using the platform regularly.

8.10 Integration with Collaborative Learning Tools

Finally, MilestoneMe can be integrated with collaborative learning platforms and tools such as discussion forums, peer review systems, and study groups. This will allow students to engage with each other, share insights, and learn collaboratively while still being monitored and guided by faculty. Combining

collaborative features with analytics can help track participation, contribution, and learning outcomes, creating a more holistic educational environment.

8.11 Conclusion on Future Plans

The future enhancements outlined above demonstrate the potential for MilestoneMe to evolve into a comprehensive, intelligent, and adaptive educational platform. By integrating multiple programming platforms, predictive analytics, personalized recommendations, gamification, and AI-driven mentoring, the system can become a powerful tool that not only tracks performance but actively facilitates learning and growth. Scalability, mobile accessibility, advanced visualizations, and security measures will further ensure that MilestoneMe remains effective, reliable, and widely usable in diverse educational contexts. These plans represent a forward-looking vision for a platform that continuously adapts to the needs of faculty and students, promotes consistent engagement, and fosters an environment where learning is efficient, data-driven, and enjoyable.