MEKELLE UNIVERSITY

ETHIOPIAN INSTITUTE OF TECHNOLOGY-MEKELLE

SCHOOL OF COMPUTING

Mekelle University Community School Authomation System

Project Submitted by

FULL NAME	ID
Haftom Abrha	Eitm/ur158030/11
Muluneh G/medhin	Eitm/ur157880/11
Haftamu Redae	Eitm/ur158047/11
Halefom Hadush	Eitm/ur129573/10
Rahel G/Hawaria	Eitm/ur156145/11

Advisor Name: Mahfuz Abdelkadir

Project Report Submitted to the School of Computing, Ethiopian Institute of Technology Mekelle, Mekelle University, in Partial Fulfillment of the Requirements for the Award of the Degree of Bachelor of Science in Software engineering.

Mekelle, Ethiopia

MAY 2016E.C

APPROVAL OF ADVISOR

This project has been submitted for examination our approval as the project advisor.

Advisor name---------- signature----------

Declaration

Declaration The Project is our own and has not been presented for a degree in any other university and all the sources of material used for the project have been duly acknowledged.

Full Name: ---------- Signature ----------

Full Name: ---------- Signature ----------

Full Name: ---------- Signature ----------

Full Name: ---------- Signature ----------

Full Name: ---------- Signature ----------

Faculty: Computing Program

Program: Bachelor of Science

Project title: Mekelle University Community School Automation System

This is to certify that I have read this project and that in my supervision and the students' performance, it is fully adequate, in scope and quality, as a project for the degree of Bachelor of Science.

Approved by: Advisor Mahfuz Abdelkadir Signature ---------- Date----------

Acknowledgment

First and foremost, we extend our deepest gratitude to Almighty God, whose blessings have made this project possible. This effort would not have been realized without supporting and contributions of many individuals. We express our sincere appreciation to our advisor, Mahfuz Abdelkadir, for His remarkable support and guidance throughout the development of this project. His expert advice and encouragement were crucial to our success. We also extend our thanks to the teachers, students, and administration of Mekelle University Community School for providing us with the necessary information and support. Their cooperation and willingness to assist were essential to the progress of Mekelle University Community School automation project. Our heartfelt thanks go to our best friends, whose unwavering support and encouragement have always been a source of strength for us. Lastly, we wish to convey our profound love and gratitude to our beloved families. Their understanding, patience, and endless love have been our pillars of support. throughout the duration of this project.

Contents pages

Abstract viii

Chapter one 1

Introduction 1

1.1 Background of the Project 1

1.2 Statement of the Problem 2

Students: - 2

Parents: - 2

Teachers: - 2

Registrar/Administrators: - 2

The school (General): - 3

1.3 Objective of the Project 3

1.3.1 General Objective: - 3

1.3.2 Specific Objectives: - 3

1.4 Methodology 4

1.4.1 Data Collection: - 4

1.4.2 Technologies and Software tools Used: - 4

1.5 Scope and Limitation of the Project 5

1.6 Significance of the Project 5

1.7 Feasibility Study 6

1.7.1. Technical Feasibility 6

Technical Skills 6

Technology Availability 6

1.7.2. Economic Feasibility 6

Development Costs 6

1.7.3. Operational Feasibility 6

Stakeholder Analysis 7

Feedback Mechanism 7

Training and Education 7

1.8 Proposed Solution 7

1.9 Team Composition 8

Chapter two 9

2.1 Overview of the Current System 9

2.1.1 Description of the Current System 9
2.2 Proposed System 11
2.2.1 Functional Requirements 11
2.2.2 Non-Functional Requirements (performance, security, user Interface, usability, reliability) 12
2.3 System Modeling 14
2.3.1 Scenario 15
2.3.3 Use case Documentation 20
Chapter Three 34
System Design 34
3.1 Design Goals 34
Automation of Administrative Tasks 34
Ensuring Data Accuracy, Integrity, and Security 34
Modernizing Administrative Processes 34
Providing a User-Friendly Interface 34
3.2 Current System Architecture 35
Manual Administrative Processes 35
Limitations 35
Paper-Based Documentation 35
Traditional Communication Methods 35
Inconsistent Grading Practices 36
3.3 Proposed System Architecture 36
Frontend Interface 36
Backend Services 36
Database 37
API Integration 37
Security Measures 37
3.4 Subsystem Decomposition 37
Admin Dashboard 38
Teacher Dashboard 38
Parent Dashboard 38
Student Dashboard 38
3.5 Component Diagram 39

3.6 Deployment Modeling 40
Description for deployment modeling 40
Client App (Web Server) 40
Application Server 40
3.7 Persistent Data Management 41
3.8 Access Control and Security 42
Authentication with JWT 42
Authorization with JWT 43
Token Expiration 43
3.9 User Interface Design 43

List of Figures

Figure 1: current system architecture 10
Figure 2: use case diagram student and parent 17
Figure 3: Use case diagram for Teacher 18
Figure 4: Use case diagram for registrar and admin 19
Figure 5: sequence diagram for login 27
Figure 6: sequence diagram for adding mark 28
Figure 7: sequence diagram for viewing result 29
Figure 8: sequence diagram for update in admin 30
Figure 9: sequence diagram for view result for parent 31
Figure 10: activity diagram for student registration 32
Figure 11 :class diagram MUCS 33
Figure 12 : component diagram MUCS 39
Figure 13: deployment diagram for MUCS 40
Figure 14: data persistence for MUCS 42
Figure 15: login user interface 43
Figure 16: admin user interface for registration 44
Figure 17: Registrar user interface student registration 45
Figure 18: Teacher user interface design 46
Figure 19: Student user interface design 47
Figure 20 : Parent user interface 48

List of Tables

Table 1: list of group members and their role. 8
Table 2 : use case Student Registration 20
Table 3 : use case Login 21
Table 4: use case result viewing 22

Table 5: use case update 23
Table 6: use case search 24
Table 7: use case Add mark 25
Table 8: use case assign course and section 26

Abstract

This project aims to develop an automated school management system for Mekelle University Community School to replace the manual processes and enhance operational efficiency of the school. The current system's is based on paper-based documentation and traditional communication method that leads to errors, time consuming, and inconsistencies and tiredness's. The proposed system automates administrative tasks such as student registration, grade management, and stakeholder communication, ensuring data accuracy, integrity, and security. Utilizing React.js for the frontend, Spring Boot for the backend, and PostgreSQL for the database, the system features contain dashboards for administrators, registrars, teachers, students, and parents. Key functionalities include efficient data entry, grade management, communication tools, and real-time access to academic information. The system design emphasizes user-friendliness, robust security through role-based access control and JWT authentication, and reliable data management. By modernizing administrative processes and enhancing communication, the system aims to improve operational efficiency and user satisfaction at Mekelle University Community School.

Chapter one

Introduction

1.1 Background of the Project

Mekelle University Community School, situated in Aynalem, Mekelle, was established in 2003 E.C. under the auspices of Mekelle University. Initially, the school encompassed grades 1 through 4. Subsequently, it expanded its scope, incorporating grade 5 in 2004, grade 6 in 2005, grade 7 in 2007, and grade 8 in 2008, thus functioning solely as a primary school from 2003 to 2008.

Transitioning in 2009, the institution extended its educational offerings to include grade 9, followed by the addition of grade 10 in 2010, grade 11 in 2011, and grade 12 in 2012. This evolution marked its transformation into a secondary school. Presently, the school comprises distinct primary and secondary divisions, yet both bear the appellation Mekelle University Community School. The school's resources and facilities are generously supported by Mekelle University, endowing it with superior educational materials, technological infrastructure, and well-equipped laboratories, thereby enhancing the learning experience.

Notably, the offspring of Mekelle University's faculty members are enrolled in this institution, alongside students from the local community of Aynalem. While the primary school operates under its own administration, the secondary school, established in 2009 with the admission of grade 9 students exclusively, exhibits a substantial enrollment rate exceeding 300 students annually. The cumulative student population surpasses 800 individuals per annum, managed by a faculty comprising 36 educators and three administrative staff members, including two administrators, four finance personnel, and two typists.

At present, the secondary school accommodates over 700 students pursuing natural sciences and more than 157 students specializing in social sciences, totaling 857 individuals. Despite its academic excellence, Mekelle University Community School relies on manual processes for student registration, grade management, and other essential administrative tasks. However, these conventional methods prove to be inefficient and time-consuming.

In response to these challenges, we want to implement a comprehensive school management system to streamline operations and enhance efficiency in student registration, grade management, and overall data management of the school.

1.2 Statement of the Problem

The manual administrative processes at Mekelle University Community School are inefficient, prone to errors, and time-consuming. This leads to delays in important tasks and hampers effective decision-making. The students, teachers, administrators, registrars, and parents are facing the manual based system at Mekelle University Community School.

Students: - Students encounter challenges in accessing their assessment results efficiently. They cannot view their result as soon as they have taken their result. In order to view their assessment results, necessitates a physical presentation to their respective teachers at the end of the semester. Alternatively, they are experiencing delays in accessing their academic records or encounter difficulties in communicating with teachers regarding their academic progress. Manual systems may introduce errors in the computation and recording of grades.

Parents: - Parents similarly encounter obstacles in accessing their children's academic results. The reliance on manual systems means that accessing their child's grades is not readily available, leading to frustration and uncertainty regarding their child's academic progress. This lack of accessibility may hinder parental involvement in their child's education and the ability to provide necessary support and guidance.

Teachers: - Teachers are also challenged with many problems, including manual recording and inputting of grades, as well as computing and generating reports. This workload may obstacle their ability to access student information efficiently or communicate effectively with both students and parents regarding academic matters. Moreover, manual process has many problems for teachers in terms of time consuming, tiredness, data inaccuracy, and their time period.

Registrar/Administrators: - Registrars and administrators grapple with inefficiencies in various aspects of school management, including the organization of student records, class scheduling, and resource allocation. Additionally, they face obstacles when attempting to produce comprehensive reports or analyze data to inform decision-making and strategic planning processes. The utilization of manual systems often results in issues such as data redundancy,

errors, and security vulnerabilities, jeopardizing the integrity and confidentiality of student information.

The school (General): - Mekelle University Community School faces problems because it still uses old-fashioned ways to do administrative work. These ways make things slower, less effective, and less clear. Using paper and manual methods means doing the same tasks over and over, making mistakes in the data, and taking a long time to make decisions. This affects how well the school performs and how people see it. Data lost is one bottleneck in the school. Since the data system is manual based, duty to different problems such as wars and other emergency an overall data of the school may lost. Accessing previous data of the school is difficult, tiredness and it takes more time.

1.3 Objective of the Project

1.3.1 General Objective: -

To develop and implement an automated school management system for Mekelle University Community School.

1.3.2 Specific Objectives: -

The specific objectives of the system are to:

- Enable teachers to easily inputting and managing grades including academic performance and to enhance student monitoring.
- Allow students to view their assessment records and academic performance in real-time, promoting accountability and engagement.
- Centralize administrative tasks such as student registration, scheduling, and resource allocation to enhance efficiency.
- Implement role-based access control to ensure data security and privacy compliance at the administrative level.
- Enable parents to access their child's academic progress, including grades
- Provide timely notifications regarding school events, announcements, and important dates to keep parents engaged and involved in their child's education.
- Modernize administrative processes to improve overall efficiency and effectiveness in school management.

- Ensure data accuracy, integrity, and security to maintain trust and compliance with regulatory standards.

1.4 Methodology

1.4.1 Data Collection: -

Data will be collected through interviews with stakeholders to understand their requirements and pain points. Observation of existing processes and analysis of relevant documents will also inform system design and development.so that we'll chat with people involved in the school to learn about what they need and what problems they face. We'll also watch how things are currently done and look at any documents that might help us understand better. This will help us design and build a system that meets everyone's needs and solves the existing problems.

1.4.2 Technologies and Software tools Used: -

Development tools and Technologies used

Frontend: VSCODE, Reactjs
*Backend***:** INTELJI, Spring boot
Database: PostgreSQL for data storage.
Version Control: Git for collaborative development and version control.
Testing Tools: POSTMAN

Design tools

EDRAWMAX: Utilized to develop diagrams illustrating the architecture of the system, user interface design, and data flow representation.

PostgresSQL: Employed to design the database schema and structure, ensuring efficient data storage and retrieval.

Microsoft Word processing: Utilized to document system requirements, design specifications, and development processes for reference and future maintenance.

1.5 Scope and Limitation of the Project

Scope: -

The project aims to develop a comprehensive school management system for the Mekelle University community school. This system will cover student registration, grade management, and overall data management within the school. It will feature web-based interfaces for easy accessibility. The system will include five dashboards: teacher, student, registrar, administrator, and parent.

Teachers: will be able to submit students' marks using their IDs.

Students: will be able to view their results on their personal dashboards.

Registrars: will be able to register new students based on their admission cards.

Parents: will be able to view their children's results using their children's IDs.

Administrators: will be able to register registrars and teachers, and assign courses and classes.

Limitations: -

The project will not address non-administrative functions such as curriculum development or teaching methodologies.

Limited resources may restrict the scope of the project, potentially leading to prioritization of certain features over others.

1.6 Significance of the Project

The implementation of the school management system at Mekelle University Community School promises several benefits, including streamlined administrative processes, improved communication among stakeholders, and a reduction in manual workload and errors. Moreover, the systems will improve an overall the manual based system mainly grading system, registration system.

1.7 Feasibility Study

1.7.1. Technical Feasibility

Technical Skills

The project requires expertise in web development, database management, and potentially system integration. Given the presence of Software Engineering students undertaking the project, and the availability of faculty guidance, the technical skills required seem achievable.

Technology Availability

We can leverage open-source technologies and readily available development tools to create the automation system, ensuring accessibility and cost-effectiveness. This approach allows us to minimize development costs while still utilizing robust and reliable technologies for the project.

1.7.2. Economic Feasibility

Development Costs

The project can utilize open-source technologies to minimize development costs. Server hosting and maintenance costs will need to be factored in, but these are likely to be offset by the efficiency gains and paper reduction achieved.

1.7.3. Operational Feasibility

User Adoption

Ensuring successful user adoption of the new school management system at Mekelle University Community School requires careful consideration of several factors. This includes assessing user readiness, addressing training requirements and user experience, and identifying strategies to promote acceptance and minimize resistance. By prioritizing these aspects, the school can facilitate the smooth implementation and utilization of the system, leading to improved efficiency and effectiveness in administrative processes.

Change Management

A successful implementation of the Mekelle University Community School Automation project requires effective change management strategies. Change management focuses on ensuring

that stakeholders within the school community are ready and willing to adopt the new system and processes. To achieve this, we will do some activities.

Stakeholder Analysis

Identify all stakeholders involved, including administrators, teachers, students, and parents. Understand their needs, concerns, and expectations regarding this automation project.

Feedback Mechanism

Establish a feedback mechanism to gather input from stakeholders throughout the implementation process. Encourage open communication and address concerns promptly to build trust and engagement.

Training and Education

Provide adequate training and education to stakeholders on how to use the new system effectively. Offer workshops, and support materials to facilitate learning and adoption.

1.8 Proposed Solution

The proposed solution aims to address the challenges faced by Mekelle University Community School by providing a comprehensive school management system. Building upon previous initiatives and leveraging insights from stakeholder analysis, the proposed solution offers a user-friendly interface for student registration, and grade management, by integrating frontend and backend components, the system ensures seamless communication and data exchange, reduce errors, avoid delay of time waiting when students view their result, avoid the tiredness and data inaccuracy for teachers. The system helps for teachers in terms of grade computation, easily viewing their time period, while implementing security measures to protect sensitive data and ensure compliance with privacy regulation.

1.9 Team Composition

This project is being undertaken by a team of five students. the project team will comprise individuals with expertise in frontend development, backend development, database management, and project management. Roles within the team will include frontend developers, backend developers. Each team member will contribute their skills and knowledge to ensure the successful execution of the project.

Name	Role
Haftom abrha	coder
Muluneh G/medhin	coder
Haftamu redae	analyzer
Halefom hadish	tester
Rahel G/Hawaria	Documentation

Table 1: list of group members and their role.

Chapter two

Requirement Analysis and System Modeling

2.1 Overview of the Current System

2.1.1 Description of the Current System

The current system at Mekelle University Community School relies on manual administrative processes for tasks such as student registration, attendance tracking, and grade management. These processes are time-consuming, error-prone, and inefficient, leading to delays and inaccuracies in data management.

Student Registration

Student registration involves manual paperwork where administrators collect students' admission card. The admission card contains personal information about the student, including their name, date of birth, address, and previous academic records. Administrators manually maintain student records in physical files or spreadsheets, making it challenging to efficiently manage and update information.

Grade Management

Teachers record students' grades and assessments manually using paper-based gradebooks or spreadsheets. Grading criteria and assessments may vary across teachers and subjects, leading to inconsistencies in grading practices. Calculating final grades and generating reports often involves manual computation and data entry, increasing the likelihood of errors.

Communication and Information Sharing

Communication between school administrators, teachers, students, and parents relies heavily on traditional methods such as phone calls, emails, and physical notices. Information sharing regarding school events, announcements, and academic updates may be disseminated through printed materials or word-of-mouth communication.

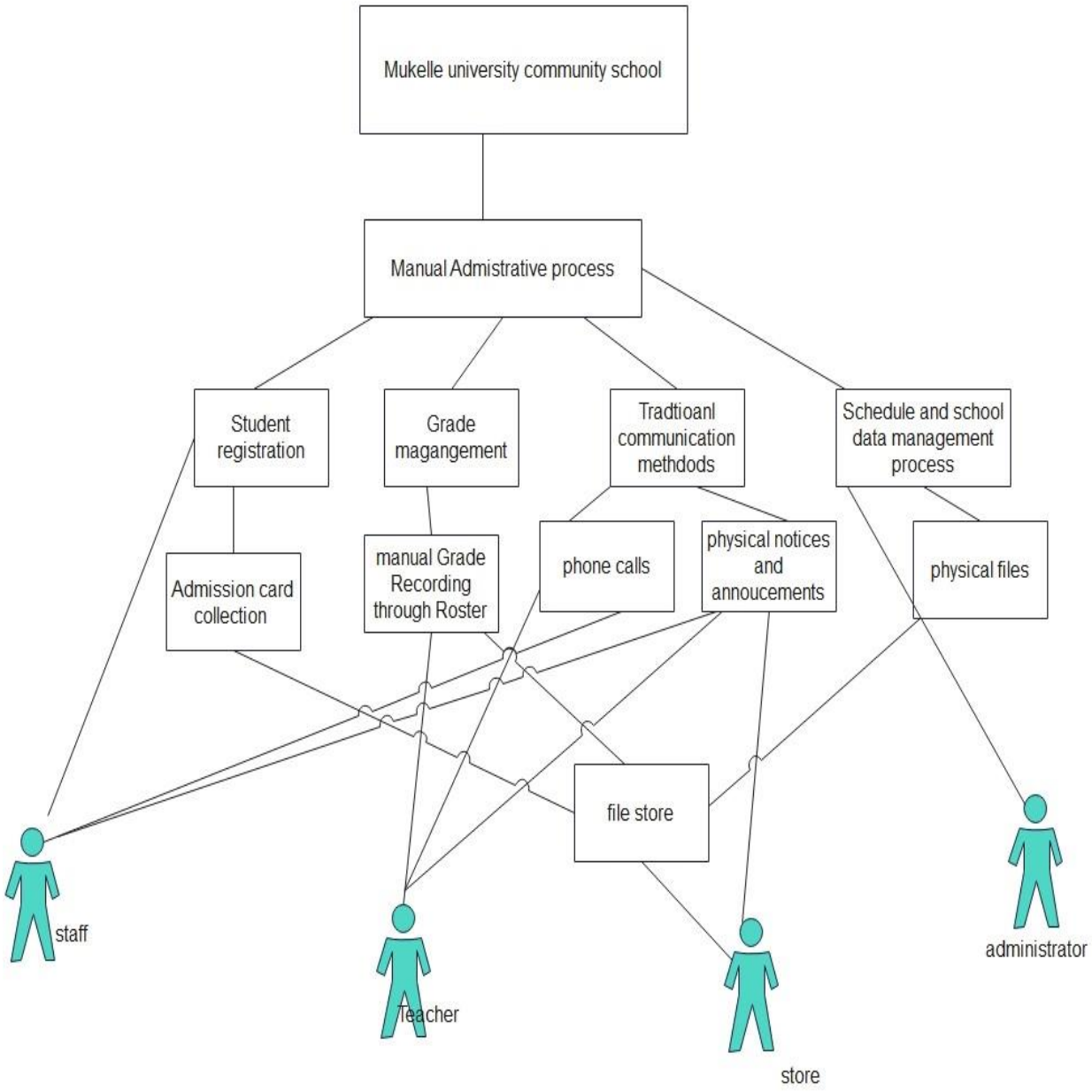


Figure 1: current system architecture

2.2 Proposed System

2.2.1 Functional Requirements

The proposed school management system aims to automate administrative tasks and improve operational efficiency at Mekelle University Community School. Here are the functional requirements of the proposed system.

Admin Dashboard Enhancements

Subject and Section Assignment

Allow administrators to assign subjects and sections to teachers and students.

Teacher Registration

Enable administrators to register new teachers by collecting their personal information, qualifications, and contact details. Include functionalities for verifying credentials and conducting background checks.

Exam Scheduling

Will implement a feature for scheduling exams, quizzes, and assessments. Administrators should be able to set exam dates, times, locations, as needed.

Registrar Dashboard Features

Admission Card Registration

Facilitate the registration process for new students based on the information provided in their admission cards. The registrar dashboard should streamline the data entry process and ensure accurate and timely registration.

Teacher Dashboard Enhancements

Assessment Result Entry

Allow teachers to input and manage students' assessment results directly into the system. Provide functionalities for entering grades, comments, and feedback for individual assignments, tests, and projects.

Progress Tracking

Enable teachers to track students' academic progress over time by accessing historical assessment data and generating performance reports.

Student Dashboard Features

Mobile Access

Develop a mobile-friendly dashboard that allows students to access their academic information, including grades, and upcoming assignments, from their smartphones or tablets.

Result Viewing

Provide students with the ability to view their assessment results, grades, and overall academic performance in real-time. Include features for filtering and sorting results by subject, term, or category.

2.2.2 Non-Functional Requirements (performance, security, user Interface, usability, reliability)

Performance

Response Time

The system should respond to user requests within acceptable time limits. For example, loading pages, retrieving data, and processing user inputs should occur quickly to enhance user experience.

Scalability

The system should be able to handle increasing numbers of users, data, and transactions without significant degradation in performance. It should scale horizontally and vertically as necessary to accommodate growth.

Security

Authentication and Authorization

Technologies and Tools:

JWT (JSON Web Tokens): Used for securely transmitting information between JSON objects. It is used for both authentication and authorization.

Role-Based Access Control (RBAC): it is used to manage user permissions based on roles.

Techniques:

JWT: Upon successful login, a JWT is issued to the user, which includes encoded user information and permissions. This token is then used for subsequent requests to verify the user's identity and access rights.

RBAC: Define roles (e.g., admin, student, teacher, parent and registrar) and assign permissions to these roles.

User Interface

Intuitive Design

Technologies and Tools:

ReactJS is a JavaScript library that we use for building user interfaces. React allows for the creation of reusable UI components, making development more efficient and the UI more consistent.

Bootstrap is a popular CSS framework that provides pre-designed components and a responsive grid system, making it easier to create visually appealing and responsive layouts.

React-Bootstrap is a library that integrates Bootstrap with React, allowing you to use Bootstrap components as React components.

Techniques:

Component-Based Architecture: - In React, the user interface will be broken down into small, reusable components. Each component will manage its own state and lifecycle, making the application more modular and easier to maintain.

Responsive Design: - Using Bootstrap's grid system and responsive utility classes, the UI will adapt to different screen sizes and devices, providing a consistent experience across desktops, tablets, and mobile phones.

Consistent Styling: Bootstrap's pre-designed components (such as buttons, forms, and navigation bars) will ensure a consistent look and feel throughout the application. Custom styles will be applied sparingly to maintain consistency.

Clear Navigation and Labels: Navigation elements will be clearly labeled and logically organized, making it easy for users to find what they need. Interactive elements like buttons and links will have descriptive labels to indicate their purpose.

Feedback and Interactivity: Using React, the UI will provide immediate feedback for user actions. For example, form validations will display error messages in real-time, and interactive elements will respond to user input without page reloads.

Usability

Technologies and Tools:

ReactJS: For building dynamic, interactive, and reusable UI components.

Bootstrap: For providing a consistent and responsive design framework.

React Testing Library: For testing the usability and accessibility of your components.

User Feedback Tools: Tools like Hotjar or Google Analytics for gathering user behavior data and feedback.

Techniques:

Confirmation Dialogs: Use confirmation dialogs for critical actions (e.g., deleting data) to prevent accidental actions.

Screen Reader Support: Ensure compatibility with screen readers by using semantic HTML and ARIA attributes. This helps users with visual impairments navigate and use the application effectively.

Reliability

The system should be reliable and available to users at all times, ensuring uninterrupted access to the system.

Requirements:

The system should have a high uptime and minimal downtime, with mechanisms in place to handle system failures and recover from them.

The system should have a robust backup and disaster recovery plan to protect against data loss and ensure the integrity of the stored information

2.3 System Modeling

2.3.1 Scenario

Scenario Name: view result

Name of User: Mesele, student

Goal: To access and review assessment results for individual students or classes, facilitating monitoring of academic progress.

Steps:

1. Mesele must login to the system using his password and username.
2. The system displays the student dashboard.
3. Mesele must click the view result button.
4. The system displays the assessment result of Mesele.
5. Mesele can see his assessment result.

Scenario Name: New Student Registration

Name of User: Grimay as registrar, Mesele as new Student

Goal: To create a new account in the system, providing necessary personal information and credentials for accessing the platform.

Steps:

1. Mesele must submit his admission card to Grimay.
2. Grimay must read and check the admission card details.
3. Grimay must login to the system using his credentials.
4. Fill the form all necessary admission card details.
5. Check whether the form is filled correctly.
6. Click register button to register Mesele.
7. System display register notification.

Scenario Name: Add Mark

Name of User: Berhe as Teacher.

Goal: to submit students mark to system database.

Steps:

1. Berhe must login to his dashboard.
2. The system displays the Berhe's dashboard as Teacher Dashboard.
3.Berher enter the student id.
3. Fill the student mark in the input field.
4.click submit
5. system displays notification

Scenario Name: Update

Name of User: Dawit as admin

Goal: To modify role of registrar.

Steps:

1. Dawit must be login to the system.
2. The system displays Dawit dashboard as admin dashboard
3. Dawit Enter the registrar's id.
4. Select the role from the dropdown menu.
5. click the update button.
6. The system displays notification.

Scenario Name: Change Password

Name of User: Andom, student

Goal: To update and change the user's password for the system.

Steps:

1. Andom must login to the system.
2. The system displays the student dashboard.
3. Andom must click the change password button.
4. The system displays a form to fill out.
5. Andom fill the form by entering current password, new password, and the coniform password and click the update button.
6. The system updates the password.

2.3.2 Use case Diagrams

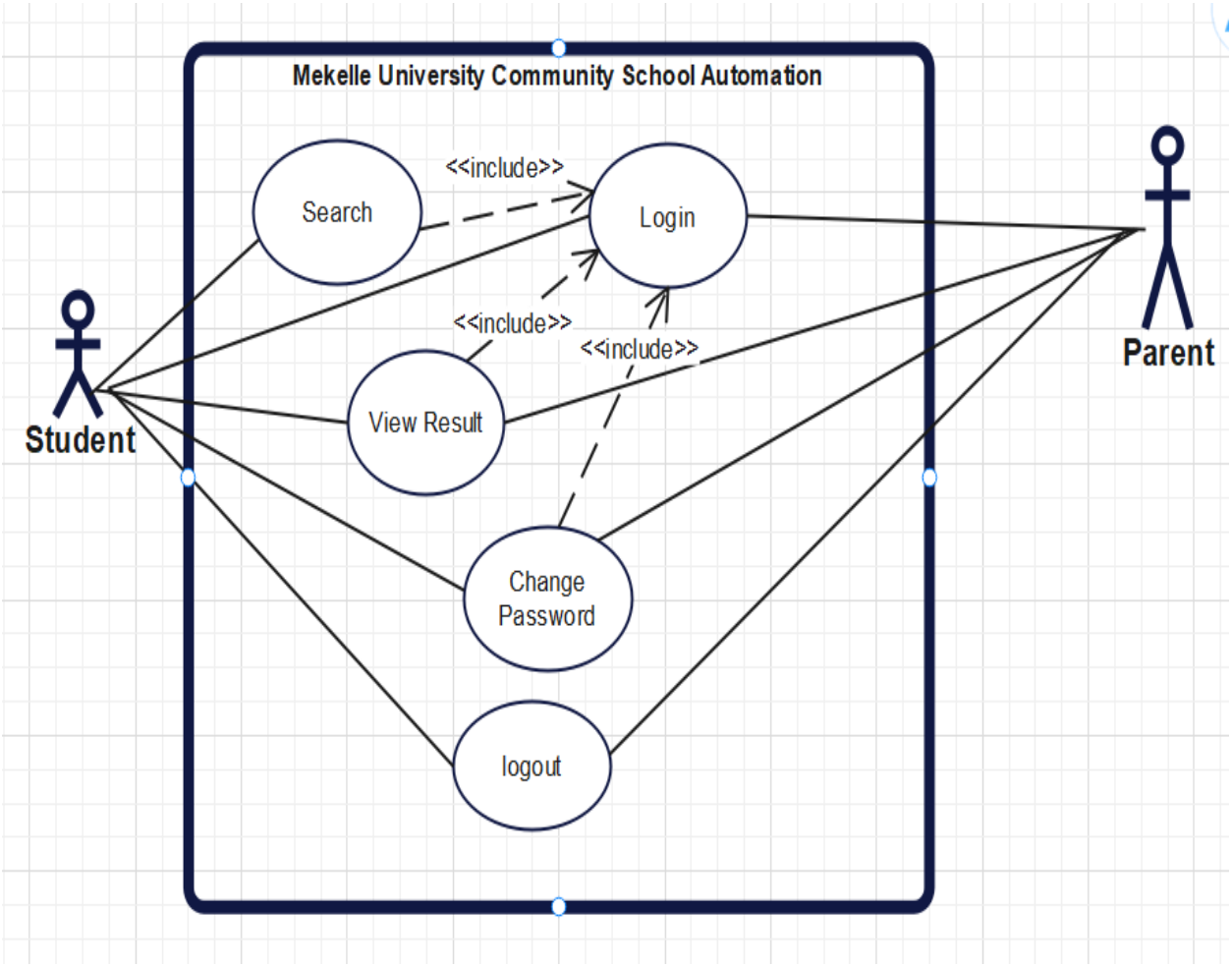


Figure 2: use case diagram student and parent

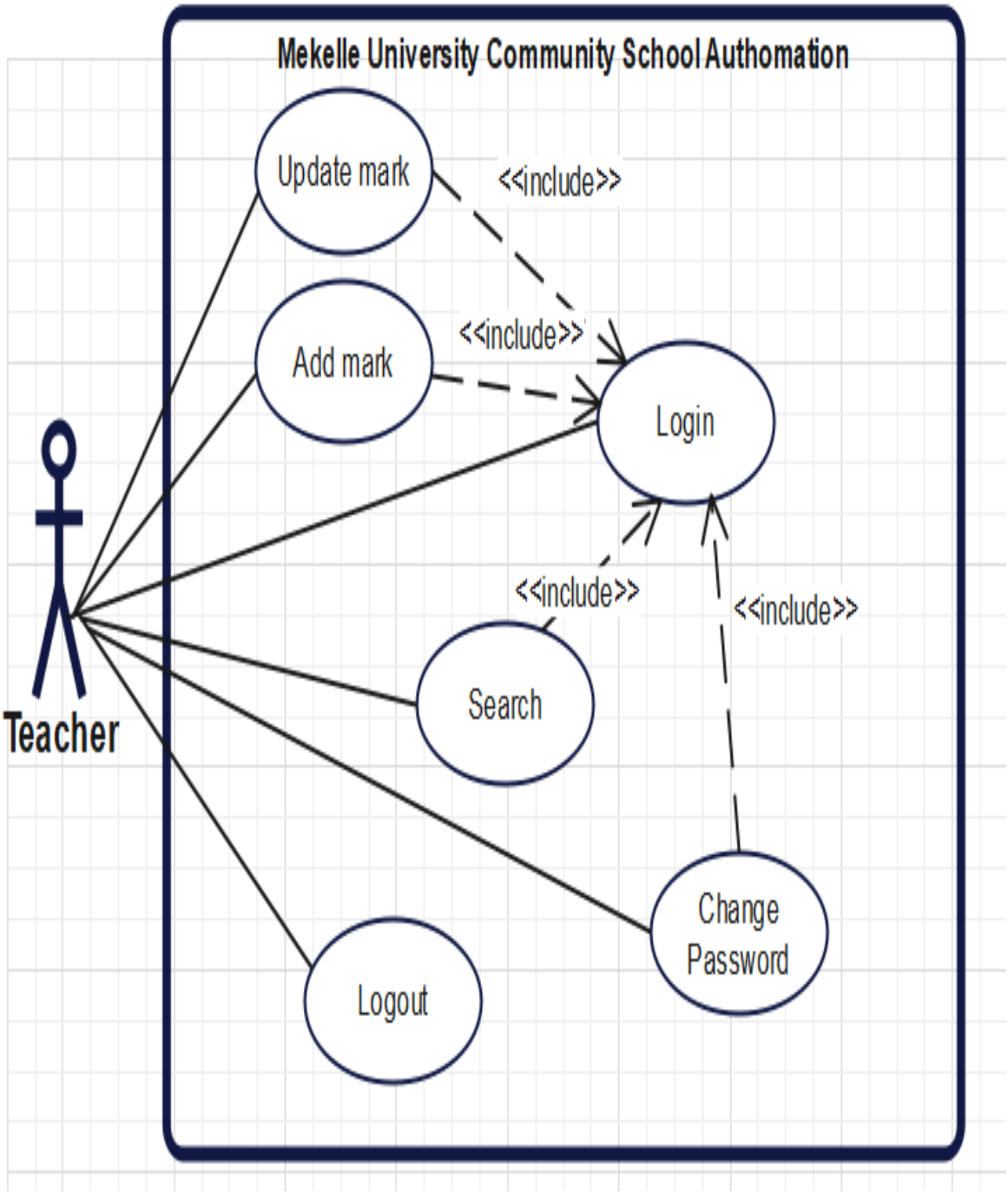


Figure 3: Use case diagram for Teacher

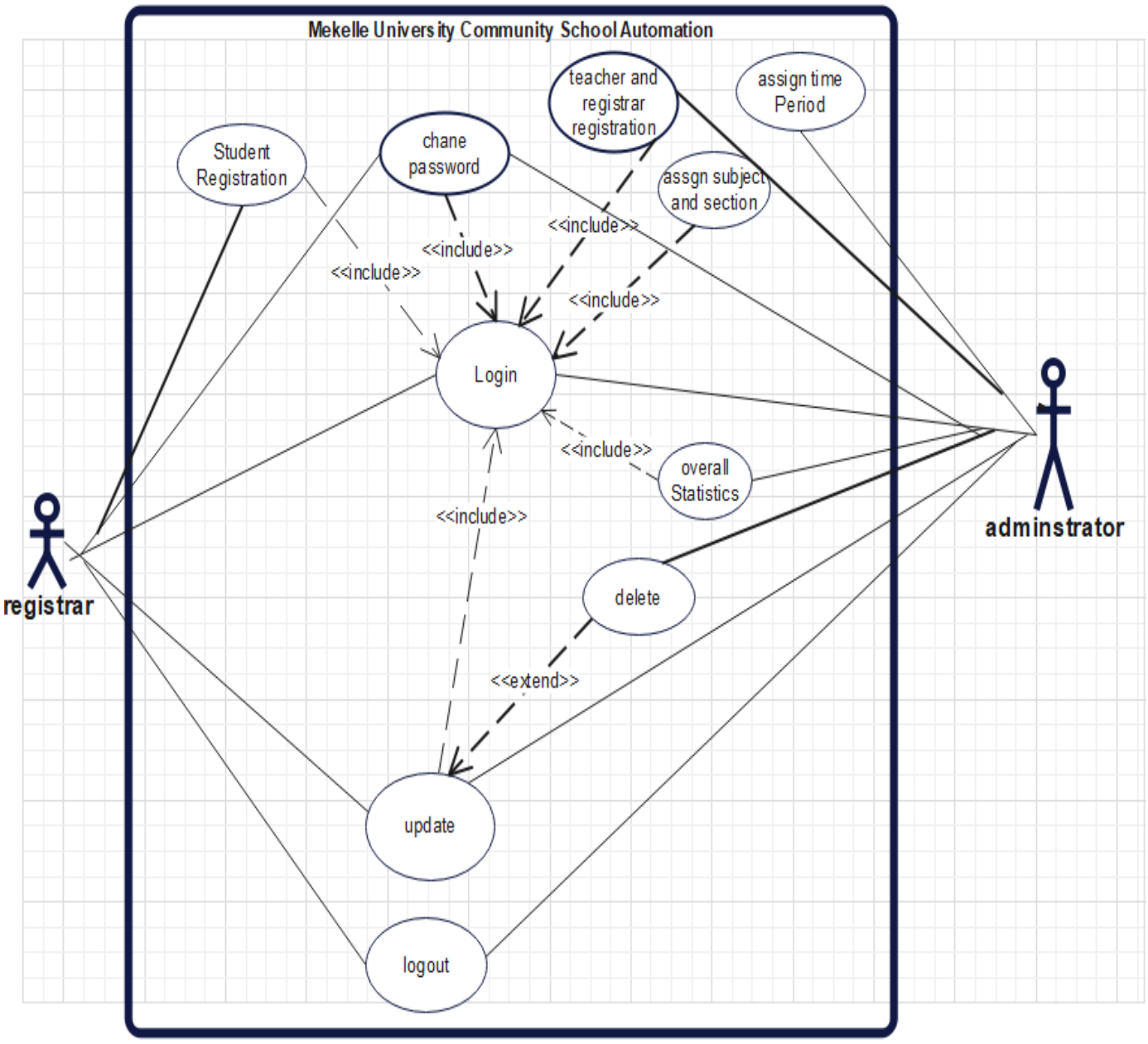


Figure 4: Use case diagram for registrar and admin

2.3.3 Use case Documentation

Use case ID	**case1**
Use Case Name	Student Registration
Actor	Registrar
Description	Registering a new student into the system.
Pre-Condition	The new student must have submitted their admission card.
Post Condition	The new student is registered in the school's database and provided with login credentials.
Basic Course of Action	
User Action	System Response
Registrar verifies the information on the admission card.	System prompts the registrar to enter the verified admission card details.
Registrar enters the verified admission card details into the system.	System confirms the validity of the submitted admission card.
Registrar enters the student's personal and academic information into the system.	System stores the entered information.
Registrar submits the information.	System generates a unique student ID and stores the information in the database.
Registrar issues the student's ID card and provides login credentials for the student dashboard.	System confirms the issuance of the ID
Alternate Course of Action	
User action	System response
Registrar submits the admission card to the system.	System verifies the submitted admission card and generates a unique student ID and stores the information in the database.
Register check success full submission of the admission card	System stores the entered information and sends the notification to registrar dashboard
Registrar issues the student's ID card and provides login credentials for the student dashboard.	System confirms the issuance of the ID

Table 2 : use case Student Registration

Use Case ID	**Case6**
Use Case Name	Login
Actor	Student, Teacher, Administrator
Description	Authenticate and access the system using valid credentials.
Pre-condition	User needs to have valid login credentials.
Post-condition	User gains access to the system.
Basic Course of Action	
User Action	System Response
User enters login credentials.	System verifies user credentials.
User successfully logs in.	System grants access to the user's dashboard.
User exits the system.	System logs the user out.
Alternate Course of Action	
User Action	System Response
User enters login credentials	System verifies user credentials
User enters incorrect credentials multiple times	System locks the user's account
User exits the system	System logs the user out

Table 3 : use case Login

Use Case ID	Case1
Use Case Name	Result Viewing
Actor	Student
Description	View assessment result from the system.
Pre-condition	Students need to access their assessment results
Post-condition	The student viewed their result.
Basic Course of Action	
User action	System response
Student enters his/her user's name and password	System verifies student credentials
Student clicks log in button	System displays student dashboard
Student clicks academic button	System retrieves the results of the student
Student exits the system	System logs the student out
Alternate Course of Action	
User action	System response
Student accesses the login page.	System verifies student credentials.
Student logs in.	System loads the assessment results page.
Student finds that the assessment result is not available.	System displays a message indicating no assessment results are available.
Student exits the system.	System logs the student out

Table 4: use case result viewing

Use Case ID	**Case2**
Use Case Name	Update
Actor	Administrator, Registrar, Teacher
Description	Modify existing information within the system.
Pre-condition	User needs to have appropriate permissions and access rights.
Post-condition	The required information in the system is updated.
Basic Course of Action	
User Action	System Response
User logs into the system	System verifies user credentials
User navigates to the section where the information needs to be updated	System loads the relevant data
User makes the necessary changes	System validates and saves the changes
User exits the system	System logs the user out
Alternate Course of Action	
User Action	System Response
User logs into the system.	System verifies user credentials.
User navigates to the section where the information needs to be updated.	System loads the relevant data.
User attempts to make changes but encounters a data validation error (e.g., invalid data entry).	System displays an error message indicating the validation issue.
User decides to cancel the update process.	System does not save any changes and retains the current data.

Table 5: use case update

Use Case ID	**Case3**
Use Case Name	Search
Actor	Student, Teacher, Administrator, Registrar , parent
Description	Search for specific information within the system.
Pre-condition	User needs to have access to the search functionality.
Post-condition	Relevant information is displayed to the user.
Basic Course of Action	
User Action	System Response
User enters search query	System retrieves matching results
User selects a result to view details	System displays the details of the selected item
User exits the search function	System returns to the previous screen
Alternate Course of Action	
User Action	System Response
User enters search query	System retrieves no matching results
User adjusts the search criteria	System prompts the user to refine the search
User exits the search function	System returns to the previous screen

Table 6: use case search

Use Case ID	**Case4**
Use Case Name	Add Mark
Actor	Teacher
Description	Input assessment marks and feedback for students.
Pre-condition	Teacher needs to have access to the assessment result entry section.
Post-condition	Assessment results are stored in the system.
Basic Course of Action	
User Action	System Response
Teacher logs into their dashboard	System authenticates teacher's credentials
Teacher selects the class and assessment type	System presents options for class and assessment type
Teacher inputs individual student's results	System stores the assessment results in the database
Teacher submits the assessment results	System confirms submission
Alternate Course of Action	
User Action	System Response
Teacher logs into their dashboard	System authenticates teacher's credentials
Teacher selects the class and assessment type	System presents options for class and assessment type
Teacher decides not to input results	System does not store any data
Teacher exits the assessment result entry section	System logs the teacher out

Table 7: use case Add mark

Use Case ID	**Case5**
Use Case Name	Assign Course and Section
Actor	Administrator
Description	Allocate subjects and sections to teachers and students.
Pre-condition	Administrator needs to have access to the subject and section assignment feature.
Post-condition	Subjects and sections are assigned to teachers and students.
Basic Course of Action	
User Action	System Response
Administrator accesses the assignment feature	System displays the assignment interface
Administrator selects subjects and sections	System presents options for subject and section selection
Administrator assigns teachers to subjects and sections	System updates the assignment information
System updates the database	System confirms the update
Alternate Course of Action	
User Action	System Response
Administrator accesses the assignment feature	System displays the assignment interface
Administrator decides not to assign any subjects or sections	System does not make any updates
Administrator exits the assignment feature	System logs the administrator out

Table 8: use case assign course and section

2.3.4 Sequence Diagrams

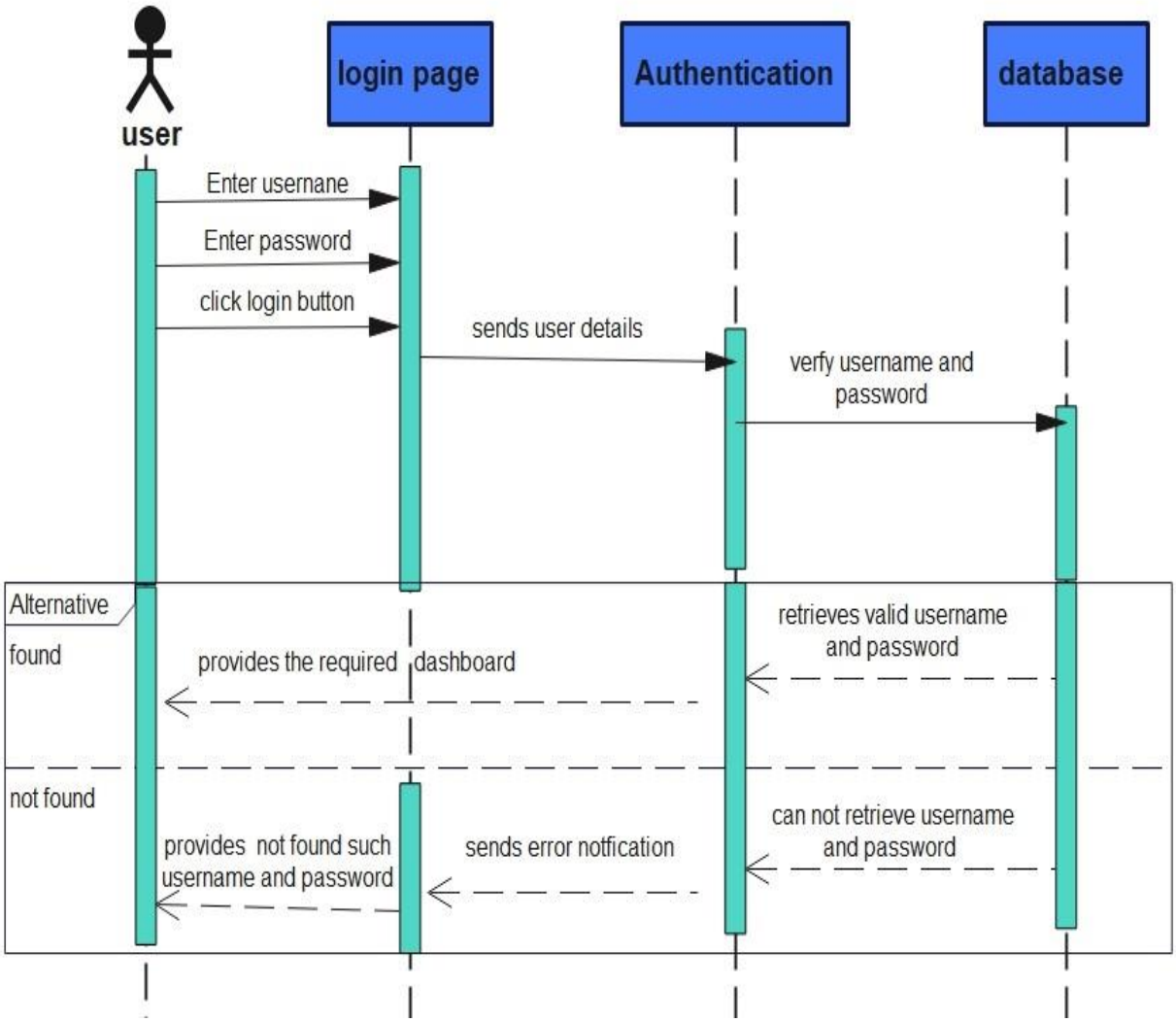


Figure 5: sequence diagram for login

Assum the teacher is successfully signed in.

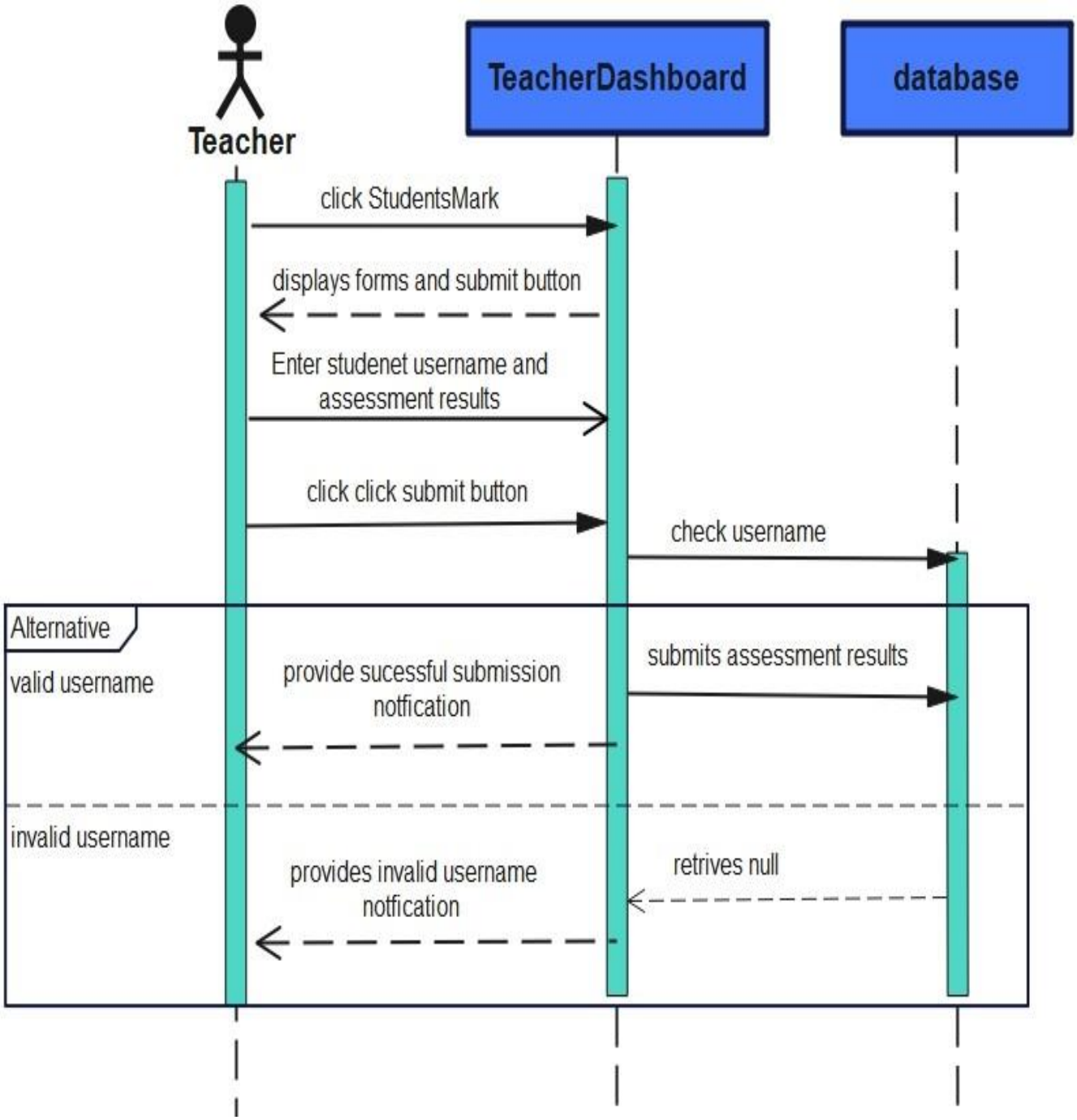


Figure 6: sequence diagram for adding mark

Assume the student is successfully signed in.

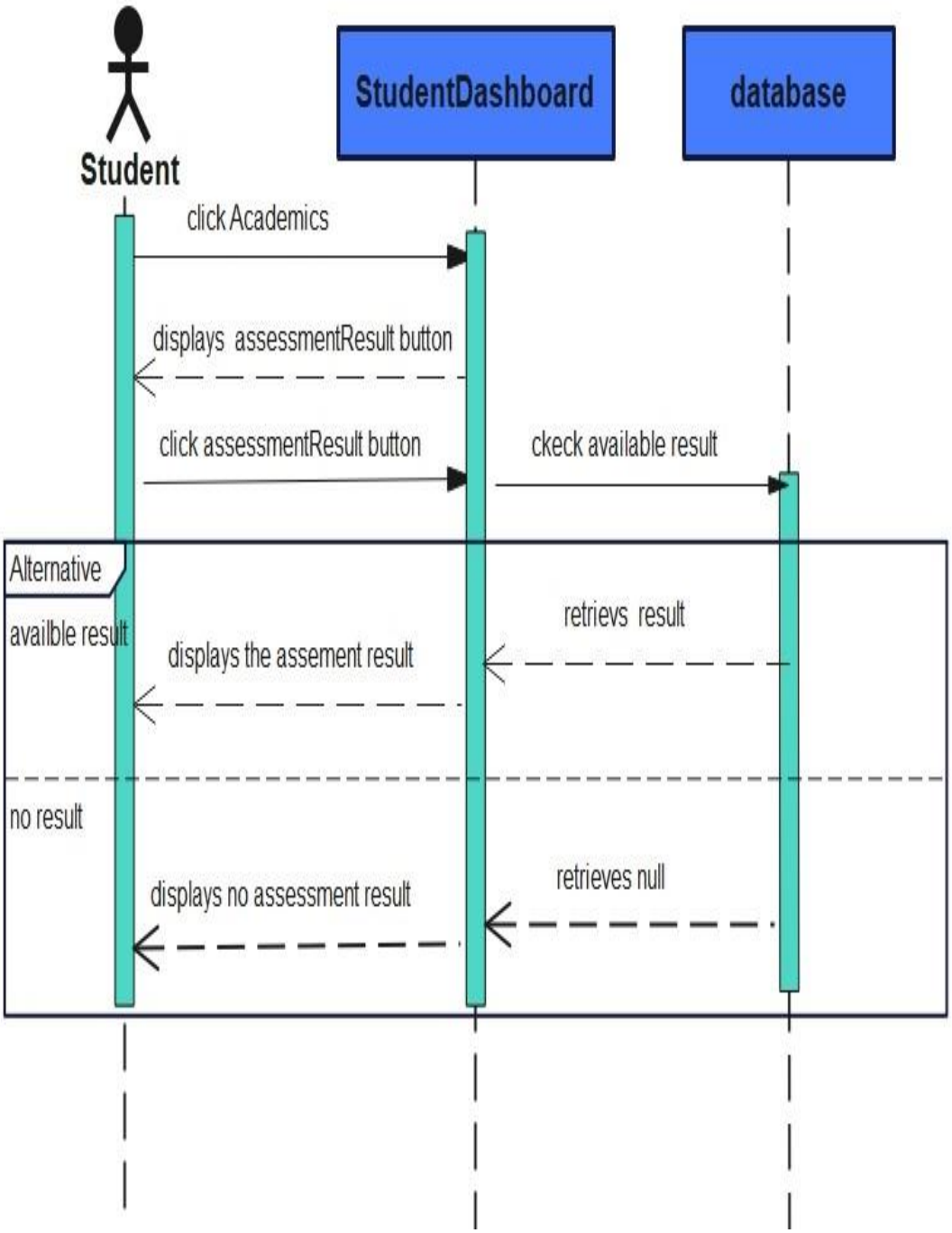


Figure 7: sequence diagram for viewing result

Assum the admin is successfully signed in.

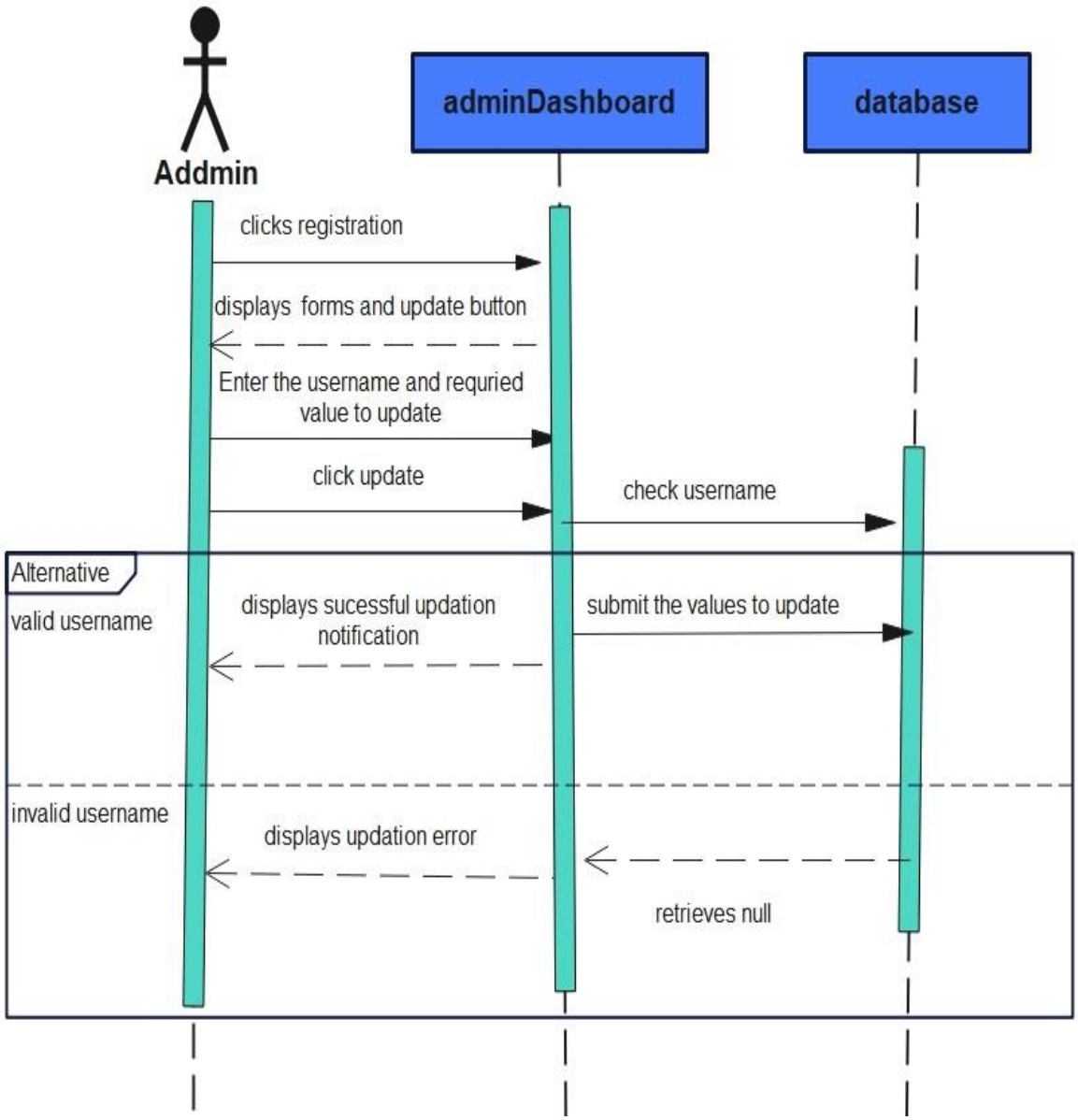


Figure 8: sequence diagram for update in admin

Assum the parent is signed in successfully.

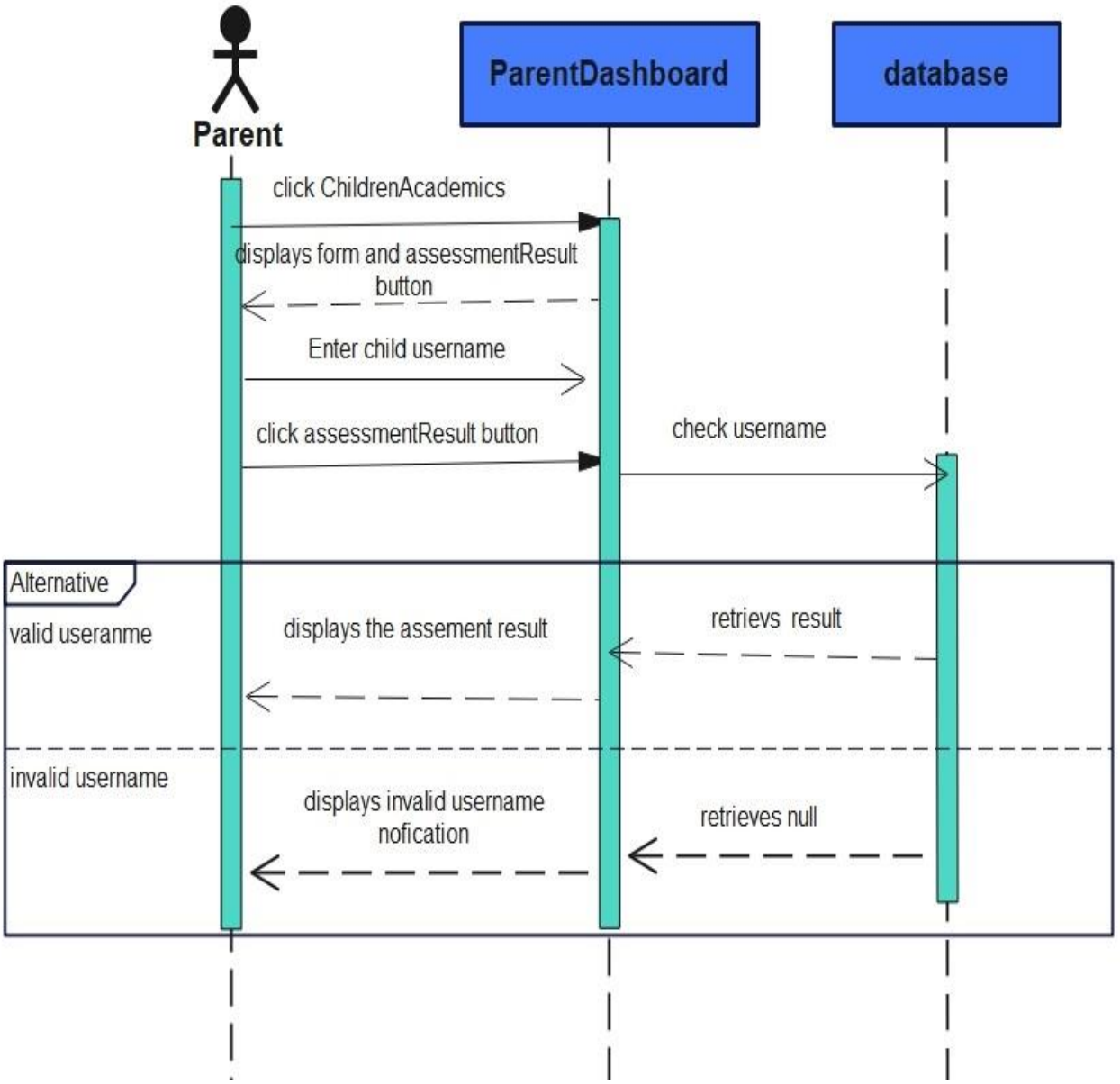


Figure 9: sequence diagram for view result for parent

2.3.5 Activity Diagram

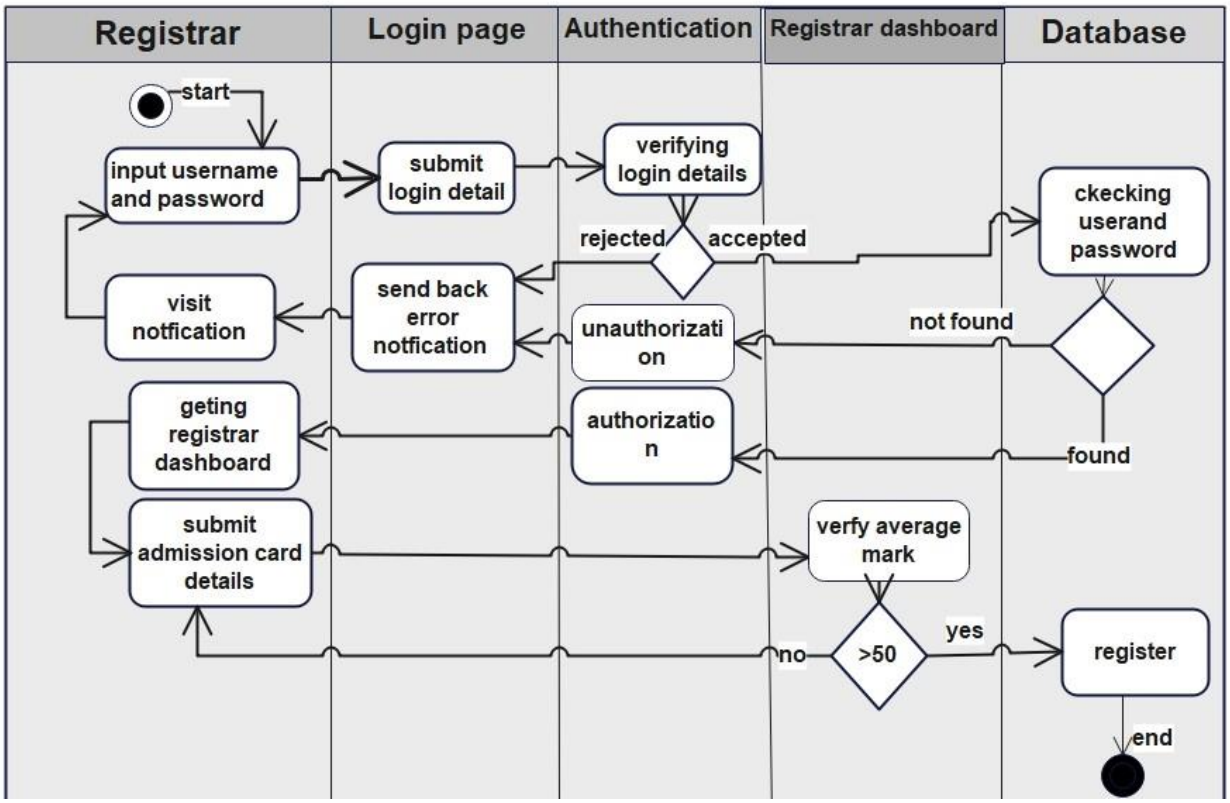


Figure 10: activity diagram for student registration

2.3.6 Class Diagram

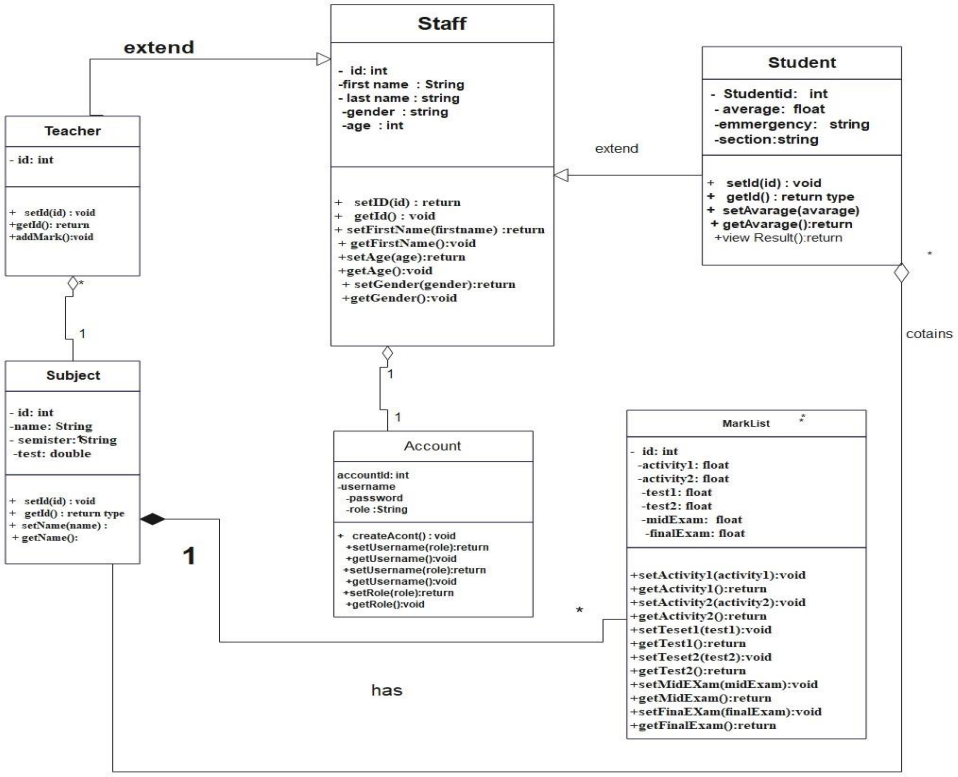


Figure 11 :class diagram MUCS

Chapter Three

System Design

3.1 Design Goals

Automation of Administrative Tasks

The primary goal is to automate manual administrative processes such as student registration, grade management, and communication, reducing the reliance on paper-based documentation and manual data entry in the school. By automating administrative tasks, the system aims to improve operational efficiency, reduce errors, and streamline workflows. This automation will save time for administrators, teachers, and staff, allowing them to focus on more value-added activities.

Enhancing Communication and Information Sharing

The system aims to enhance communication and information sharing among stakeholders, including administrators, teachers, students, and parents. This enhanced communication will facilitate timely dissemination of important announcements, academic updates, and event notifications.

Ensuring Data Accuracy, Integrity, and Security

Data accuracy, integrity, and security are paramount considerations for the system design. The goal is to ensure that all data stored and processed by the system is accurate, consistent, and secure.

Modernizing Administrative Processes

The system aims to modernize outdated administrative processes by leveraging technology and best practices in school management. Modernizing administrative processes will enable the school to keep pace with advancements in educational technology and adapt to changing needs and expectations. By embracing digital transformation, the school can improve efficiency, productivity, and competitiveness in the education sector.

Providing a User-Friendly Interface

The system will feature a user-friendly interface designed to enhance usability, accessibility, and user satisfaction. A well-designed user interface will make the system intuitive

and easy to navigate for users of all levels of technical proficiency. Clear navigation, intuitive controls, and responsive design will ensure a positive user experience, promoting adoption and acceptance of the system among stakeholders.

These design goals collectively aim to address the challenges identified in the current manual-based system and lay the foundation for a modern, efficient, and user-centric school management solution

3.2 Current System Architecture

The current system architecture at Mekelle University Community School relies primarily on manual administrative processes and traditional communication methods. It encompasses the following key components:

Manual Administrative Processes

Administrative tasks such as student registration, and grade management are predominantly manual. Student registration involves collecting admission cards manually, which contain personal information and academic records. Data and grade management are also paper-based, with teachers manually recording and calculating grades.

Limitations

Manual processes are time-consuming, error-prone, and inefficient. They rely on physical paperwork and documentation, making it challenging to manage and update information accurately and in a timely manner.

Paper-Based Documentation

Student records, gradebooks, and administrative documents are maintained in physical files or spreadsheet. This makes susceptible to loss, damage or misplacement. Accessing and retrieving information from paper-based documents can be cumbersome and time-consuming, hindering efficient data management and decision-making.

Traditional Communication Methods

Communication among stakeholders, including administrators, teachers, students, and parents, relies heavily on traditional methods such as phone calls, emails, and physical notices. This may cause miscommunication, and information gaps. Important announcements, academic updates, and event notifications may not reach all stakeholders effectively, leading to misunderstandings or missed opportunities.

Inconsistent Grading Practices

Grading criteria and assessment practices may vary across teachers and subjects, leading to inconsistencies in grading standards and practices. Inconsistent grading practices can affect the accuracy and fairness of student evaluations, potentially influencing academic outcomes and student perceptions of fairness.

The current system architecture, characterized by manual processes, paper-based documentation, and traditional communication methods, poses significant challenges in terms of efficiency, accuracy, and accessibility. Addressing these challenges requires a modern, automated school management system that streamlines administrative processes, enhances communication, and facilitates data-driven decision-making.

3.3 Proposed System Architecture

The proposed system architecture for Mekelle University Community School is designed to address the limitations of the current manual-based system and streamline administrative processes. It comprises the following key components:

Frontend Interface

The frontend interface serves as the user-facing component of the system, providing an intuitive and interactive platform for users to access and interact with system functionalities. React.js is used to develop frontend interface that offers dynamic and responsive user interfaces to facilitate efficient navigation and interaction. The frontend interface includes dashboards for administrators, registrar, teachers, students, and parents, each tailored to their specific roles and responsibilities. It allows users to perform tasks such as student registration, grade management, communication, and information access.

Backend Services

The backend services form the backbone of the system, handling business logic, data processing, and communication with the database. Spring Boot is used to develop the backend services to ensure robustness, scalability, and reliability of the system. The backend service facilitates user authentication, authorization, data validation, and processing.

Database

The database serves as the central repository for storing and managing the school's data, including student records, grades, schedules, and user information. We use PostgreSQL, the database offers a robust and scalable solution for data storage and retrieval.

API Integration

API integration facilitates communication between the frontend interface and backend services, enabling seamless data exchange and system interaction. RESTful APIs are used for communication, offering a lightweight and flexible approach to intercomponent communication. APIs define standardized endpoints and data formats for transmitting requests and responses between frontend and backend components. They enable functionalities such as user authentication, data retrieval, and system updates.

Security Measures

Security measures are implemented to ensure data security, integrity, and privacy throughout the system. Role-based access control (RBAC) is employed to restrict access to specific functionalities and data based on user roles and permissions.

The proposed system architecture integrates frontend(react.js), backend (spring boot), and database (Postgres) components, leveraging modern technologies and best practices to provide a robust, efficient, and user-friendly school management solution. By automating administrative processes, enhancing communication, and ensuring data security and integrity, the proposed system aims to modernize school management practices and improve overall operational efficiency.

3.4 Subsystem Decomposition

The subsystem decomposition of the proposed system for Mekelle University Community School involves breaking down the system into smaller, more manageable components or subsystems.

Each subsystem is responsible for specific functionalities or tasks within the overall system. The subsystem decomposition includes the following components:

Admin Dashboard

The Admin Dashboard subsystem provides administrators with tools and functionalities to manage user accounts, oversee system operations, and access comprehensive reports. It includes features such as user account management, subject and section assignment, exam scheduling, and report generation. The Admin Dashboard empowers administrators to efficiently oversee administrative tasks, monitor system performance, and generate insights to inform decision-making processes. **Registrar Dashboard**

The Registrar Dashboard subsystem facilitates student registration, admission card processing, and enrollment management. It includes features such as admission card registration, new student registration, and enrollment verification. The Registrar Dashboard streamlines the registration process, ensures accurate and timely enrollment, and maintains up-to-date student records.

Teacher Dashboard

The Teacher Dashboard subsystem empowers teachers to manage assessment results, track student progress, and communicate with students and parents. It includes features such as assessment result entry, progress tracking, and communication tools. The Teacher Dashboard enables teachers to effectively assess student performance, provide timely feedback, and facilitate communication with students and parents to support academic success.

Parent Dashboard

The Parent Dashboard subsystem provides parents with a centralized platform to access their child's academic information, communicate with teachers, and stay updated on school events and announcements. It serves as a crucial component of the proposed school management system, enhancing parental involvement, engagement, and support in their child's education journey.

Student Dashboard

The Student Dashboard subsystem provides students with access to their academic records, assessment results, and school announcements. It includes features such as result viewing, and event notifications. The Student Dashboard empowers students to monitor their academic progress, access important information, and stay informed about school events and announcements.

Security Subsystem

The Security Subsystem ensures data security, integrity, and privacy throughout the system. It includes features such as authentication, authorization, and role-based access control.

3.5 Component Diagram

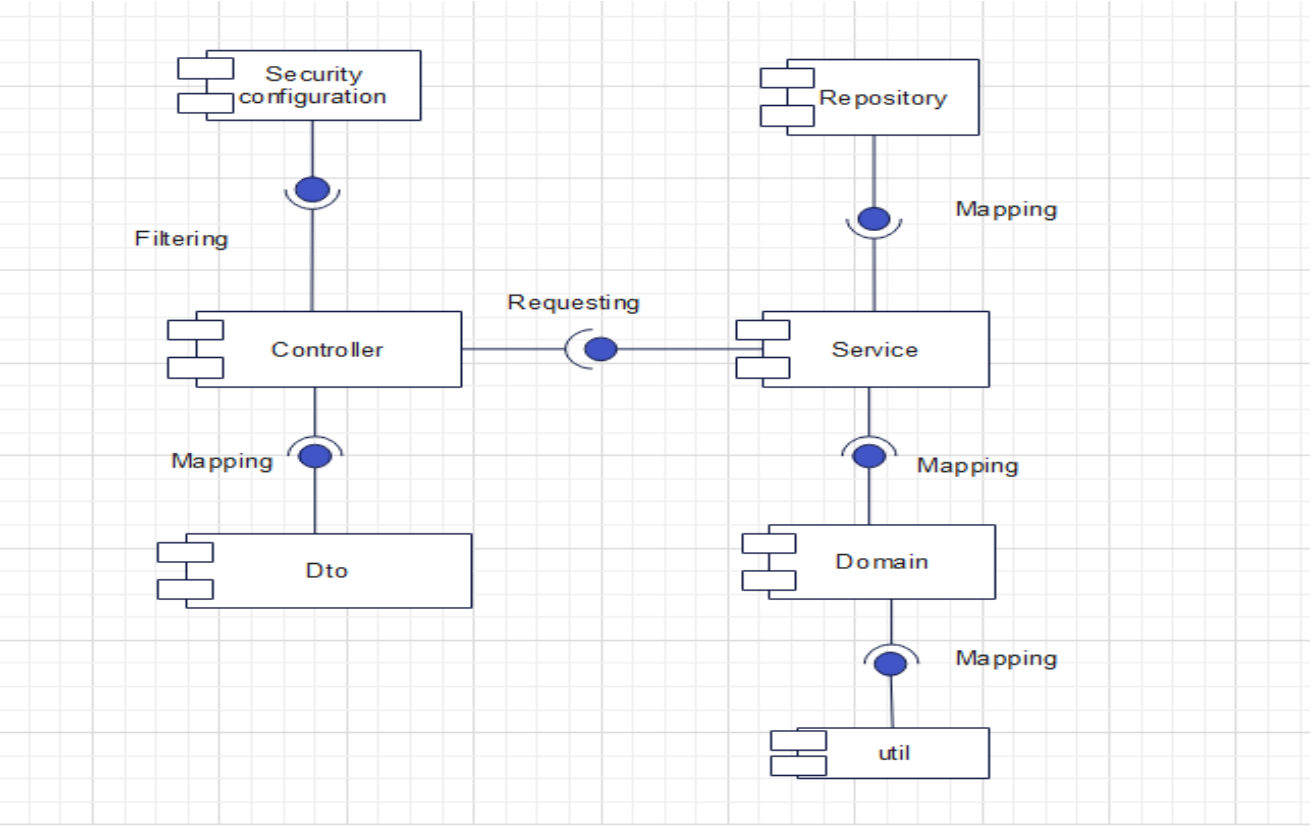


Figure 12 : component diagram MUCS

3.6 Deployment Modeling

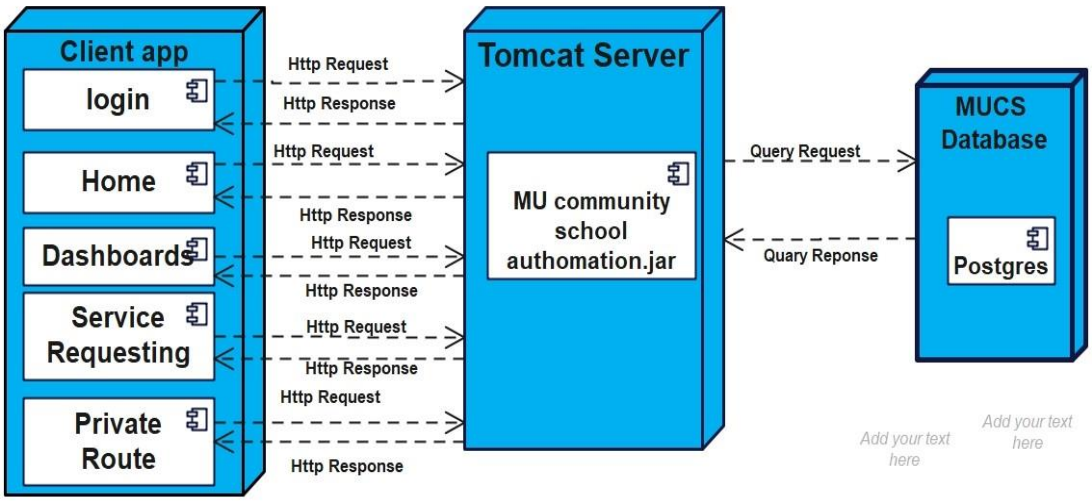


Figure 13: deployment diagram for MUCS

Description for deployment modeling

The deployment model illustrates how the software components of the school management system, utilizing PostgreSQL for the database, React.js for the frontend, and Spring Boot for the backend, are mapped onto hardware infrastructure to ensure efficient deployment and operation.

Client App (Web Server)

Hosts the frontend interface components developed using React.js, serving web pages and managing client requests.

Application Server

Hosts the backend services components developed using Spring Boot, executing business logic, processing data, and communicating with the PostgreSQL database.

3.7 Persistent Data Management

Data Storage

The PostgreSQL database serves as the backbone of our data storage system, acting as the central repository for a wide range of information. This encompasses vital data related to teachers, parents, administrators, and all other users involved in the educational process. Our database architecture is designed to manage information into structured tables. These tables are created with defined relationships and constraints, ensuring the integrity and consistency of the stored data across all categories of users. From grades and schedules to user profiles and system configurations, every piece of data finds its place within this robust and organized framework.

Data Retrieval

In data retrieval, Structured Query Language (SQL) queries serve as the primary tool for accessing and manipulating database information. SQL queries allow users to specify precisely the data they need, whether it's retrieving specific subsets of data, performing complex calculations, or aggregating information from multiple tables.

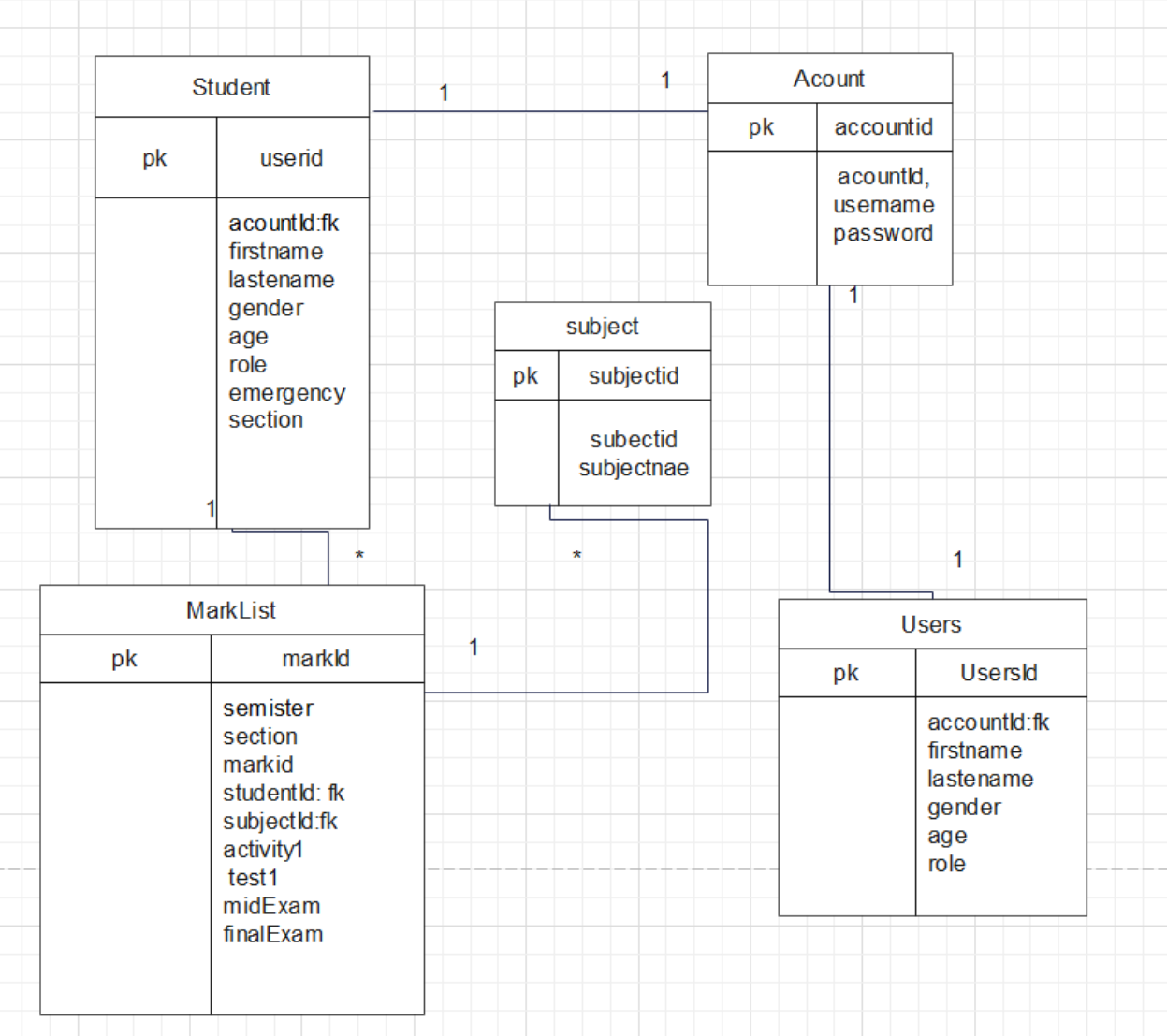


Figure 14: data persistence for MUCS

3.8 Access Control and Security

Access control and security mechanisms ensure that only authorized users can access the system's resources and that sensitive data is protected from unauthorized access or manipulation. Leveraging JWT for authentication and authorization enhances the security posture of the system.

Authentication with JWT

Users authenticate by providing their credentials (e.g., username and password) to the system. Upon successful authentication, the system generates a JWT containing the user's identity and additional claims. After authentication, the server generates a JWT signed with a secret key or

private key/public key pair. The JWT encapsulates user identity, permissions, and other relevant information in its payload.

Authorization with JWT

When a user attempts to access protected resources, they include the JWT in the request's Authorization header. The server verifies the JWT's signature and decodes its payload to extract user identity and permissions. Based on the information extracted from the JWT, the server enforces access control policies to determine whether the user is authorized to access the requested resource. Access control policies is role-based.

Token Expiration

JWLTS include expiration timestamps to limit their validity period. Token expiration mitigates the risk of authorized access.

3.9 User Interface Design

User Interface Design focuses on creating interfaces that are intuitive, user-friendly, and visually appealing. It aims to optimize the interaction between users and the system, ensuring ease of use and efficiency in completing tasks.

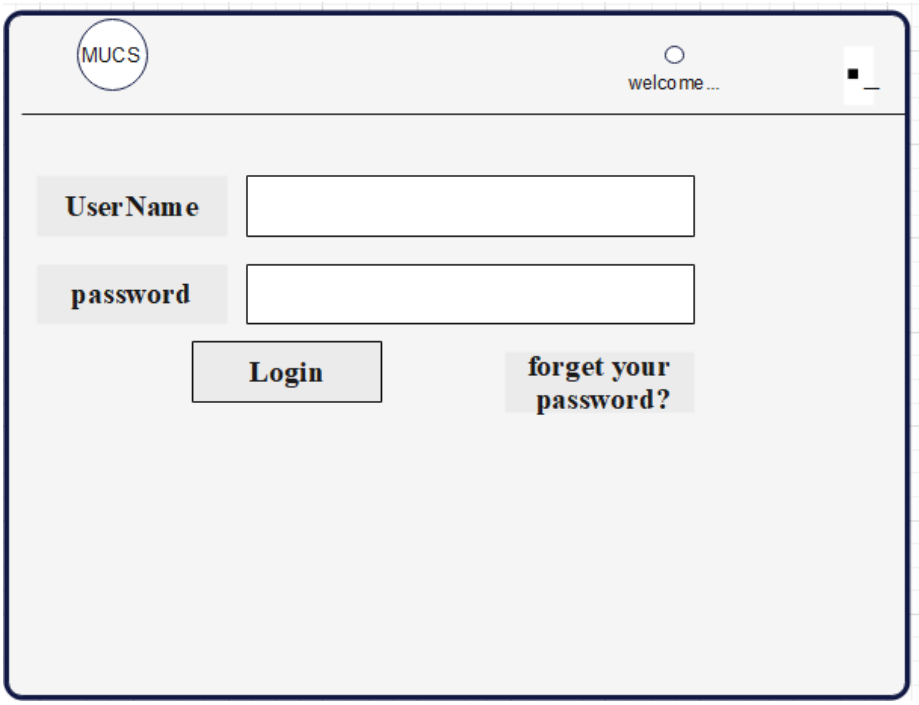


Figure 15: login user interface

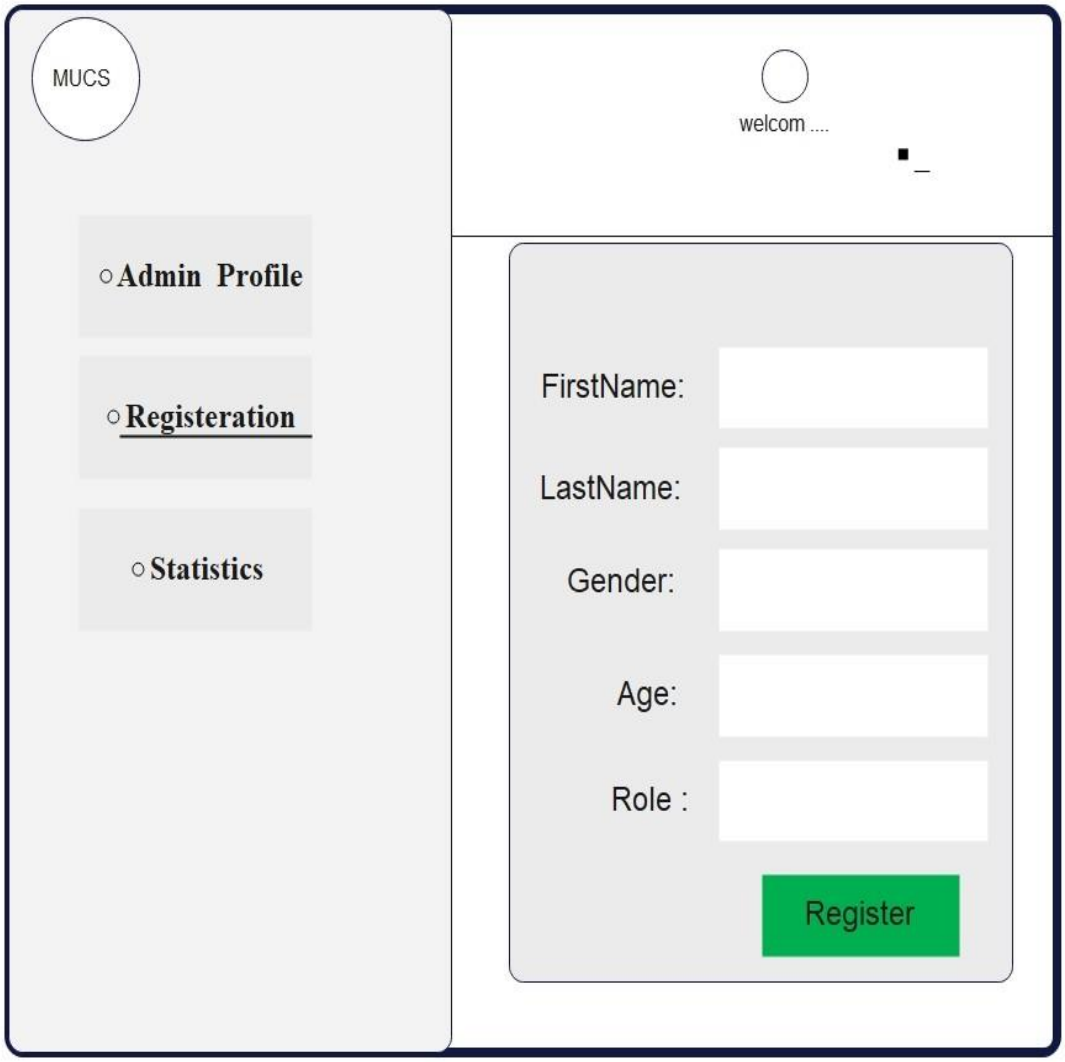


Figure 16: admin user interface for registration

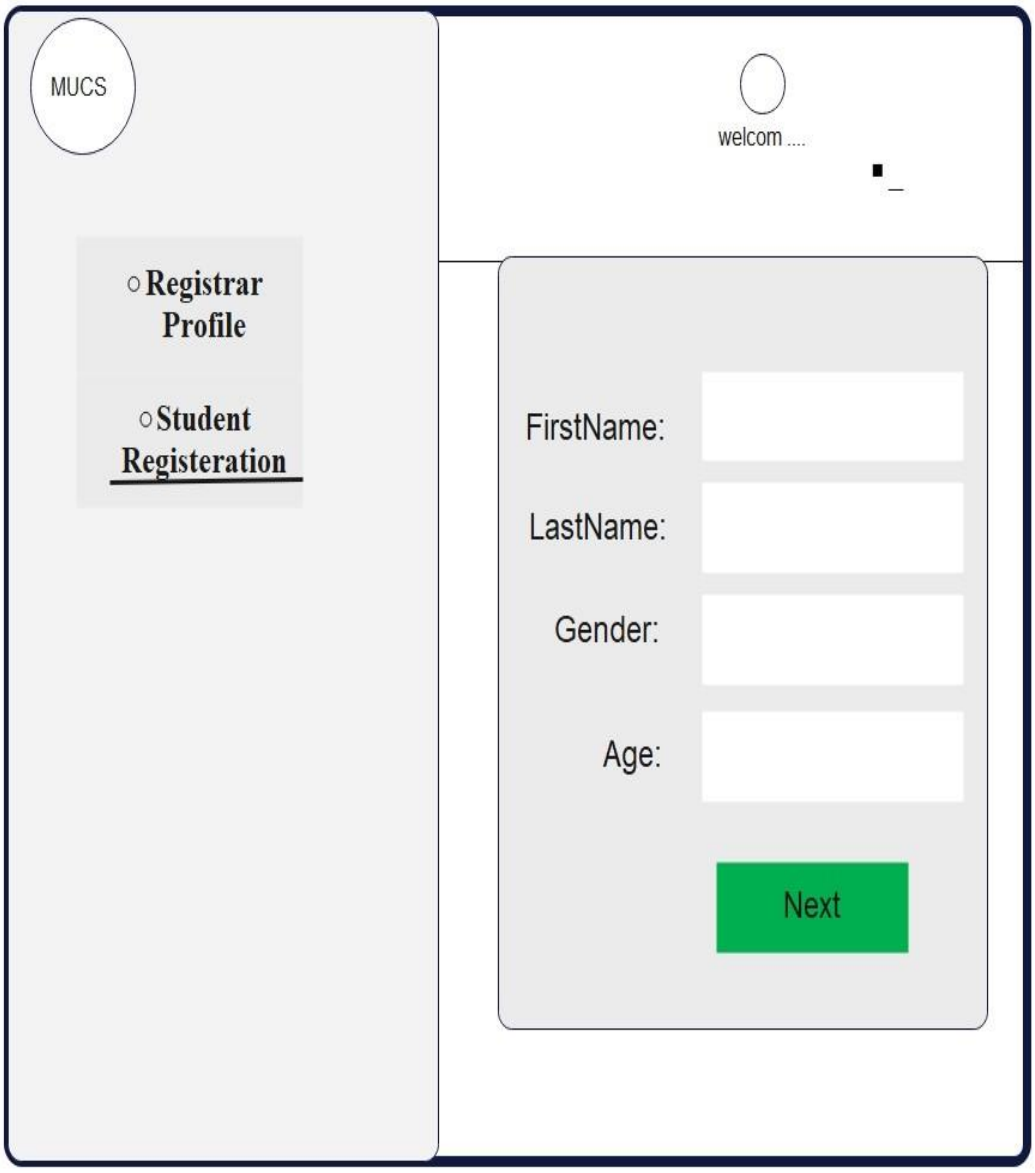


Figure 17: Registrar user interface student registration

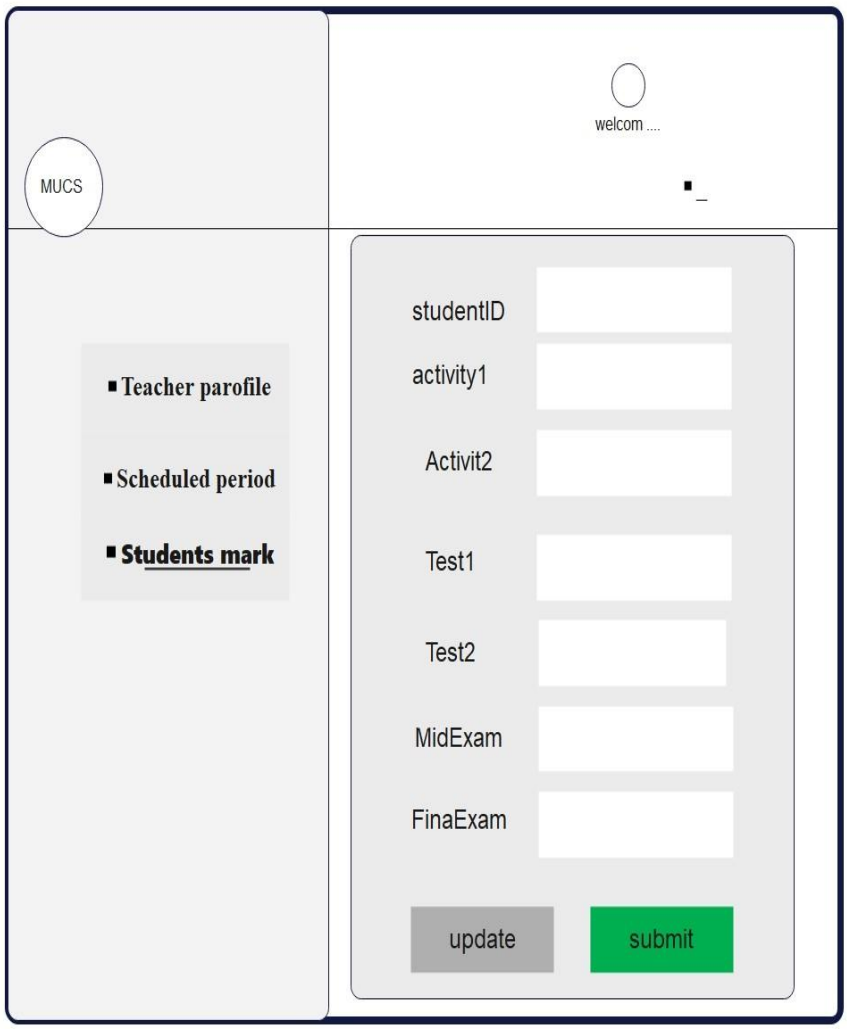


Figure 18: Teacher user interface design

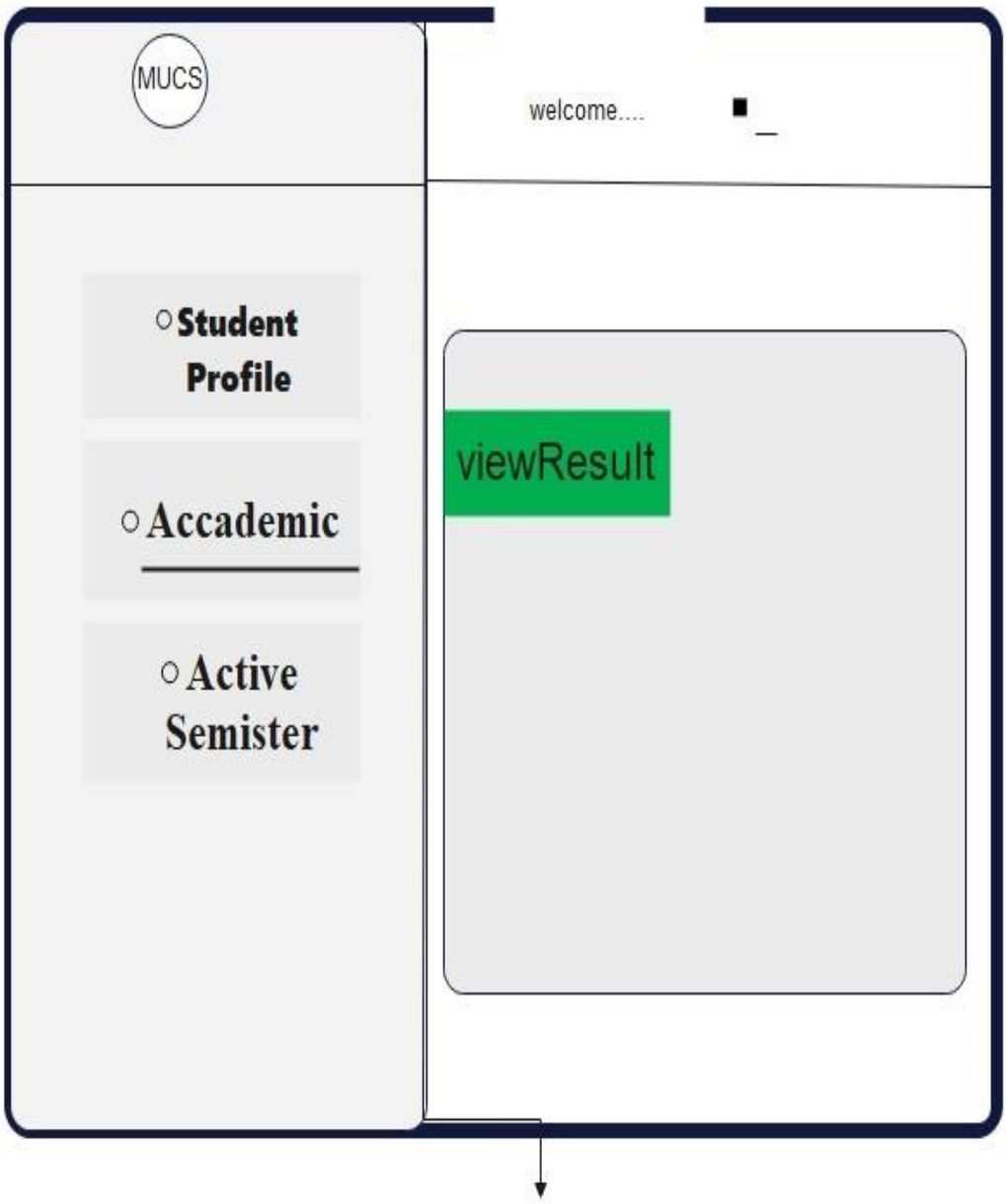


Figure 19: Student user interface design

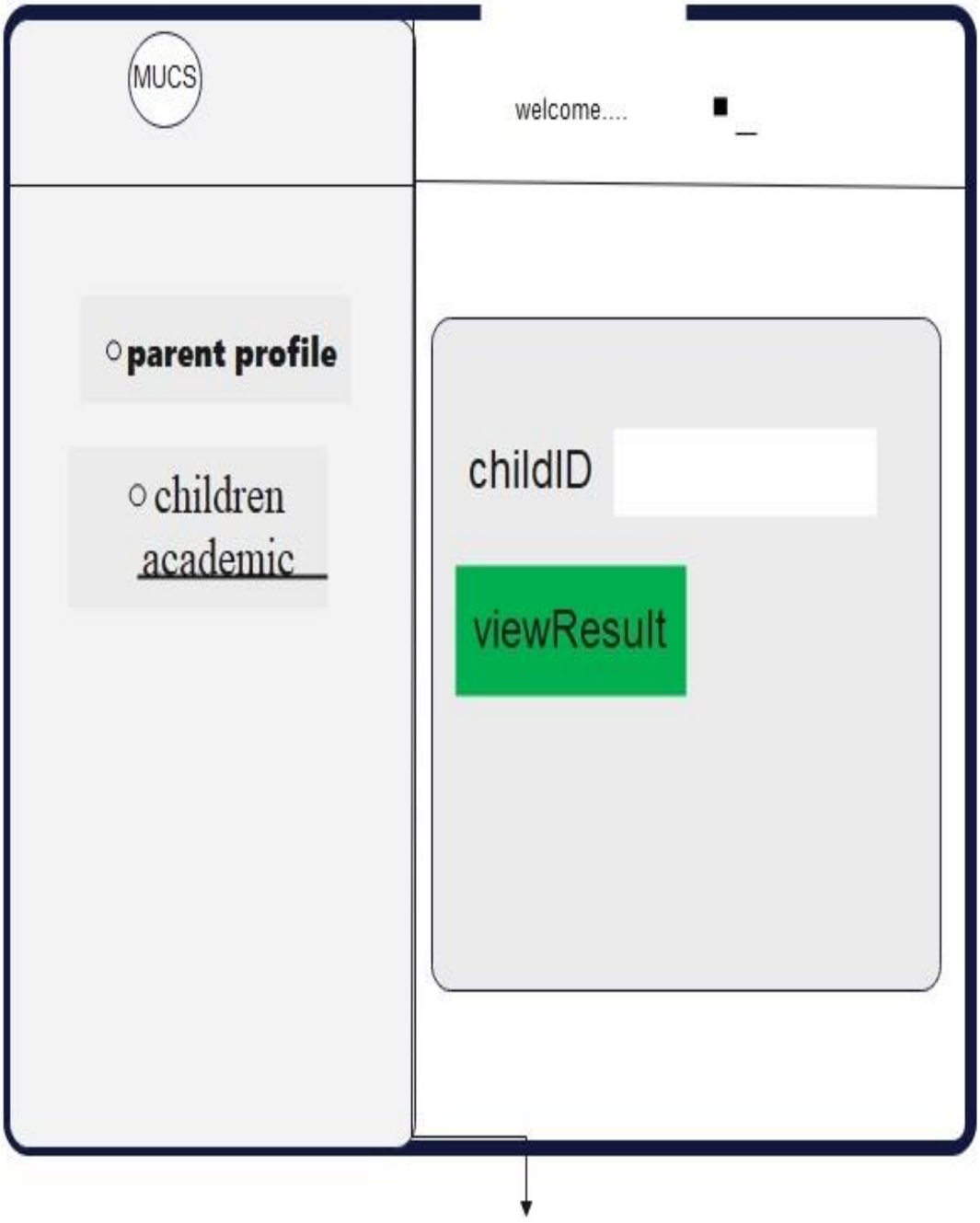


Figure 20 : Parent user interface

Chapter Four: Implementation and Testing

4.1 Coding

This section presents sample code snippets from the UserController class of the high school automation system, demonstrating key functionalities implemented within the controller.

4.1.1 User Controller Class

```
package com.mekelleuniversity.comunityschool.controller;

import com.mekelleuniversity.comunityschool.demain.*;

import com.mekelleuniversity.comunityschool.dto.*;

import com.mekelleuniversity.comunityschool.service.AuthenticationService;

import com.mekelleuniversity.comunityschool.service.JwtService;

import com.mekelleuniversity.comunityschool.service.RegistrarService;

import com.mekelleuniversity.comunityschool.service.UserService;

import io.jsonwebtoken.ExpiredJwtException;

import lombok.RequiredArgsConstructor;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.annotation.AuthenticationPrincipal;

import org.springframework.web.bind.annotation.*;

@RestController

@RequestMapping("/api/users")

@RequiredArgsConstructor
```

```
public class UserController {
    @Autowired
    private UserService userService;
    @Autowired
    private JwtService jwtService;
    private final RegistrarService registrarService;
    private final AuthenticationService service;

    @PostMapping("/register")
    public ResponseEntity<AuthenticationResponse> register(
        @RequestBody RegisterRequest request) {
        return ResponseEntity.ok(service.register(request));
    }
    @PostMapping("/login")
    public ResponseEntity<AuthenticationResponse> authenticate(
        @RequestBody AuthenticationRequest request) {
        return ResponseEntity.ok(service.authenticate(request));
    }

    @PostMapping("/adminCreate")
    public ResponseEntity<Users> registerUser(@RequestBody UserDTO userDTO) {
        Users createdUser = userService.adminCreateAccounts(userDTO);
        return ResponseEntity.ok(createdUser);
    }
```

```
@PostMapping("/studentAccount")
public ResponseEntity<Student> createStudentsAccount(
    @RequestBody StudentDto studentDto) {
    Student createdUser = registrarService.createStudentAccount(studentDto);
    return ResponseEntity.ok(createdUser);
}

@PostMapping("/parentAccount")
public ResponseEntity<Parent> createParentsAccount(
    @RequestBody ParentDto parentDto) {
    Parent createdUser = registrarService.createParentAccount(parentDto);
    return ResponseEntity.ok(createdUser);
}

@PutMapping("/assignSection/{username}")
public ResponseEntity<Teacher> assignSection(
    @PathVariable String username,
    @RequestBody TeacherDto teacherDto) {
    Teacher teacher = registrarService.assignSection(username, teacherDto);
    return ResponseEntity.ok(teacher);
}

@PostMapping("/addCourse")
```

```
    public ResponseEntity<Course> addCourses(@RequestBody CourseDto courseDto) {
        Course course = registrarService.addCourse(courseDto);
        return ResponseEntity.ok(course);
    }

    @GetMapping("/validate")
    public ResponseEntity<?> validateToken(
        @RequestParam String token,
        @AuthenticationPrincipal Accounts accounts) {
        try {
            Boolean isTokenValid = jwtService.validateToken(token, accounts);
            return ResponseEntity.ok(isTokenValid);
        } catch (ExpiredJwtException e) {
            return ResponseEntity.ok(false);
        }
    }
}
```

4.1.2 Explanation of Code

User Authentication*:* The /login endpoint authenticates users based on their credentials provided in the Authentication Request.

Admin User Creation*:* The /adminCreate endpoint enables the admin to create new user accounts.

Student and Parent Account Creation: The /studentAccount and /parentAccount endpoints facilitate the creation of student and parent accounts, respectively by registrar**.**

Assign Section to Teacher: The /assignSection/{username} endpoint assigns sections and course to teachers based on the username and provided Teacher to**.**

Course Addition: The /addCourse endpoint allows the addition of new courses to the system**.**

Token Validation: The /validate endpoint checks if the provided JWT token is valid**.**

4.2 System Testing

There are different types of testing conducted on the system, including unit testing, integration testing, system testing, and specifications for test cases.

4.2.1 Types of Testing

4.2.1.1 Unit Testing

Unit testing focuses on testing individual components or methods in isolation. The purpose is to validate that each unit of the software performs as intended.

Example: Using JUnit for testing the Authentication Service.

```
import static org.mockito.Mockito.*;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;
```

```
import org.mockito.junit.jupiter.MockitoExtension;
import org.junit.jupiter.api.extension.ExtendWith;
@ExtendWith(MockitoExtension.class)
public class AuthenticationServiceTest {
    @Mock
    private UserService userService;
    @InjectMocks
    private AuthenticationService authenticationService;
    @Test
    void testRegister() {
        RegisterRequest request = new RegisterRequest("John", "Doe", "john.doe", "password");
        AuthenticationResponse response = authenticationService.register(request);
        assertNotNull(response);
        assertEquals("john.doe", response.getUsername());
    }}
```

4.2.1.2 Integration Testing

Integration testing evaluates the interactions between different modules or services to ensure they work together correctly.

Example: Testing the interaction between UserController and AuthenticationService**.**

```
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.test.web.servlet.MockMvc;
@WebMvcTest(UserController.class)
public class UserControllerIntegrationTest {
    @Autowired
    private MockMvc mockMvc;
    @MockBean
    private AuthenticationService authenticationService;
    @Test
    void testUserRegistration() throws Exception {
        RegisterRequest request = new RegisterRequest("John", "Doe", "john.doe", "password");
        when(authenticationService.register(any())).thenReturn(new
AuthenticationResponse("token", "john.doe"));
        mockMvc.perform(post("/api/users/register")
                .contentType("application/json")
.content("{\"firstname\":\"muluneh\",\"lastname\":\"Gebre\",\"username\":\"MU/ST1580/SC24",\
"password\":\"mu1234#\"}"))
                .andExpect(status().isOk())
                .andExpect(jsonPath("$.username").value("MU/ST1580/SC24"));
    }
}
```

4.2.1.3 System Testing

System testing involves testing the complete and integrated software application to ensure it meets specified requirements. This includes end-to-end testing of user flows.

Example: Verifying the full registration and login flow through the application.

4.2.1.4 Black-box Testing

In black-box testing, the tester evaluates the application based on the inputs and outputs without any knowledge of internal code structure. This type of testing ensures that the software behaves as expected from an end-user perspective**.** Example login. If user enters his correct username and password, he will get his respected page based on his role.

4.2.1.5 White-box Testing

White-box testing involves testing internal structures or workings of an application, as opposed to its functionality. It requires knowledge of the internal code and is often used to ensure that all pathways are tested.

4.2.2 Test Case Specification

Test case Id	test case description	*Input data*	*Expected output*
TC-001	Test admin creating a user account	Valid User DTO	Success response with created User Account
TC-002	Test user login	Valid AuthenticationReques	Success response with AuthenticationRespons
TC-003	Test parent account creation	Valid ParentDto	success response with created Parent
TC-004	Test student account creation	Valid StudentDto	Success response with created Student
TC-005	*Test section assignment for a teacher*	*Valid Teacher username*	*Success response with assigned Teacher*
TC-006	*Test entering grade mark*	*Valid student usernam*	*Successful response submit grade mark of student.*

4.2.3 Test Execution

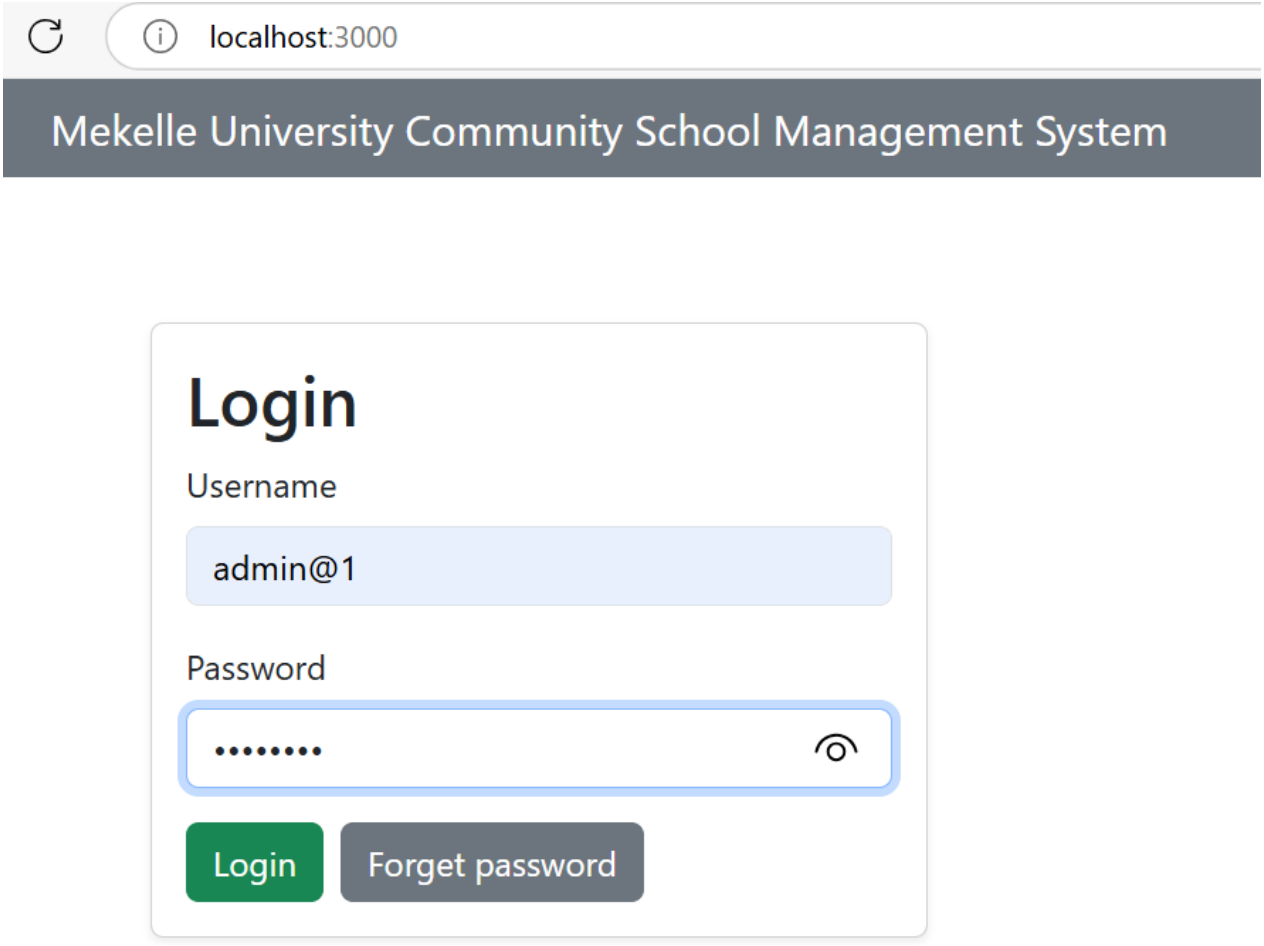
Each test case will be executed, and the actual outputs will be recorded. The status will be updated to reflect whether the test passed or failed. Testing tools such as JUnit and Mockito will be utilized for unit and integration testing.

4.3 User Interface/Sample Screenshots

The sample screenshots of the user interface, demonstrating the functionalities of the application.

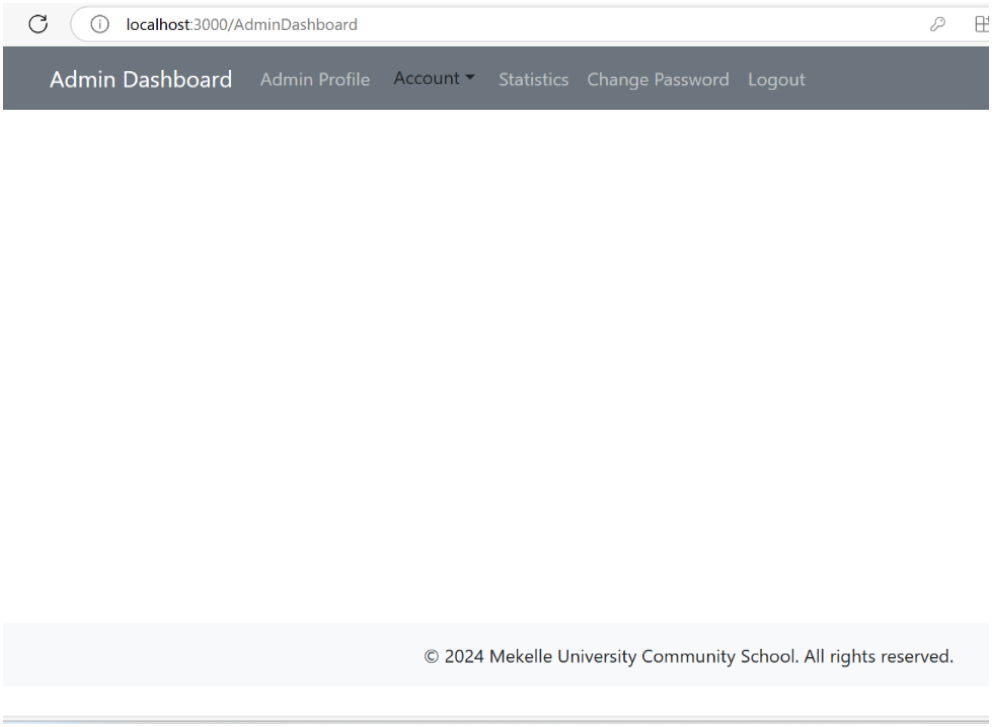
4.3.1 User Login Page

Description: The user login interface allows users to enter their credentials to access their respective pages.



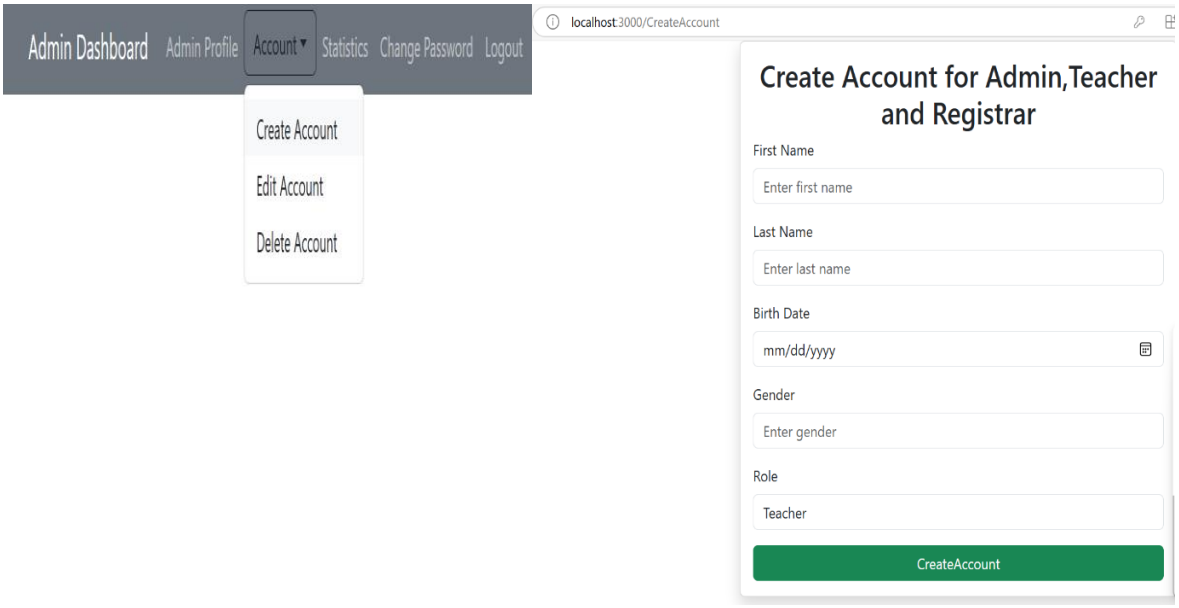
4.3.2 Admin Dashboard

Description: This admin dashboard Contains Activities in which admins perform. Such as view personal details, manage user accounts and monitoring system activities**.**



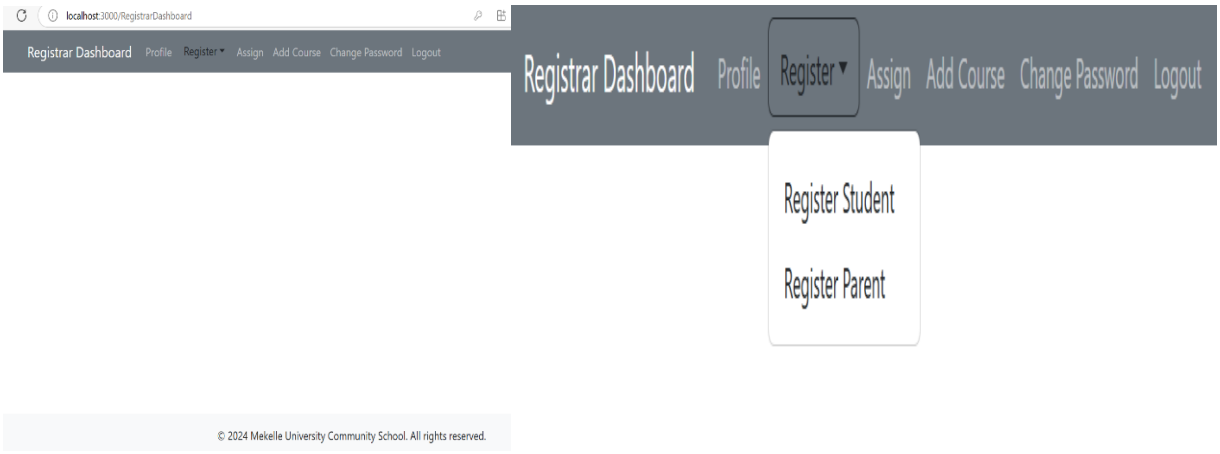
4.3.1 User Registration Page

Description: user registration enables the admin to create and manage accounts and personal details of admin, registrar and teacher.



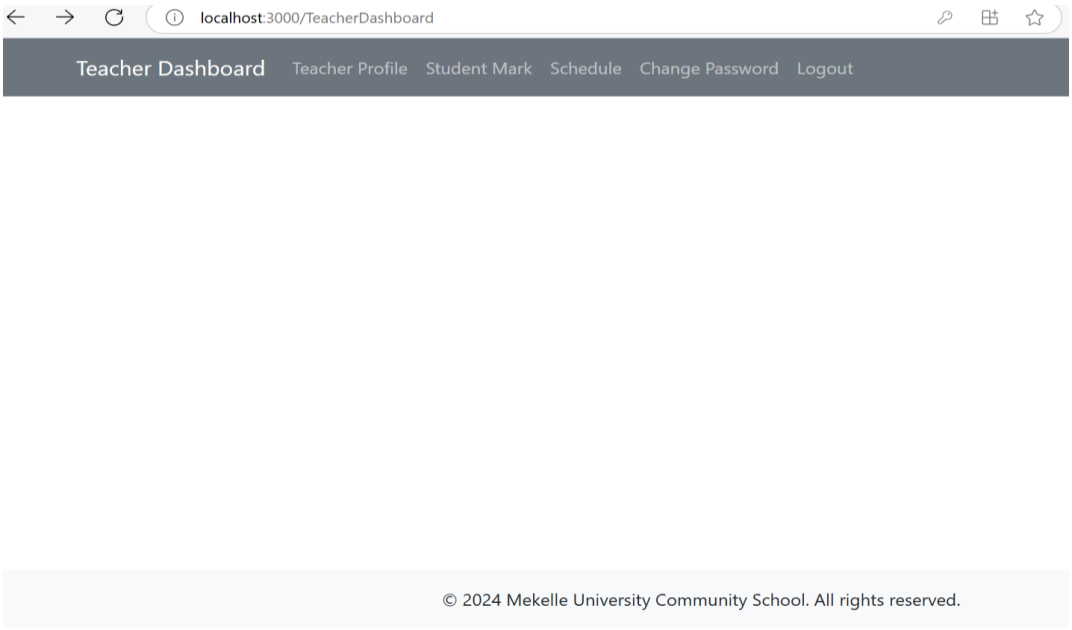
4.3.4 Registrar Dashboard

Description: Registrar dashboard Contains activities what performs Registrars such as to view personal details to manage student, parent accounts and assign section and course to teachers.



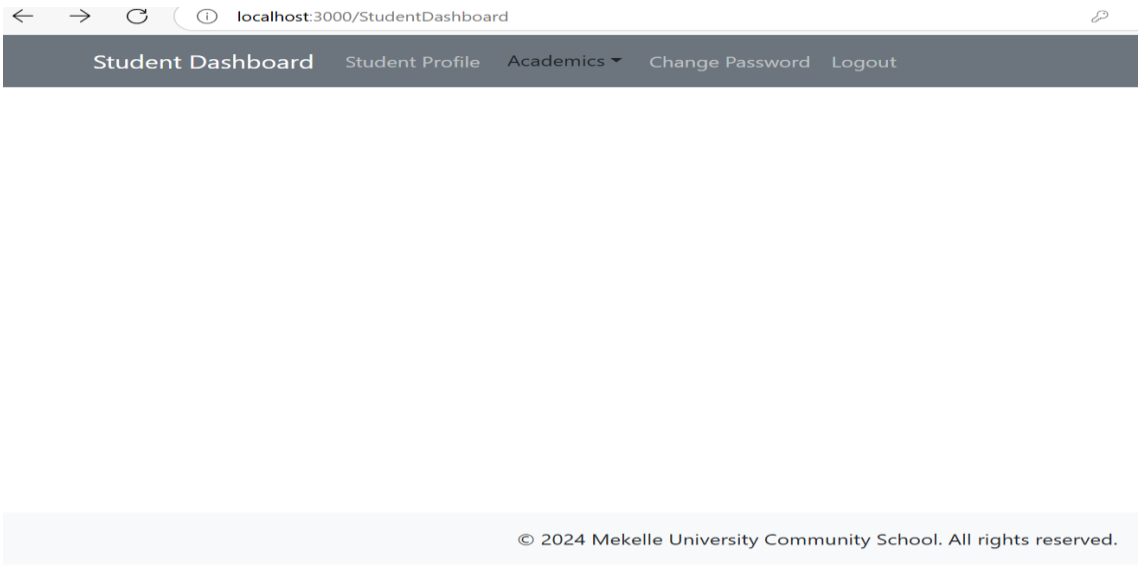
4.3.4 Teacher Dashboard

Description: Teacher dashboard Contains activities what performs teacher such as to view personal details, change password, enter student marks.



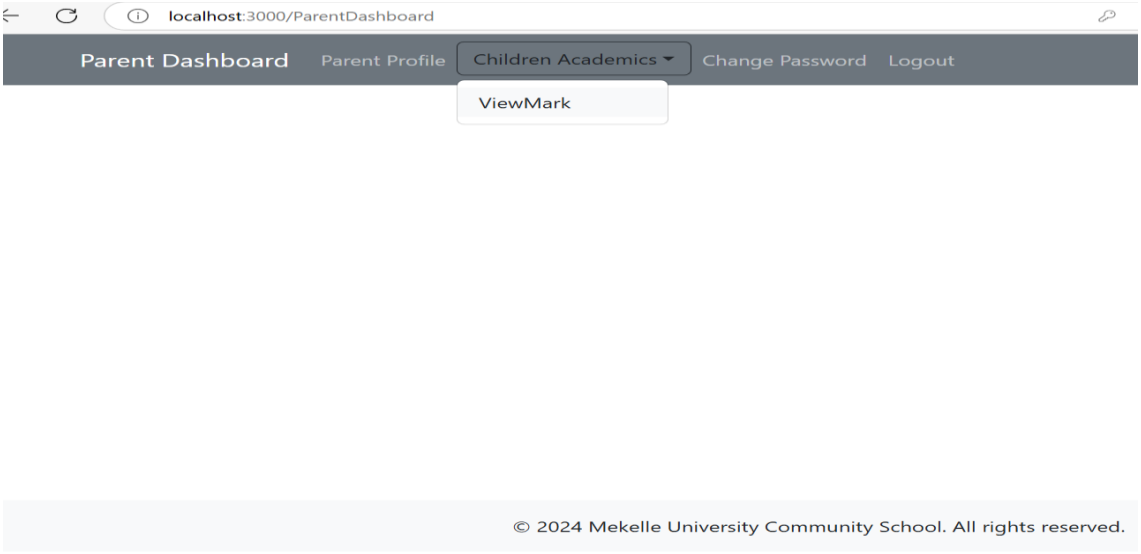
4.3.4 Student Dashboard

Description: Student dashboard Contains activities students can perform on their pages such as view personal details, change password, view results**.**



4.3.5 Parent Dashboard

Description: Parent Dashboard contain activities in which parents can perform on their pages such as personal details, change password, view children academics**.**



Appendix

Appendix A: Technical Specifications

A.1 Software Requirements

Frontend: React.js

Backend: Spring Boot

Database: PostgreSQL

APIs: RESTful APIs

A.2 Hardware Requirements

Client Devices: Computers or mobile devices with internet access.

Servers: Application server for backend services, web server for frontend, and database server for PostgreSQL.

A.3 Development Tools

IDE: IntelliJ IDEA for backend, Visual Studio Code for frontend.

Version Control: Git, hosted on GitHub or GitLab.

Appendix B: User Roles and Permissions

B.1 Roles

Administrator: Full access to all system functionalities.

Registrar: Access to student registration and enrollment management.

Teacher: Access to grade entry, student progress tracking, and communication tools.

Parent: Access to child's academic information and school announcements.

Student: Access to own academic records and school announcements.

B.2 Permissions

Admin Dashboard: User management, report generation.

Registrar Dashboard: Student registration, admission card processing.

Teacher Dashboard: Grade entry, communication with students/parents.

Parent Dashboard: View academic records, receive notifications.

Student Dashboard: View grades, receive notifications.

Appendix C: Data Structure

Tables: Users, Students, Teachers, Courses, Grades, Announcements.

Relationships: Foreign keys linking students to grades, teachers to courses, etc.

Appendix D: Security Measures

D.1 Authentication and Authorization

JWT Tokens: Used for secure authentication and authorization.

Role-Based Access Control (RBAC): Ensures users access only what they're permitted.

Appendix E: interview Questionnaire used during requirement gathering

Questions

School Establishment

Q1: When was the school established?

Student Admission

Q2: How many students do you accept per year?

1. Challenges Faced by the School

Q3: As a school, what problems do you face with the current manual-based system?

2. Challenges Faced by Teachers

Q4: As a teacher, what problems do you encounter due to the school using a manual-based system?

3. Challenges Faced by Students

Q4: As a student, what problems do you encounter due to the school using a manual-based system?

References

Books and Textbooks:

1. Januszewski, A., & Molenda, M. (Eds.). (2008). Educational Technology: A Definition with Commentary. Routledge.

2. Gorton, R. A. (2019). School Leadership and Administration: Important Concepts, Case Studies, and Simulations. SAGE Publications Inc.

3. Ramakrishnan, R., & Gehrke, J. (2002). Database Management Systems. McGraw-Hill Education.

4. Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.

Articles and Papers

1. Smith, J., & Johnson, L. (2017). The Role of Technology in Modern School Management Systems. Journal of Educational Technology & Society, 20(2), 112-125.

2. Brown, A., & Davis, C. (2018). Enhancing Communication and Information Sharing in Educational Institutions. Computers & Education, 126, 45-58.

3. Patel, R., & Gupta, S. (2020). Data Accuracy and Security Measures in Educational Systems. Journal of Information Technology Education, 19, 78-91.

4. Chen, H., & Wang, L. (2019). Modernizing Administrative Processes in Schools: A Case Study Analysis. Educational Technology Research and Development, 67(5), 1023-1036.