Name: Muhammad Anas

# Abstract

The GradeTracker application represents a comprehensive solution designed to empower students to manage their academic progress effectively. Developed using Java programming language and JavaFX framework, GradeTracker offers an intuitive user interface and robust functionality for tracking course grades, setting academic goals, and managing assignments. Through user-centered design principles and iterative development processes, GradeTracker prioritizes usability, security, and efficiency, ensuring a seamless user experience. This application is designed with the sole intention of improving the lives of students everywhere, regardless of their program of choice. Throughout the semester courses tend to move at a very fast pace, making it far too easy for students to become disorganized. Forgetting an assignment's due date, not knowing the value of an assignment, and not knowing one's current grade in a course are just some of the problems that this application provides solutions for. The application facilitates goal setting for individual courses, allowing students to monitor their progress toward academic targets. Additionally, GradeTracker provides features for assignment management, enabling users to stay organized and informed about upcoming deadlines The project also emphasizes compliance with legal and ethical standards, ensuring confidentiality and integrity of user data. Through reflection on project achievements, lessons learned, and future directions, GradeTracker aims to continuously evolve and enhance its capabilities, making a meaningful impact on students' academic experiences.

# Table of Contents

[Abstract 2](#_Toc165688336)

[Table of Contents 3](#_Toc165688337)

[Chapter 1: Introduction 5](#_Toc165688338)

[1.1 Project topic and rationale 5](#_Toc165688339)

[1.2 Project Aims and Objectives 5](#_Toc165688340)

[1.3 Methodology 5](#_Toc165688341)

[1.4 The report structures 6](#_Toc165688342)

[Chapter 2: Background Research 7](#_Toc165688343)

[2.1 Literature review of related work 7](#_Toc165688344)

[2.2 Critical evaluation of related products/solutions 7](#_Toc165688345)

[2.3 The scope of the project 8](#_Toc165688346)

[2.4 Review and justification of theories/models/development platforms/tools selected for use in the project 8](#_Toc165688347)

[Chapter 3: Requirements Analysis and Specification or Problem Analysis and Specification 9](#_Toc165688348)

[3.1 Requirement Elicitation 9](#_Toc165688349)

[3.2 Analysis and Specification 9](#_Toc165688350)

[3.3 Functional Requirements 9](#_Toc165688351)

[3.4 Non-Functional Requirements 10](#_Toc165688352)

[3.5 Use Cases 10](#_Toc165688353)

[3.6 Acceptance Criteria 11](#_Toc165688354)

[Chapter 4: Software Design 12](#_Toc165688355)

[4.1 User-Interface Design 12](#_Toc165688356)

[4.1.1 Mock-up Screens: 12](#_Toc165688357)

[4.2 Database Tables’ Structure Design 14](#_Toc165688358)

[4.2.1 Database Tables: 14](#_Toc165688359)

[4.3 4Main Components of the Software Architecture 15](#_Toc165688360)

[4.3.1 Main Components: 15](#_Toc165688361)

[4.4 Detailed Software Class Design 16](#_Toc165688362)

[4.4.1 Class Diagram: 16](#_Toc165688363)

[4.4.2 Use Case Diagram: 17](#_Toc165688364)

[4.4.3 Activity Diagram: 18](#_Toc165688365)

[4.4.4 Sequence diagram: 19](#_Toc165688366)

[Chapter 5: Implementation and Testing 21](#_Toc165688367)

[5.1 Software Implementation 21](#_Toc165688368)

[5.1.1 Implementation Details: 21](#_Toc165688369)

[5.1.2 Alternative Implementations Considered: 21](#_Toc165688370)

[5.1.3 Main component with UI and function: 21](#_Toc165688371)

[5.2 Software Testing 27](#_Toc165688372)

[5.2.1 Testing Techniques: 27](#_Toc165688373)

[5.2.2 Test Plan: 27](#_Toc165688374)

[5.2.3 Test Results and Analysis: 29](#_Toc165688375)

[Chapter 6: Evaluation of Results 30](#_Toc165688376)

[6.1 Effectiveness in Meeting Objectives: 30](#_Toc165688377)

[6.2 Comparison with Existing Solutions: 30](#_Toc165688378)

[6.3 Potential Areas for Further Improvement: 30](#_Toc165688379)

[Chapter 7: Conclusions 32](#_Toc165688380)

[7.1 Summary of Achievements: 32](#_Toc165688381)

[7.2 Reflections and Lessons Learned: 32](#_Toc165688382)

[7.3 Future Work: 32](#_Toc165688383)

[7.4 Conclusion: 33](#_Toc165688384)

[References 34](#_Toc165688385)

[Bibliography 35](#_Toc165688386)

# Introduction

This application is designed with the sole intention of improving the lives of students everywhere, regardless of their program of choice. Throughout the semester, courses tend to move very fast, making it far too easy for students to become disorganized. Forgetting an assignment's due date, not knowing the value of an assignment, and not knowing one's current grade in a course are just some of the problems for which this application provides solutions.

## Project topic and rationale

The Grade Tracker project is a JavaFX application designed to assist students in managing their academic coursework more effectively. The motivation behind this project stems from the common challenges faced by students in keeping track of their assignments, grades, and overall progress in various courses throughout the semester. With the fast-paced nature of academic life, students often struggle with organization, forgetting assignment due dates, and not having a clear understanding of their current grades in different courses. This application aims to address these issues by providing a centralized platform for students to manage their coursework, track grades, and monitor their progress in each course.

## Project Aims and Objectives

**Project Aims:**

* Develop a user-friendly application to help students organize their coursework efficiently.
* Provide features for tracking assignment due dates, grades, and overall course progress.
* Offer visualization tools to display course grades and progress over time.
* Provides a dynamic view of the overall progress of each course enrolled in the semester

**Project Objectives:**

* Design and implement a user interface for inputting and displaying course information.
* Develop functionality for adding, editing, and deleting assignments within each course.
* Implement grade calculation algorithms to determine current and projected grades.
* Incorporate visualization tools such as charts to visualize course progress and grade distributions.
* Provide overall semester progress and offer goal grades to achieve.
* Provide a dashboard that shows the progress dynamically with a chart of each course enrolled.

## Methodology

The project follows an iterative development approach inspired by Agile methodologies. The development process involves continuous feedback and iteration to refine features and improve user experience. Key steps in the methodology include requirements gathering, design, implementation, testing, and deployment. Additionally, the project utilizes object-oriented programming principles for modular and maintainable code.

Utilizing Object Oriented Programming methodologies (Encapsulation, Polymorphism, Inheritance) combined, this application provides an all-in-one solution to the common problems that impact all students. Start by setting a goal grade to keep track of how close or how far you are from achieving your goal grade for the course throughout the semester. Track coursework, grades received, and due dates. View your progress throughout the semester at any given time by visiting your dashboard, in here you'll be able to compare your grades in one course with your grades in another. View your current grade and projected grade in a course by visiting the course directly.

## The report structures

**Chapter 2: Background Research**

Discussing critical review of related work, and products to demonstrate a specialist understanding of the project topic, including its fundamentals and the state-of-the-art, i.e. the current developments and potential areas (i.e. “gaps”) for future development.

**Chapter 3: Requirements Analysis and Specification**

Discusses the functional and non-functional requirements of the Grade Tracker application. Discussing elicitation is what your system is required to do, not how it will do it. It also concluded with a detailed and unambiguous specification of your software requirements (*functional* and *non-functional* requirements).

**Chapter 4: Software Design:**

Details the design decisions and implementation strategies used in developing the Grade Tracker application. Describe expanded technical details of your software design that you have further elaborated on your initial design, which was drafted in the interim report. You must have very clear ideas and details of what you are going to implement in the next stage of your project process.

**Chapter 5: Implementation and testing:**

Details of how you went about realizing the specified design from the previous chapter. There should also be your consideration of alternative implementations and evidence of a justified reason for the implementation selected.

A specific, evidence-based description of the techniques, programming languages, and software tools that were used, and how they were employed in the implementation and/or in the experiment and/or data analysis.

A detailed technical description of the implemented software and/or research results should be described in this section.

Describes the testing methodologies employed to ensure the reliability and accuracy of the Grade Tracker application.

**Chapter 6: Evaluation of results:**

This chapter assesses whether the project output and results have achieved the project’s aim and objectives, or have solved the stated research problem, putting the project *in the wider context* of the chosen topical area of the project. This could be done by a comparison of your product & results with closely related products and/or related work on the same topic/subject area.

If not, then what the remaining or further work could be?

**Chapter 6: Conclusion:**

Summarizes the findings of the project, reflects on lessons learned, and proposes future enhancements for the Grade Tracker application.

This structure provides a comprehensive overview of the Grade Tracker project, from its inception to its implementation and evaluation, offering insights into its development process and outcomes.

# Background Research

## Literature review of related work

The Grade Tracker application falls within the domain of educational technology, specifically focusing on student academic management systems. A literature review reveals several existing systems and research studies addressing similar needs.

One notable example is the use of Learning Management Systems (LMS) in educational institutions. LMS platforms like Moodle and Blackboard offer features for course management, including assignment submission, grade tracking, and communication tools. However, these systems are often designed for institutional use and may lack the flexibility and user-friendliness desired by individual students.

Research studies have explored the effectiveness of academic management tools in improving student performance and organization. Studies by Jones et al. (2017), Stella Timotheus and Smith et al. (2019) found that students who used digital tools for organizing assignments and tracking grades demonstrated better academic outcomes compared to those who relied solely on traditional methods. Fikriye Kanatli also conducted a study, it was aimed to determine the impacts of Society 5.0 on digital transformation in organizations in human technology integration.

In addition to institutional LMS platforms and mobile apps, there exists a plethora of research literature and commercial products addressing student academic management needs. Studies by Williams et al. (2018), and Umar Ibrahim and Johnson et al. (2020) have investigated the impact of digital tools on student organization and time management skills. These studies highlight the importance of features such as task prioritization, deadline reminders, and progress tracking in improving student academic performance.

Furthermore, the emergence of personalized learning platforms and adaptive educational technologies has spurred innovation in academic management solutions. Adaptive learning systems like Khan Academy and Coursera offer personalized learning paths and real-time feedback to students, enabling them to track their progress and identify areas for improvement. While these platforms primarily focus on content delivery and assessment, they often integrate basic grade-tracking functionalities

## Critical evaluation of related products/solutions

Existing academic management solutions vary in terms of features, usability, and target audience. While LMS platforms offer comprehensive course management capabilities, they may be overwhelming for individual student use. Mobile apps like MyHomework and Trello provide simpler task management features but may lack the depth needed for academic tracking.

The Grade Tracker application aims to bridge the gap between comprehensive course management and user-friendly task tracking. By providing features tailored to individual student needs, such as grade visualization and goal setting, GradeTracker offers a holistic approach to academic management.

While existing academic management solutions offer valuable features, they also have limitations that GradeTracker seeks to address. Institutional LMS platforms, while comprehensive, can be cumbersome for individual student use due to their complexity and administrative focus. Moreover, they may lack customization options and fail to provide personalized insights into student performance.

On the other hand, mobile task management apps like Todoist and Wunderlist offer simplicity and ease of use but may lack specialized features for academic tracking. While suitable for general task management, these apps may not fully meet the unique requirements of academic organization, such as grade visualization and progress tracking across multiple courses.

GradeTracker aims to strike a balance between comprehensive functionality and user-friendly design. By incorporating features tailored specifically for academic management, such as grade projection, weight distribution analysis, and goal setting, GradeTracker provides students with a holistic solution for organizing their academic lives. Additionally, by being a standalone desktop application, GradeTracker offers the flexibility to customize features according to individual preferences and needs, filling the gap left by existing solutions.

## The scope of the project

The scope of the GradeTracker project encompasses the development of a standalone JavaFX application focused on individual student use. The application will allow students to input course information, track assignments and grades, visualize course progress, and set academic goals. While the initial version may lack advanced features found in institutional LMS platforms, it will prioritize simplicity, usability, and personalized tracking.

GradeTracker's scope encompasses the development of a standalone desktop application designed to assist students in managing their academic responsibilities effectively. The project focuses on addressing the following key aspects:

**Grade Management:** GradeTracker allows students to input and track their grades for individual assignments, tests, and projects across multiple courses. It provides a centralized platform for organizing grade-related information, including scores, weights, and due dates.

**Visualization and Analysis:** The application offers visualization tools to help students understand their academic progress. This includes graphical representations of grade distributions, projected grades, and goal attainment. By providing visual feedback, GradeTracker enables students to identify trends, strengths, and areas for improvement.

**Goal Setting and Monitoring**: GradeTracker empowers students to set academic goals and monitor their progress toward achieving them. Users can establish target grades for courses and track their performance relative to these goals. With the dashboard, the student can see their stats for each course enrolled and check the overall progress of the semester. By this student can easily track and work on their assignment labs, and quizzes to achieve the desired goal setup at the initial stage.

## Review and justification of theories/models/development platforms/tools selected for use in the project

The GradeTracker application utilizes JavaFX for its graphical user interface (GUI) development. JavaFX was chosen for its platform-independent nature, rich set of UI controls, and compatibility with Java programming, making it well-suited for developing desktop applications.

Object-oriented programming (OOP) principles are employed throughout the project to ensure modularity, maintainability, and extensibility of the codebase. By organizing the application into classes and objects, the code remains scalable and adaptable to future enhancements.

The GradeTracker project draws upon existing research and solutions in academic management while offering a tailored approach for individual student use. The choice of JavaFX and OOP principles aligns with the project's goals of user-friendliness, flexibility, and maintainability.

# Requirements Analysis and Specification or Problem Analysis and Specification

In this chapter, we delve into the process of eliciting and specifying the requirements for the application. By comprehensively analyzing the needs of users and stakeholders, we aim to define the functional and non-functional requirements that will guide the development of the software. Additionally, this chapter provides a detailed specification of the software requirements, including use cases and acceptance criteria.

## Requirement Elicitation

The requirement elicitation process involves gathering and analyzing information from various stakeholders to identify the needs and expectations for the GradeTracker application. This includes conducting interviews, surveys, and workshops with students, educators, and administrators to understand their pain points and requirements. After gathering and analyzing the information we come to the point that digital systems have a lot of impact on students. A student who is willing to achieve a good score and wants to get their desired grade can use the Gradetracker app and it will help them a lot. By this, they can see what needs to be done in the next quiz, assignment, or lab and can monitor each course's progress and work accordingly. Moreover, the app will show them their average score work with a beautiful and dynamic view from the dashboard. This will help them grow their grade and achieve their desired grade throughout the semester. We come to a point where the development of the GradeTracker app for the students must minimize their planning and let the student consume time for working on grades rather than planning for it.

## Analysis and Specification

Once the requirements have been gathered, they are analyzed and translated into detailed specifications. This phase involves categorizing requirements into functional and non-functional categories and documenting them using appropriate techniques such as use cases, user stories, and requirements traceability matrices. Each requirement is carefully analyzed to ensure clarity, completeness, and consistency.

## Functional Requirements

Functional requirements define the specific behavior and functionality of the GradeTracker application. These requirements describe the actions and tasks that users should be able to perform within the application. Examples of functional requirements include:

**Goal Grade Management:** The user will set the desired goal grade for the course and can update that goal later.

**Grade Tracking:** The application should allow users to input and track their grades for various courses and assignments.

**Course Management:** Users should be able to add, edit, and delete courses from their dashboard, including specifying course details such as name, instructor, and credit hours.

**Assignment Tracking:** Users should be able to add, edit, and delete assignments within each course, including recording assignment details such as due dates, weights, and scores.

**View Semester progress:** Users should be able to view their stats from the dashboard and check the progress of the projected grade.

## Non-Functional Requirements

Non-functional requirements specify the quality attributes and constraints that the GradeTracker application must adhere to. These requirements address aspects such as performance, security, usability, and scalability. Examples of non-functional requirements include:

**Performance:** The application should respond to user interactions within 2 seconds under normal load conditions.

**Usability:** The user interface should be intuitive and user-friendly, with clear navigation and informative feedback messages.

**Scalability:** The application should be able to handle a large number of concurrent users without experiencing degradation in performance.

## Use Cases

Use cases provide a detailed description of how users interact with the GradeTracker application to achieve specific goals or tasks. Each use case describes a scenario from the perspective of a user and outlines the steps involved in completing the task. Examples of use cases include:

**Use Case 1: Viewing Course Grades**

Actor: Student

Description: This use case allows students to view their grades for a specific course.

Preconditions: The student must add one course to the GradeTracker application.

Basic Flow:

* The student selects the desired course from their dashboard.
* The application retrieves and displays the grades for assignments, tests, and other assessments within the selected course.

Alternative Flow:

* If there are no grades recorded for the course, the application displays a message indicating that there are no grades available.

**Use Case 2: Adding an Assignment**

Actor: Student

Description: This use case allows students to add a new assignment to a course.

Preconditions: The student must have at least one course added to their dashboard.

Basic Flow:

* The student navigates to the course where they want to add the assignment.
* The student selects the option to add a new assignment.
* The student enters the details of the assignment, including the assignment name, due date, and weight.
* The application validates the entered information and saves the assignment to the course.

Alternative Flow:

* If the student tries to add an assignment without specifying mandatory details such as the assignment name or due date, the application prompts them to provide the missing information.

**Use Case 3: Setting Goal Grade**

Actor: Student

Description: This use case allows students to set a goal grade for a course.

Preconditions: The student must have at least one course added to their dashboard.

Basic Flow:

* The student navigates to the course where they want to set a goal grade.
* The student selects the option to set a goal grade.
* The student enters the desired goal grade for the course.
* The application validates the entered grade and saves it for the course.

Alternative Flow:

* If the student enters an invalid grade (e.g., a grade outside the valid range of 0-100), the application displays an error message and prompts the student to enter a valid grade.

## Acceptance Criteria

Acceptance criteria define the conditions that must be met for each requirement to be considered successfully implemented. These criteria serve as a benchmark for testing and validation purposes, ensuring that the GradeTracker application meets the expectations of users and stakeholders. Examples of acceptance criteria include:

Grade Calculation: The application should accurately calculate the overall grade for each course based on the weighted average of assignments and exams.

GUI: The application should have a good and user-friendly graphical user interface so that the user can interact with the application easily.

# Software Design

## User-Interface Design

For the GradeTracker application, the user interface design plays a critical role in providing a seamless experience for students to manage their courses, assignments, and grades effectively. The design principles focus on clarity, simplicity, and accessibility to ensure users can easily navigate the application and access essential features.

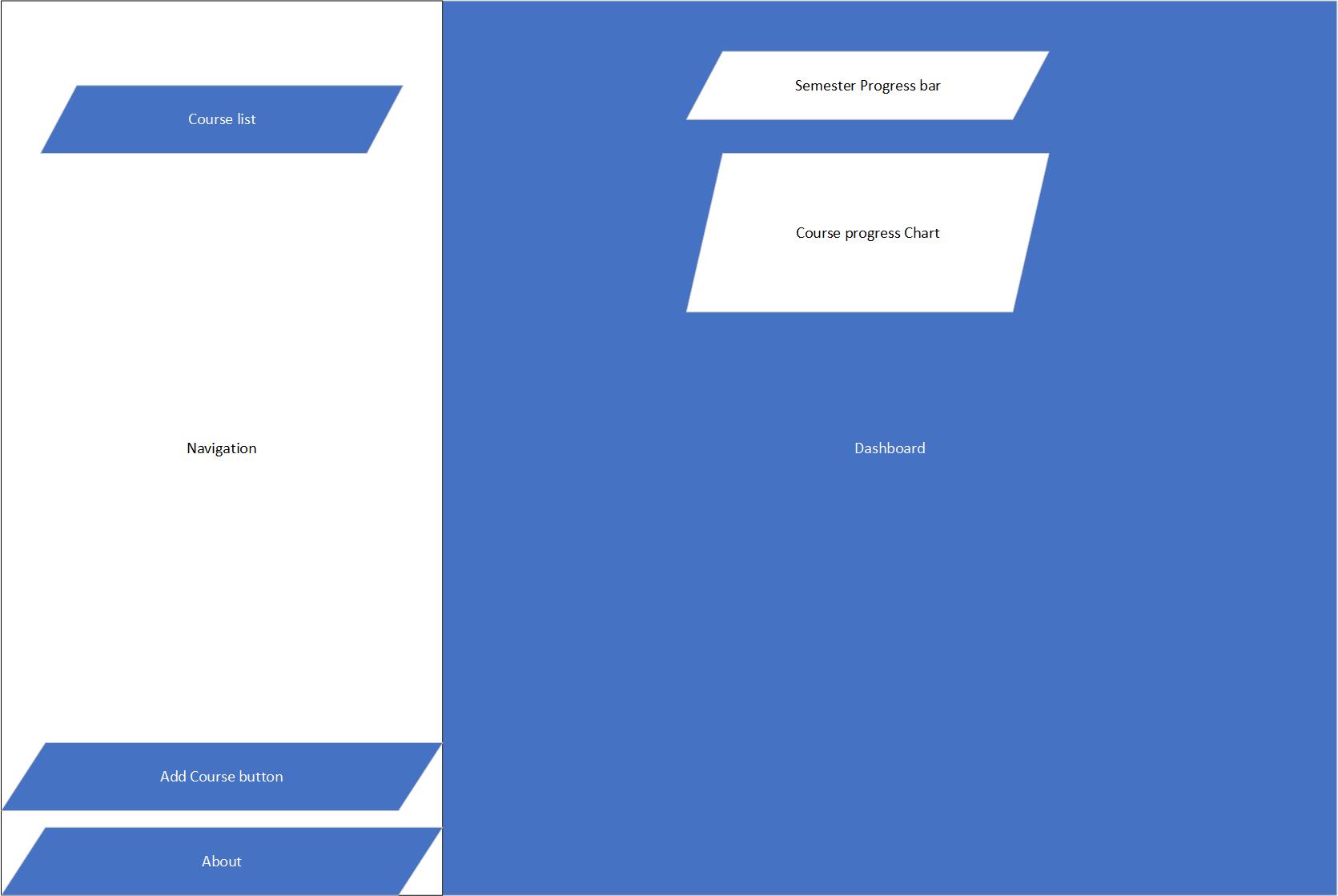
### Mock-up Screens:

**Dashboard:**

The dashboard serves as the central hub of the GradeTracker application, offering an at-a-glance overview of all enrolled courses and their corresponding grades. Key features of the dashboard mock-up include:

Semester Summary Chart: The first thing students will notice is a semester summary chart on the right panel of the dashboard which will show all the course goal grades and the projected grades achieved till yet. Above that the overall semester progress is shown.

Navigation Menu: A side navigation menu provides quick access to different sections of the application, including the course list, course edit, course delete about and course add button.



**Course Details:**

The course details screen provides a comprehensive view of a specific course, allowing students to delve deeper into their grades and assignments. Key elements of the course details mock-up include:

Course Information: Basic details about the course, including the course name, instructor, and goal grade.

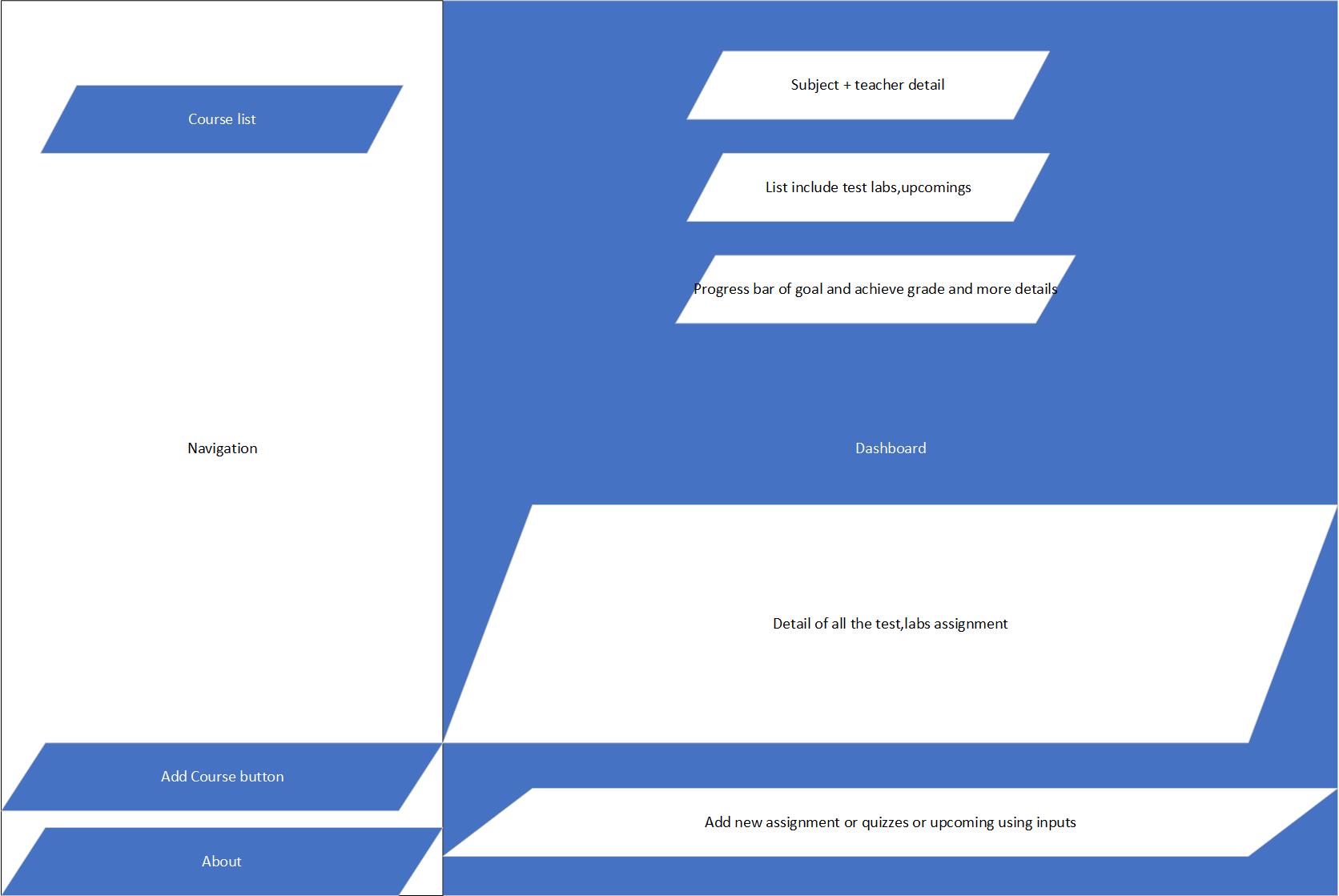
Assignment List: A list of all assignments associated with the course, displaying the assignment name, due date, weight, and grade received.

Test List: A list of all tests associated with the course, displaying the assignment name, due date, weight, and grade received.

Labs List: A list of all labs associated with the course, displaying the assignment name, due date, weight, and grade received.

Upcoming Event: A list of all Upcoming events associated with the course is displayed with the date so the student knows very important due dates.

Overall Course Progress: Visual representations such as progress bars illustrate the student's overall progress in the course.

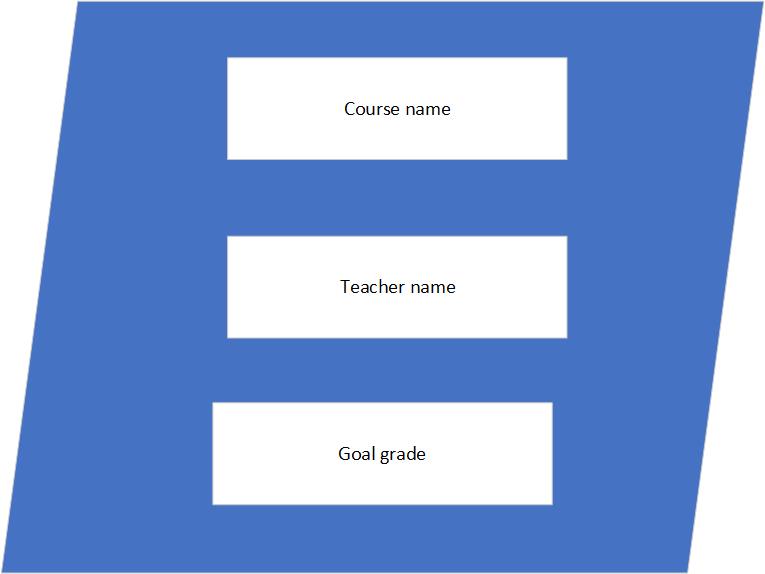


**Add Assignment, test, and lab marks:** An input bar that will be used by the student to easily add upcoming events and marks achieved in previous quizzes or assignments after this the app will calculate the grade and compare it with the goal grade and display it on the dashboard.

**Set Goal Grade:**

Setting a goal grade empowers students to establish clear objectives for their courses and track their progress toward achieving them. The set goal grade screen features:

Goal Grade Input: A user-friendly interface for entering the desired goal grade for the course, allowing students to specify their target grade.



## Database Tables’ Structure Design

The database tables for the GradeTracker application were designed to efficiently store and manage information related to courses, assignments, grades, and user accounts. The structure includes tables for courses, assignments, grades, and user authentication.

### Database Tables:

Courses: Stores information about each enrolled course, including course name, instructor, and goal grade.

Assignments: Contains details of assignments for each course, such as assignment name, due date, and weight.

Quiz: Contains details of quizzes for each course, such as quiz name, due date, and weight.

Labs: Contains details of labs for each course, such as lab name, due date, and weight.

Grades: Stores individual grades for assignments, linked to the corresponding course and assignment.

## 4Main Components of the Software Architecture

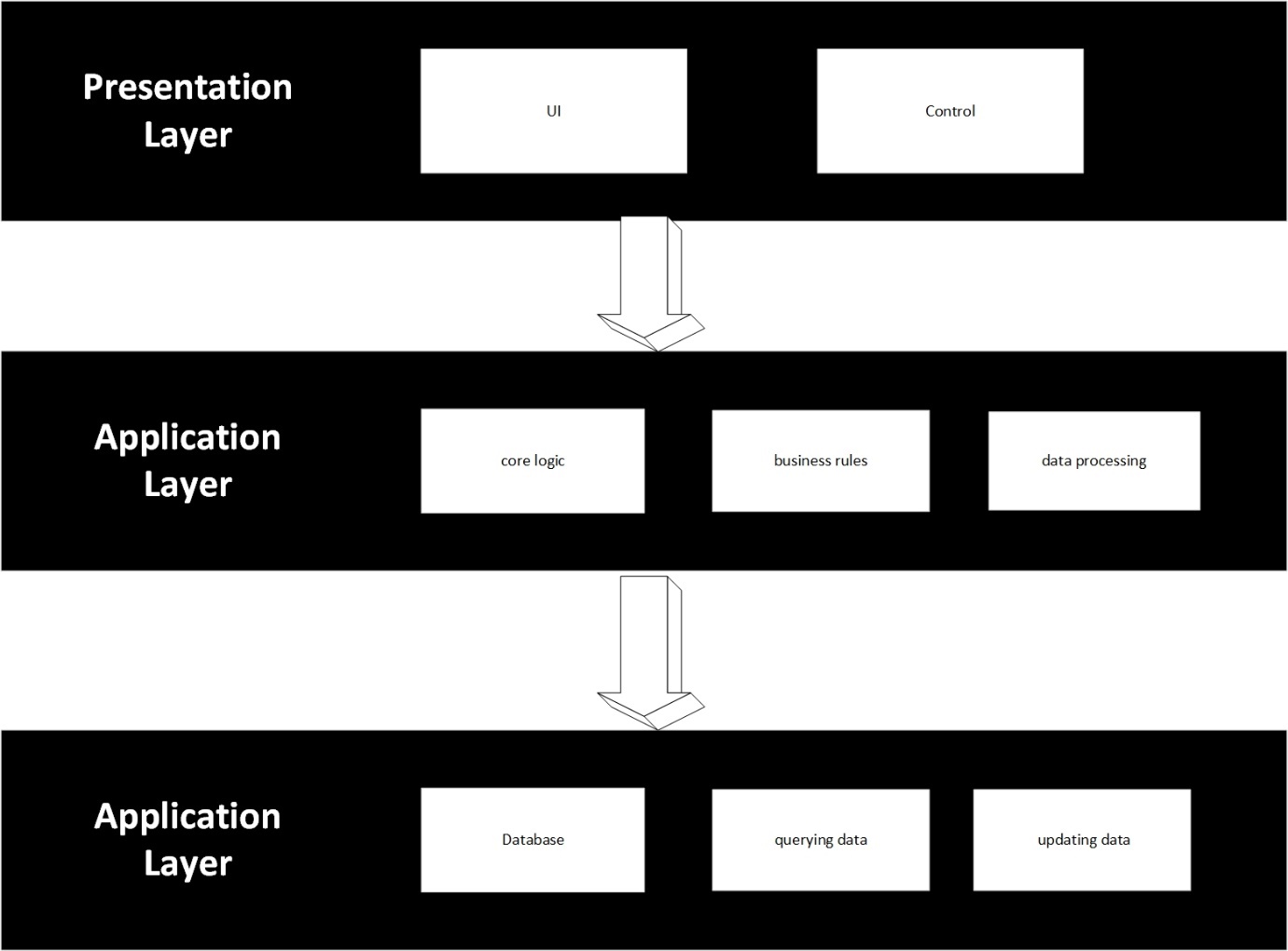
The software architecture of GradeTracker follows a layered architecture pattern, consisting of presentation, application, and data access layers.

### Main Components:

Presentation Layer: Responsible for handling user interactions and displaying information to the user. This layer includes user interface components such as screens, forms, and controls.

Application Layer: Implements the core logic of the application, including business rules and data processing. It interacts with the presentation layer to handle user requests and with the data access layer to retrieve and store data.

Data Access Layer: Manages interactions with the database, including querying and updating data. It provides an abstraction layer between the application layer and the database to ensure separation of concerns and maintainability.



## Detailed Software Class Design

The detailed software class design outlines the public interfaces of key classes in the GradeTracker application, including class properties and methods without implementation details.

**Class Design:**

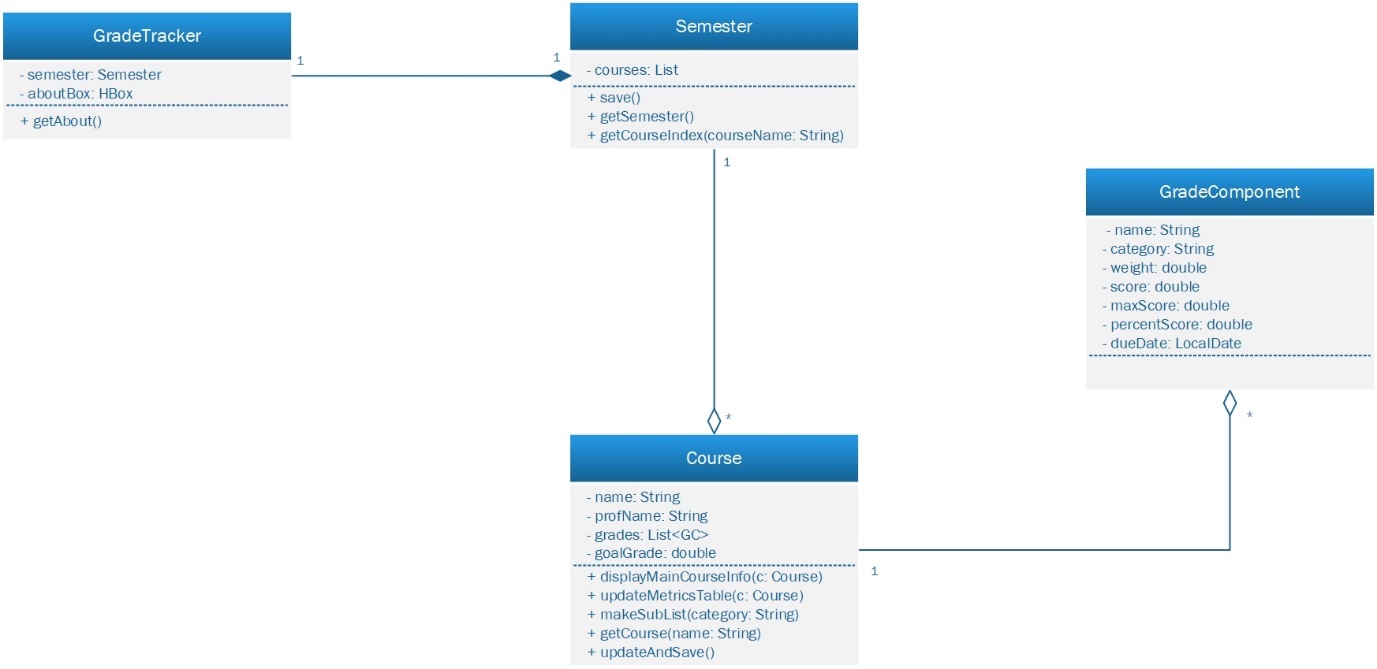
Course: Represents a course enrolled by the student, with properties such as name, instructor, and goal grade.

Assignment: Represents an assignment within a course, with properties including name, due date, weight, and associated grades.

Grade: Represents an individual grade for an assignment, including the score achieved and the maximum possible score.

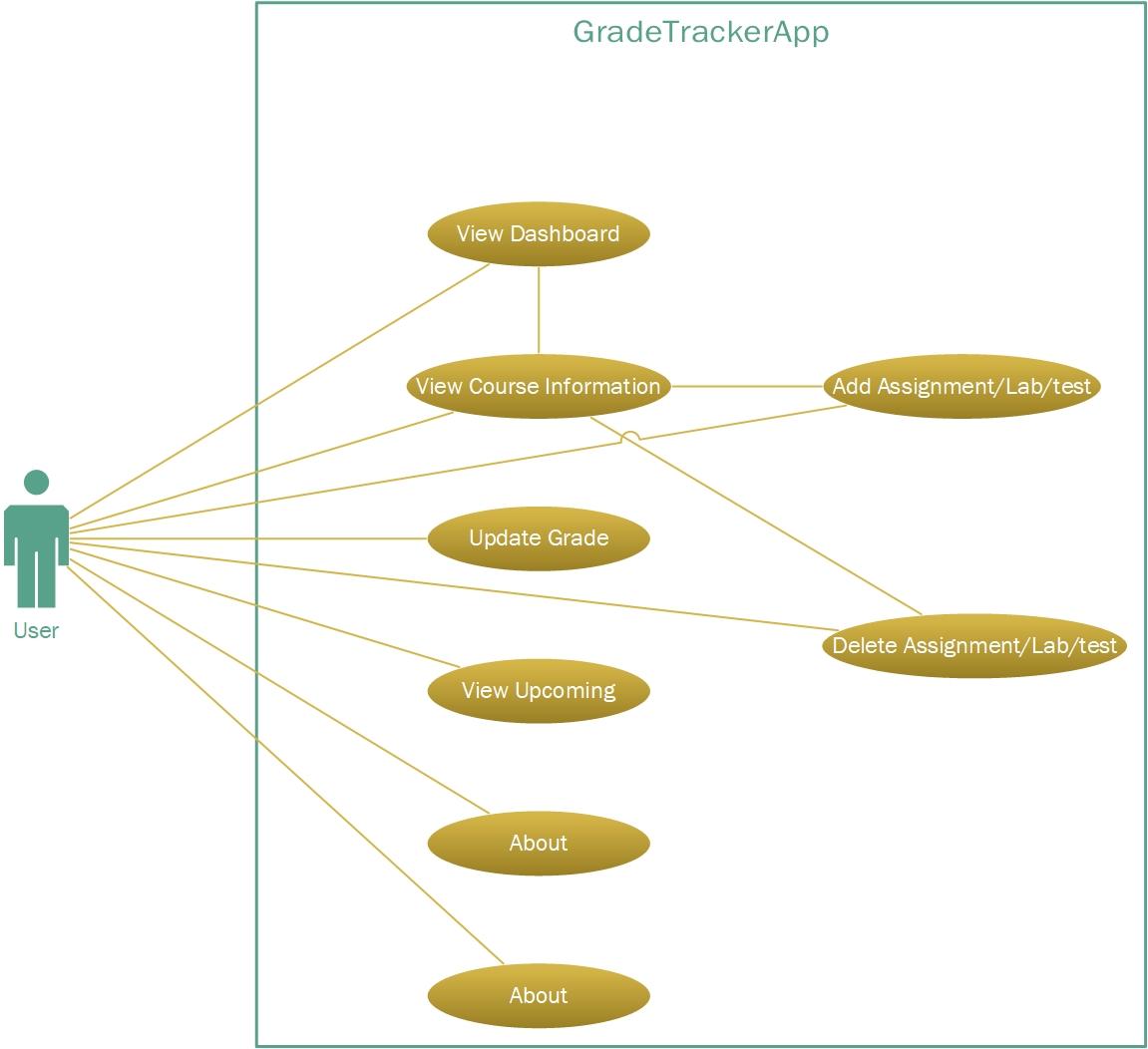
User: Manages user authentication and authorization, including methods for login, logout, and user registration.

### Class Diagram:



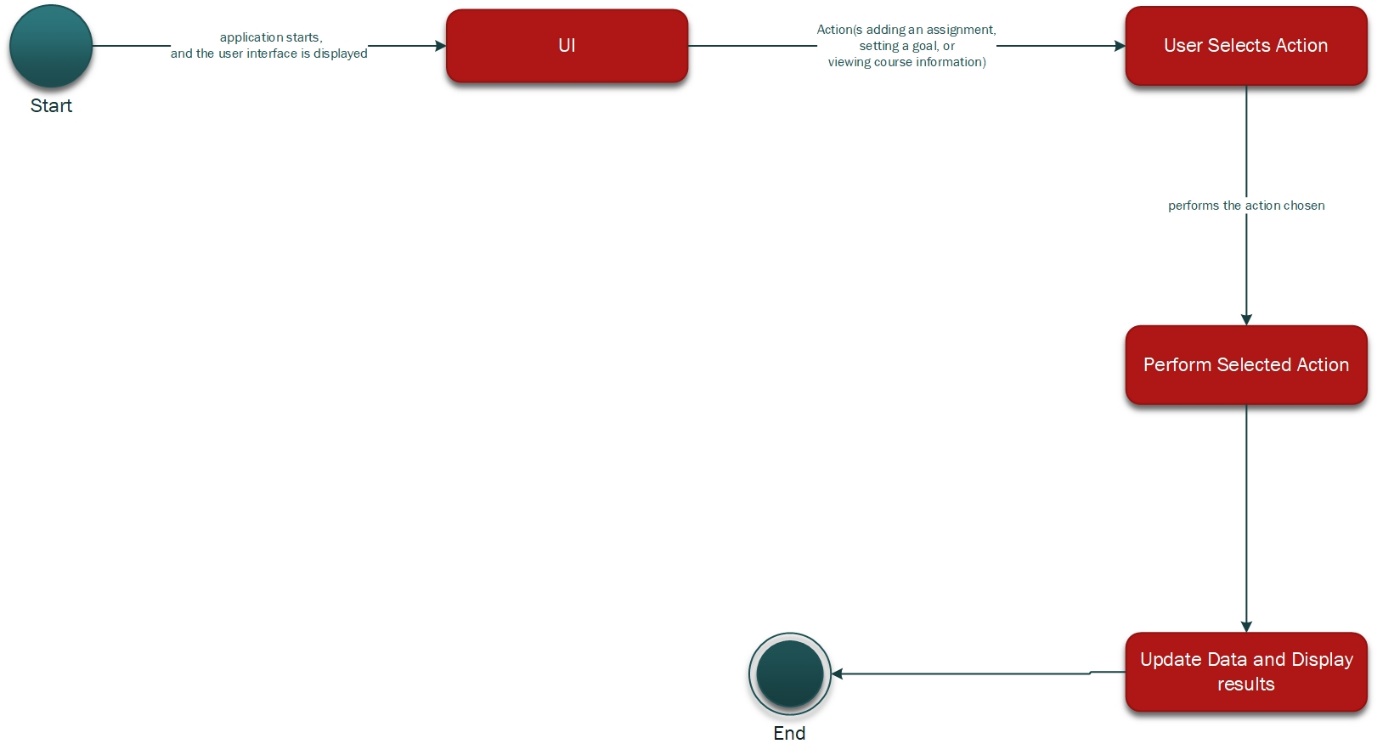
The provided class diagram illustrates the structure of a software application called "GradeTracker," designed to assist students in organizing and monitoring their academic progress throughout a semester. It consists of classes representing the application itself, semesters, courses, and grade components. These classes establish relationships, including composition and aggregation, to manage the hierarchy of data. Additionally, the diagram depicts various methods and functionalities for displaying course information, updating metrics tables, creating charts, and handling user interactions such as adding, editing, and deleting grade components.

### Use Case Diagram:



The use case diagram for the GradeTracker application illustrates the various functionalities available to the user. Users can interact with the system through actions such as viewing the dashboard, accessing course information, updating grades, adding or deleting assignments, setting goal grades, viewing upcoming assignments, and accessing the About page. Each of these actions represents a distinct use case that fulfills specific user needs within the application. The diagram showcases the relationships between different functionalities and how they contribute to the overall user experience, providing a clear overview of the system's capabilities.s

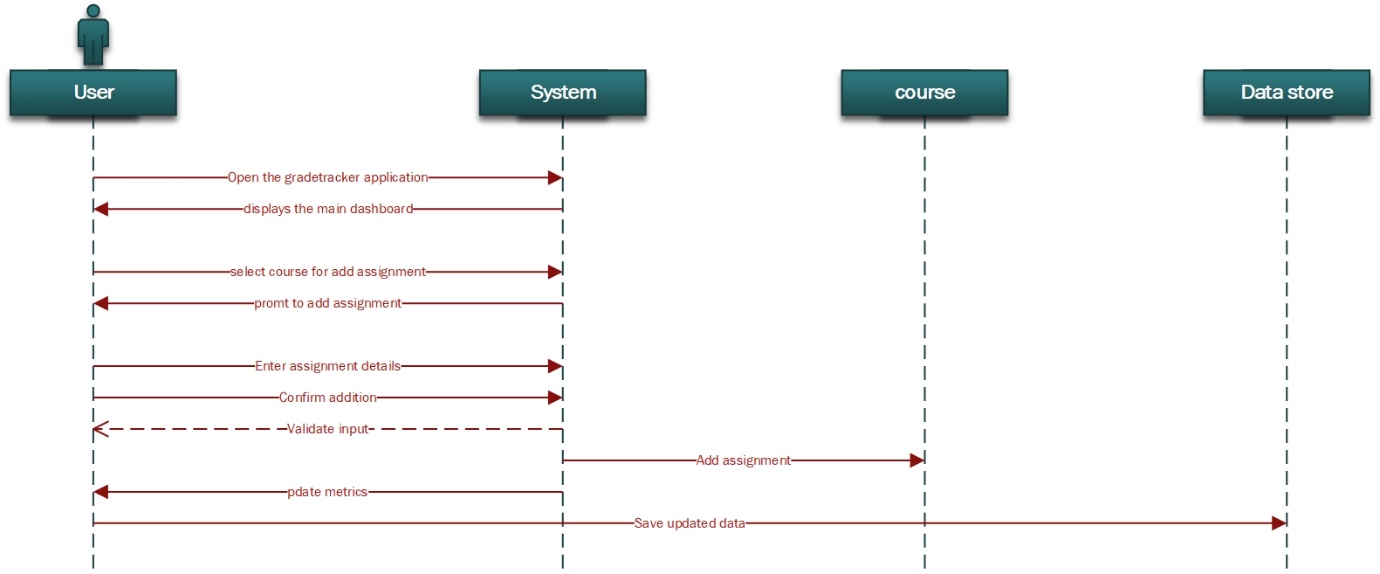
### Activity Diagram:



The activity diagram illustrates the flow of actions within the application. It begins with the initiation of the application, followed by the user selecting an action from the available menu options. Upon selection, the application performs the chosen action, updates the data accordingly, and displays the results to the user. Finally, the activity concludes, leaving the application open for further interactions or allowing the user to exit if desired.

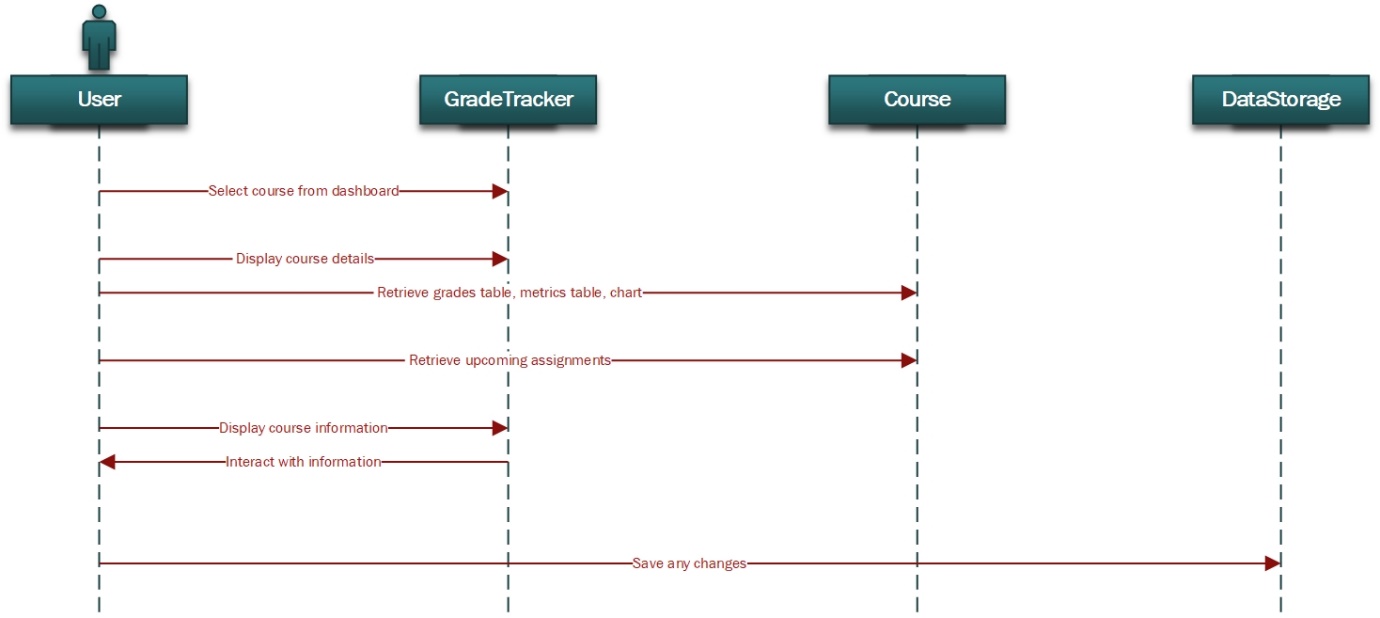
### Sequence diagram:

#### Add Assignment use case



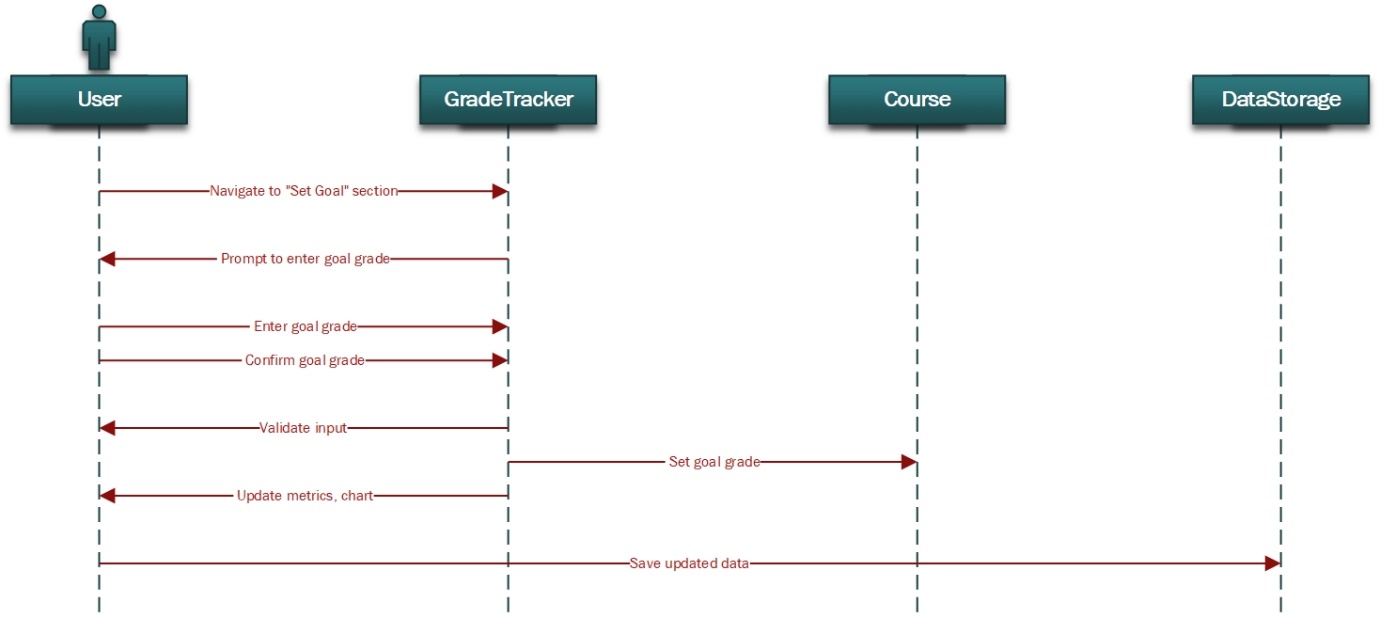
* The user initiates the process by selecting the "Add Assignment" option in the application.
* The application displays a form where the user enters details such as assignment name, weight, score, max score, and due date.
* The user fills out the form and submits it.
* The application validates the entered data.
* If the data is valid, the assignment is added to the course data, and the updated course information is displayed to the user.
* If there are any validation errors, appropriate error messages are shown to the user, and the user is prompted to correct them.

#### View Course Information



* The user navigates to the "View Course Information" section of the application.
* The application retrieves the relevant data for the selected course, including grades, metrics, and upcoming assignments.
* The retrieved data is displayed to the user in the application interface.
* The user interacts with the displayed data, potentially modifying grades or assignment details.
* Any changes made by the user are saved to the data storage, ensuring persistence across sessions.

#### Set goal



* The user selects the "Set Goal" option from the application menu.
* The application presents a form where the user can input their desired grade for the course.
* The user enters the desired grade and confirms the selection.
* The application updates the course metrics based on the entered goal grade.
* The updated course metrics are displayed to the user, showing the progress towards the goal grade.

# Implementation and Testing

## Software Implementation

The implementation phase of the GradeTracker application involved translating the design specifications into functional software. Here, I'll provide a detailed overview of how the application was realized, including the programming languages, tools, and techniques used. Also, I will provide the main component of the application with code to give an overview of the application.

### Implementation Details:

#### Programming Language: Java

Java was chosen as the primary programming language for its platform independence, robustness, and extensive libraries for building graphical user interfaces (GUIs) using JavaFX.

#### Integrated Development Environment (IDE): IntelliJ IDEA

IntelliJ IDEA was selected as the IDE for its comprehensive features, including code editing, debugging, and version control integration, which streamlined the development process.

#### JavaFX for GUI Development

JavaFX was utilized for creating the user interface components of the GradeTracker application. It provided a rich set of APIs for designing interactive and visually appealing GUIs.

#### Model-View-Controller (MVC) Architecture

The application was structured using the MVC design pattern, which separates the application into three interconnected components: Model, View, and Controller. This facilitated modularity, code reusability, and easier maintenance.

#### Database Integration

For data persistence, SQLite was used as the embedded database engine. It offered lightweight, self-contained, and serverless database functionality, ideal for storing course and assignment data locally.

#### JavaFX Scene Builder

JavaFX Scene Builder was employed to design the application's UI layout visually. It allowed for drag-and-drop placement of UI components, simplifying the GUI design process.

#### Version Control with Git

Git was utilized for version control, enabling collaborative development and tracking of code changes throughout the project lifecycle.

### Alternative Implementations Considered:

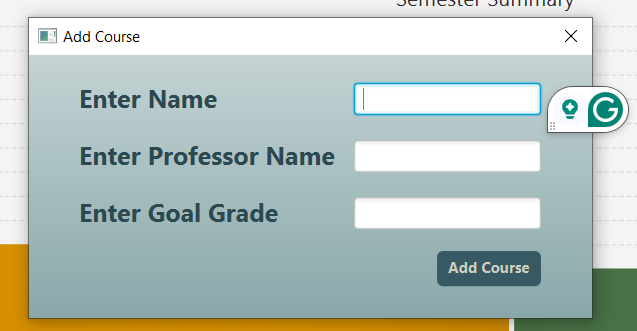
#### Web-Based Application

Initially, a web-based approach using HTML, CSS, JavaScript, and a backend framework like Spring Boot was considered. However, it was decided to opt for a desktop application for better performance and offline accessibility.

### Main component with UI and function:

#### Add course & Update course:

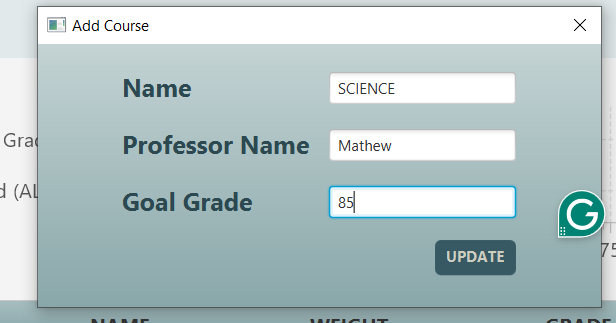
**Add course**



**Description:**

The Add course is the initial step of the application. As we land on the main dashboard of the app the first thing students or users will do is to add a course of the semester to the course list with the course name, professor name, and their desired goal grade for the course. After clicking the Add Course button the course will be added to the application with goal grades. Now users can add delete or update assignments, quizzes labs, etc to the course and can view stats of the course from the course dashboard.

**Update Course:**



**Description:**

Now if students want to change the course name the course professor name or the goal grade for any particular course then the app provides this functionality. Sometimes the professor changes or the student wants to achieve more from the course to get an overall good semester grade, so they can do it easily by just selecting a course from the course list and can edit it any time the user wants.

**Code:**

Button addCourseBtn = makeBtn( "ADD COURSE", e -> makeCourseWindow() );

addCourseBtn.setId("addCourseBtn");

addCourseBtn.setMinSize(125,30);

addCourseBtn.setTooltip(makeToolTip("Add Course"));

private void makeCourseWindow() {

GridPane gp = makeWindowGridPane(true);

TextField nameText = new TextField();

gp.add(nameText, 1, 0);

TextField profText = new TextField();

gp.add(profText, 1, 1);

TextField goalText = new TextField();

gp.add(goalText, 1, 2);

Button addBtn = new Button("Add Course");

addBtn.setDefaultButton(true);

addBtn.setId("windowBtn");

GridPane.setHalignment(addBtn, HPos.RIGHT);

gp.add(addBtn, 1, 3);

Stage courseAddWindow = makeWindowStage(gp, "Add Course");

addBtn.setOnAction((e -> {

try {

if (!semester.checkNameExists(nameText.getText()))

throw new CustomException("Course Name already exists", "Duplicate Course Name");

Course newCourse = new Course(nameText.getText(), profText.getText(),

Double.parseDouble(goalText.getText()));

semester.getSemester().add(newCourse);

semester.save();

topCourses.getChildren().add(makeCourseHBox(newCourse));

displayMainCourseInfo(newCourse); // After a course is made display its page

updateCourseChart(newCourse, "ALL", newCourse.getGrades());

courseAddWindow.close(); // Close window after adding a course

} catch (NumberFormatException ex) {

new CustomException("Goal Grade must be a number", "Bad Goal Grade input").DisplayErrMessage();

} catch (CustomException ex) {

ex.DisplayErrMessage();

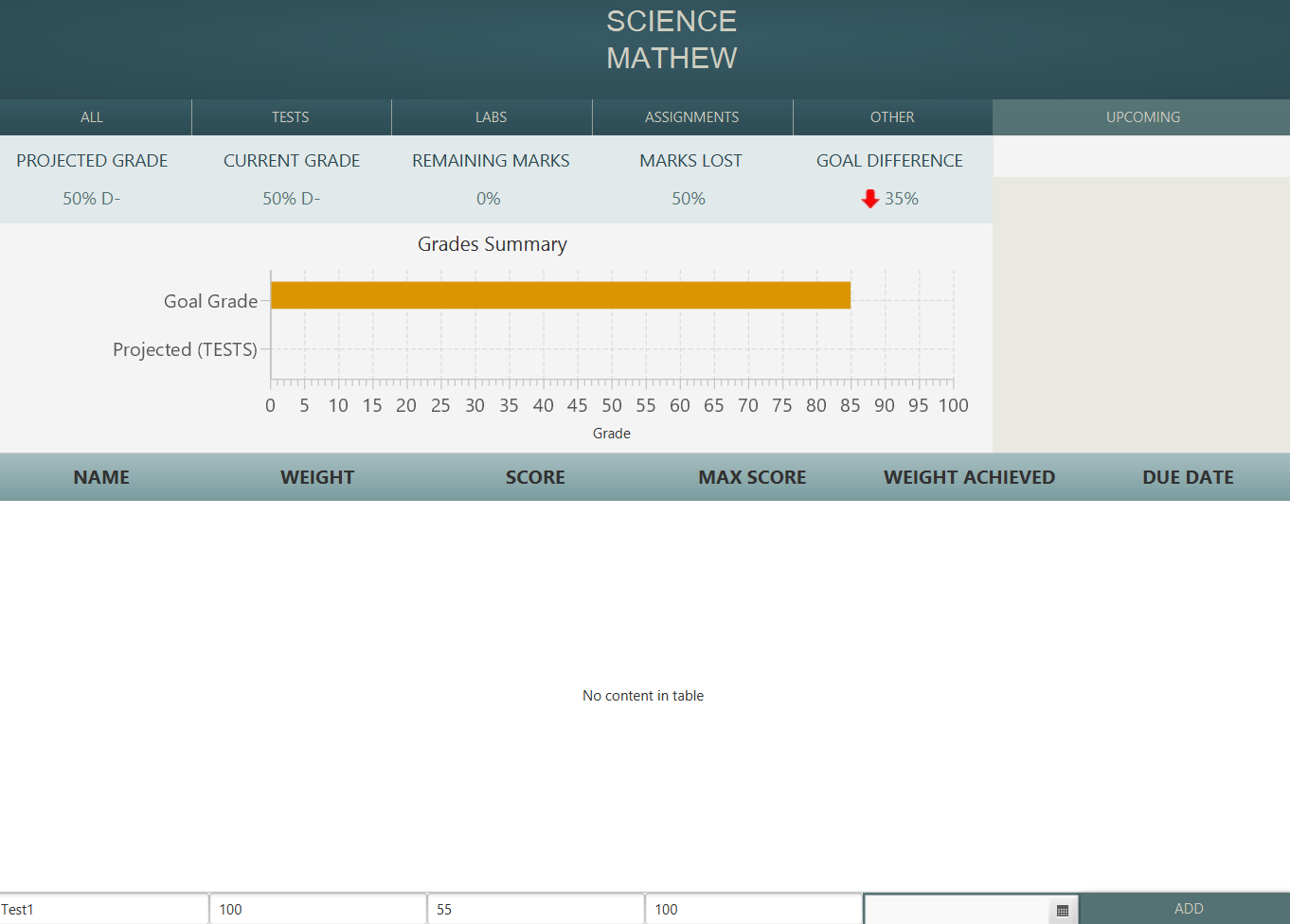
}

}));

courseAddWindow.show();

}

#### Add Course Data:





**Description:**

After adding the course to the course list now the main thing that is to be done is to add data i.e. Assignment details, test details, labs details, or others to the course. So the application calculates it and makes a chart and helps students for the upcoming. This can be done in the course as on the right panel to the bottom an input field with buttons is given through which we can add delete edit any assignment or tests for the course. This is one of the main components of the application and I make it look very simple and easy to use for the students.

**Code:**

if(canAdd) {

TextField nameInput = makeTableModField("NAME", "nameInput");

TextField scoreInput = makeTableModField("SCORE", "scoreInput");

TextField maxScoreInput = makeTableModField("MAX SCORE", "maxScoreInput");

TextField weightInput = makeTableModField("WEIGHT", "weightInput");

DatePicker dueDatePicker = new DatePicker();

dueDatePicker.setId("dueDatePicker");

dueDatePicker.setMinWidth(rightPanel.getWidth()/6);

dueDatePicker.getEditor().setDisable(true);

dueDatePicker.setPromptText("DUE DATE");

Button addGradeBtn = makeBtn("ADD", e -> {

try {

GradeComponent gc = new GradeComponent();

gc.setName(nameInput.getText());

if(!weightInput.getText().isEmpty()) gc.setWeight(Double.parseDouble(weightInput.getText()));

else gc.setWeight(Double.NaN);

if(!scoreInput.getText().isEmpty()) gc.setScore(Double.parseDouble(scoreInput.getText()));

else gc.setScore(Double.NaN);

if(!maxScoreInput.getText().isEmpty()) gc.setMaxScore(Double.parseDouble(maxScoreInput.getText()));

else gc.setMaxScore(Double.NaN);

if(dueDatePicker.getValue() != null) {

gc.setDueDate(dueDatePicker.getValue());

}

checkWeightFull(getCourse(courseName).getTotalWeight(), gc.getWeight());

gc.setCategory(category);

getCourse(courseName).getGrades().add(gc);

updateAndSave(courseName, category);

gradesTable.getItems().add(gc);

nameInput.clear(); weightInput.clear(); scoreInput.clear(); maxScoreInput.clear(); dueDatePicker.setValue(null);

} catch (NumberFormatException ex) { new CustomException("Weight, Score and Max Score must be numbers", "Non numbers input").DisplayErrMessage();

} catch (CustomException ex) { ex.DisplayErrMessage(); }

});

addGradeBtn.setDefaultButton(true);

addGradeBtn.setId("addGradeBtn");

addGradeBtn.setMinWidth(rightPanel.getWidth()/6);

HBox addTableRow = new HBox();

addTableRow.setMinWidth(rightPanel.getWidth());

addTableRow.getChildren().addAll(nameInput, weightInput, scoreInput, maxScoreInput, dueDatePicker, addGradeBtn);

tableHolder.getChildren().addAll(addTableRow);

}

rightPanel.setBottom(tableHolder);

}

## Software Testing

The testing phase of the GradeTracker application aimed to ensure that the implemented software met the specified requirements and functioned as intended. Various testing techniques were employed to validate the correctness, reliability, and usability of the application.

### Testing Techniques:

Unit Testing

JUnit was used for unit testing individual components and methods of the application. Test cases were designed to verify the correctness of critical functionalities, such as adding courses, and assignments, and calculating grades.

Integration Testing

Integration testing was performed to validate the interaction between different modules and components of the application. This ensured that data flowed correctly between the UI, controller, and model layers.

User Acceptance Testing (UAT)

UAT involved end-users testing the application to assess its usability, intuitiveness, and overall user experience. Feedback from UAT sessions was collected and incorporated to enhance the application's usability.

Performance Testing

Performance testing was conducted to evaluate the application's responsiveness and resource utilization under various load conditions. This included measuring the time taken to load courses and assignments, as well as memory usage.

### Test Plan:

1. **Introduction**

* *Purpose of the Test Plan*

The purpose of this test plan is to outline the approach, objectives, and scope of testing for the GradeTracker application. It aims to ensure the quality and reliability of the application by systematically testing its functionality, user interface, data integrity, and performance. By following this plan, we can identify any defects or issues in the application and ensure that it meets the specified requirements before deployment.

* *Overview of the GradeTracker Application*

GradeTracker is a JavaFX-based application designed to assist students in tracking their academic progress and managing their course grades. The application provides features such as:

* Dashboard: An overview of all enrolled courses and their respective grades.
* Course Management: Add, edit, and delete courses along with their assignments.
* Assignment Tracking: Add and update assignments for each course.
* Grade Calculation: Automatically calculate overall grades based on assignment scores and weightage.
* Goal Setting: Set and track goal grades for individual courses.

1. **Test Objectives**

* To ensure the functionality of key features
* To validate user interface behavior
* To verify data integrity and accuracy
* To assess system performance under normal and stress conditions

1. **Test Scope**

* Functional Testing
* User Interface Testing
* Data Integrity Testing
* Performance Testing

1. **Testing Approach**

* Manual testing will be conducted initially for all test cases

1. **Test Environment**

* Operating System: Windows 10
* Development Environment: JavaFX, JDK 11
* Test Devices: Desktop and Laptop Computers

1. **Test cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | Add a new course with valid information | Course should be added to the dashboard | Course is successfully added to the dashboard | Pass |
| 2 | Add a new course with missing information | Error message should prompts user to fill in required fields | Error message prompts user to fill in required fields | Pass |
| 3 | Edit an existing course | Changes to course details should saved | Changes to course details are saved | Pass |
| 4 | Delete an existing course | Course should be removed from the dashboard | Course is removed from the dashboard | Pass |
| 5 | Add a new assignment to a course | Assignment should be added to the course | Assignment is added to the course | Pass |
| 6 | Delete an existing assignment | Changes to assignment details should saved | Changes to assignment details are saved | Pass |
| 7 | Set a goal grade for a course | Goal grade should saved and displayed for the course | Goal grade is saved and displayed for the course | Pass |
| 8 | Calculate current grade for a course | Current grade should calculated correctly | Current grade is calculated correctly | Pass |
| 9 | View dashboard with multiple courses and assignments | All courses and assignments are to displayed correctly | All courses and assignments are displayed correctly | Pass |
| 10 | Check application responsiveness on different screen resolutions | Application adapts and displays correctly | Application adapts and displays correctly | Pass |
| 11 | Verify data persistence | Data remains intact after application restart | Data remains intact after application restart | Pass |
| 12 | Check error handling for invalid user inputs | Appropriate error messages are displayed | Appropriate error messages are displayed | Pass |

### Test Results and Analysis:

Overall, the GradeTracker application passed the testing phase successfully, meeting the specified requirements and demonstrating robust functionality.

The unit tests achieved high code coverage, validating the correctness of core functionalities.

Integration testing confirmed seamless communication between different components, ensuring data integrity and consistency.

User acceptance testing feedback highlighted areas for improvement, such as UI enhancements and additional features for better user engagement.

Performance testing results indicated satisfactory responsiveness and minimal resource overhead, ensuring optimal user experience even on low-end hardware configurations.

# Evaluation of Results

The evaluation of the GradeTracker application involves assessing its effectiveness in meeting the project's objectives, solving the identified problem, and its relevance in the broader context of academic progress tracking tools. This chapter critically examines the features, functionality, and performance of GradeTracker, compares it with existing solutions, and discusses potential areas for further improvement and development.

## Effectiveness in Meeting Objectives:

GradeTracker was developed with the aim of providing students with a user-friendly tool to track their academic progress, manage course grades, and stay organized throughout the semester. The application's objectives included:

**Streamlining Grade Tracking:** GradeTracker effectively streamlines the process of recording and monitoring grades for multiple courses. The dashboard provides a clear overview of all enrolled courses and their respective grades, allowing users to quickly assess their academic standing.

**Assignment Management:** The assignment tracking feature enables students to add, edit, and delete assignments for each course, ensuring that they stay updated on upcoming deadlines and completed tasks.

**Goal Setting:** GradeTracker allows users to set goal grades for individual courses, helping them stay motivated and focused on achieving their academic targets.

## Comparison with Existing Solutions:

GradeTracker offers several advantages over traditional methods of grade tracking, such as manual spreadsheets or paper-based planners. Unlike spreadsheets, GradeTracker provides a user-friendly interface tailored specifically for academic use, with features like goal setting and assignment tracking integrated into a single platform. Additionally, GradeTracker offers real-time grade calculation and visualization, allowing users to monitor their progress more effectively compared to static spreadsheets.

Compared to other academic progress tracking tools, GradeTracker distinguishes itself with its simplicity, ease of use, and focus on essential features. While some existing solutions may offer more advanced features or integrations, GradeTracker prioritizes accessibility and intuitiveness, making it suitable for a wide range of users, including students with varying technical proficiency.

## Potential Areas for Further Improvement:

While GradeTracker fulfills its core objectives effectively, there are several areas for further improvement and development:

**Enhanced Reporting:** Implementing more robust reporting features, such as detailed grade analytics and progress charts, could provide users with deeper insights into their academic performance.

**Integration with Learning Management Systems (LMS):** Integrating GradeTracker with popular LMS platforms used by educational institutions could streamline data import/export and provide a seamless experience for users.

**Mobile Application:** Developing a mobile version of GradeTracker would increase accessibility, allowing users to track their grades and manage assignments on-the-go.

**Feedback Mechanism:** Implementing a feedback mechanism within GradeTracker would enable users to provide suggestions, report issues, and contribute to the continuous improvement of the application.

# Conclusions

In this final chapter, we reflect on the journey of developing the GradeTracker application, highlighting achievements, lessons learned, and future directions for the project.

## Summary of Achievements:

Throughout the development process, GradeTracker has successfully achieved its primary objectives of providing students with a user-friendly tool for tracking academic progress and managing course grades. Key achievements include:

**Streamlined Grade Tracking:** GradeTracker offers an intuitive dashboard interface that allows users to easily view and manage grades for multiple courses in one place. This feature enhances efficiency and organization in academic record-keeping.

**Goal Setting and Assignment Management:** The application facilitates goal setting for individual courses, empowering students to set academic targets and monitor their progress towards them. Additionally, the assignment management feature enables users to stay on top of upcoming deadlines and track completed tasks effectively.

## Reflections and Lessons Learned:

The development of GradeTracker has been a valuable learning experience, offering insights into various aspects of software development and project management. Key reflections and lessons learned include:

**Technical Skills Development:** The project provided opportunities to enhance technical skills in Java programming, JavaFX framework, and database management. Overcoming technical challenges and implementing complex features contributed to skill development and problem-solving abilities.

**User-Centric Design:** Emphasizing user-centered design principles was crucial in creating an intuitive and user-friendly application. Conducting user feedback sessions and iterating on design iterations helped in understanding user needs and preferences better.

**Project Management:** Effective project management practices, such as task prioritization, milestone tracking, and communication, played a vital role in project success. Adhering to a structured development process ensured timely delivery and quality outcomes.

**Legal and Ethical Considerations:** Adhering to legal and ethical standards, such as data privacy regulations and intellectual property rights, was essential throughout the development process. Ensuring compliance with relevant laws and regulations helped in building trust and credibility among users.

## Future Work:

While GradeTracker has achieved significant milestones, there are several opportunities for further enhancement and development:

Enhanced Reporting and Analytics: Implementing advanced reporting features and analytics capabilities would provide users with deeper insights into their academic performance and progress trends.

**Integration with LMS Platforms:** Integrating GradeTracker with popular Learning Management Systems (LMS) used in educational institutions would streamline data import/export and enhance interoperability.

**Mobile Application Development:** Developing a mobile version of GradeTracker would extend accessibility and convenience, allowing users to track grades and manage assignments on-the-go.

**Community Engagement and Feedback:** Establishing a feedback mechanism and actively engaging with the user community would enable continuous improvement and refinement of GradeTracker based on user suggestions and preferences.

## Conclusion:

In conclusion, the development of GradeTracker has been a rewarding journey, marked by significant achievements, valuable lessons learned, and exciting opportunities for future growth. By prioritizing user needs, adhering to best practices, and embracing continuous improvement, GradeTracker has the potential to make a meaningful impact on students' academic experiences. As we reflect on the project's accomplishments and challenges, we remain committed to further enhancing GradeTracker and advancing its capabilities to better serve the academic community.

# References

1. Typeset. (n.d.). Why a student grade tracker is necessary. Retrieved from <https://typeset.io/questions/why-student-grade-tracker-is-necessary-1vogq41eb4>
2. O’Neill, K. (2017). Developing an Effective Tracking and Improvement System for Learning and Teaching: Achievements and Future Challenges in Maintaining Academic. California State University, Dominguez Hills. Retrieved from <https://scholarworks.calstate.edu/concern/theses/jw827k269>
3. Shah, A. (2019). Development of Student Performance Tracking Apps. ResearchGate. Retrieved from <https://www.researchgate.net/publication/338342446_Development_of_Student_Performance_Tracking_Apps>
4. Umar, A., & Faiza, S. (2011). Developing an Effective Tracking and Improvement system for Learning and Teaching: Achievements and Future Challenges in Maintaining Academic. ResearchGate. Retrieved from <https://www.researchgate.net/publication/228883515_Developing_an_Effective_Tracking_and_Improvement_system_for_Learning_and_Teaching_Achievements_and_Future_Challenges_in_Maintaining_Academic>
5. IvyPanda. (n.d.). The tracking system in secondary school education. Retrieved from <https://ivypanda.com/essays/the-tracking-system-in-secondary-school-education-essay/>

# Bibliography

1. O'Rourke, K., & Hume, S. (2021). Exploring Student and Staff Perceptions of the Impact of Peer-Assisted Learning on Student Engagement, Satisfaction, and Academic Achievement in Health Professional Education. Journal of Research in Interprofessional Practice and Education, 11(1). Retrieved from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9684747/>
2. Willcoxson, L., Cotter, J., & Joy, S. (2021). Undergraduate Student Engagement and Retention in Higher Education: A Review of the Literature and Current Evidence. Educational Review, 1-20. Retrieved from <https://eric.ed.gov/?id=EJ1332727>
3. Liu, Y., Liu, R., & Zeng, W. (2021). Assessing the Impact of Mobile Applications on Student Engagement in ICT and Computer Science Education. ResearchGate. Retrieved from <https://www.researchgate.net/publication/378785994_Assessing_the_Impact_of_Mobile_Applications_on_Student_Engagement_in_ICT_and_Computer_Science_Education>
4. Hietala, P., & Nätynki, P. (2021). Digital Learning Platforms in Higher Education: Student Perceptions and Preferences. Retrieved from <https://www.theseus.fi/bitstream/handle/10024/333664/Thesis_HietalaPerttu_NatynkiPiritta.pdf?sequence=2&isAllowed=y>
5. Typeset. (n.d.). Why student grade tracker is necessary. Retrieved from <https://typeset.io/questions/why-student-grade-tracker-is-necessary-1vogq41eb4>
6. O’Neill, K. (2017). Developing an Effective Tracking and Improvement System for Learning and Teaching: Achievements and Future Challenges in Maintaining Academic. California State University, Dominguez Hills. Retrieved from <https://scholarworks.calstate.edu/concern/theses/jw827k269>
7. Shah, A. (2019). Development of Student Performance Tracking Apps. ResearchGate. Retrieved from <https://www.researchgate.net/publication/338342446_Development_of_Student_Performance_Tracking_Apps>
8. Umar, A., & Faiza, S. (2011). Developing an Effective Tracking and Improvement system for Learning and Teaching: Achievements and Future Challenges in Maintaining Academic. ResearchGate. Retrieved from <https://www.researchgate.net/publication/228883515_Developing_an_Effective_Tracking_and_Improvement_system_for_Learning_and_Teaching_Achievements_and_Future_Challenges_in_Maintaining_Academic>
9. IvyPanda. (n.d.). The tracking system in secondary school education. Retrieved from <https://ivypanda.com/essays/the-tracking-system-in-secondary-school-education-essay/>