University of Dhaka

Department of Computer Science and Engineering

Course: CSE-2112: Object-Oriented Programming Lab

Project Name: Smart Academic Manager

Submitted to :

Dr. Muhammad Ibrahim
Md. Ashraful Islam

Submitted by:

Md. Mahmudul Hasan (Roll: 10)
Afia Lubaina (Roll: 14)
Ahona Rahman (Roll: 59)

Introduction:

The Smart Academic Manager software is an application where a student can view and manage his daily academic tasks. It is a Java-based program that will be used to keep all study-related data, including student activity logs. It will offer the capability of generating exam and class schedules, displaying to-do lists and attendance records, showing upcoming events (like classes and due assignments and exams), revealing results, and providing information about the courses that are offered at the academy.

The primary goal of the Smart Academic Manager program is to integrate all academic activities into one central database. This system will be created to offer a safe, user-friendly, and dependable system. Ultimately, it will make the job of any student easier.

Objective:

This software will provide the user with an easy-to-use user interface for updating student educational data. Students can use it to simply preserve records of their academic progress, according to its various features. Using a manual method makes it challenging to accomplish this goal because the information is dispersed and often redundant, and gathering pertinent data can take a lot of time. This effort can help to lessen all of these issues.

The project's primary goal is to offer convenience by simplifying the record-generating process and minimising paperwork for students. The project's main objective is to convey information simply and understandably.

Project Features:

The primary intent of the Smart Academic Manager software is to give a student the ability to update, change, and discover his academic process. Here a student can manage his current running courses, task list, upcoming events. Besides he can also manage attendance, results, syllabus, assignments, booklist of any particular list. The detailed description is given below:

1. Log In and New Account Creation:

With their name and password, a student will be able to log in to this software. The student can access and use the software if the name and password are correct. If the name or password is not validated, an error message will appear. The user must sign up to create a new account. A student must enter his name and password to create a new account.

2. Summary:

There will a screen dedicated to show summary of all the important information of the student such as his attendance percectage, task completion rate, events that is gonna be held today and also a short list of upcoming events and to do list.

3. Course:

Here a student can add, update or delete any courses as per his wish. Every course will contain the course name, course code and course teacher name information here.

4. Track:

Under every course, a student can keep a track of the topics that has been taught in the classroom or that can arrive in the examination. He can also keep update whether he has studied that topic or not.

5. Assignments:

Besides the track screen, there will also be a screen dedicated to assignments of any particular course where he can keep the list of the assignments of any particular course.

6. Attendance:

A dedicated screen to attendance will be provided where a student can see through a graph of his attendance percentage in every enrolled courses. He can update this any time. And also can see details information about attendance in every course.

7. Results:

Student can manage his internal evaluation and final exam marks, grade, grade points here.

8. Events:

Here all the events such as class, lab or exam can be listed. They will get sorted according to their time so that student can understand which events are going to be held after another event.

9. To-Do List:

A to do list can also be maintained in the app.

10. Book List:

A list of all books can be stored here to get reference of the books whenever there will be any need.

Tools and Technologies:

- **Platform:** Windows, Linux, macOS
- **Code IDE :** Intellij with SceneBuilder
- **CodeBase:** JAVA
- **Framework:** JavaFX
- **Database:** MySQL
- **Design Code Base:** CSS
- **UX Code Base:** FXML

System Design:

Class Hierarchy:

The class hierarchy of the Smart Academic Manager is described below with the UML diagram. The main features of oop access control, inheritance, method overriding, method overloading, interface, exception handling, string operations, collections framework, etc. are properly used in the software. The class hierarchy given below is used for storing information for the students. The details of the classes in the diagram are given below:

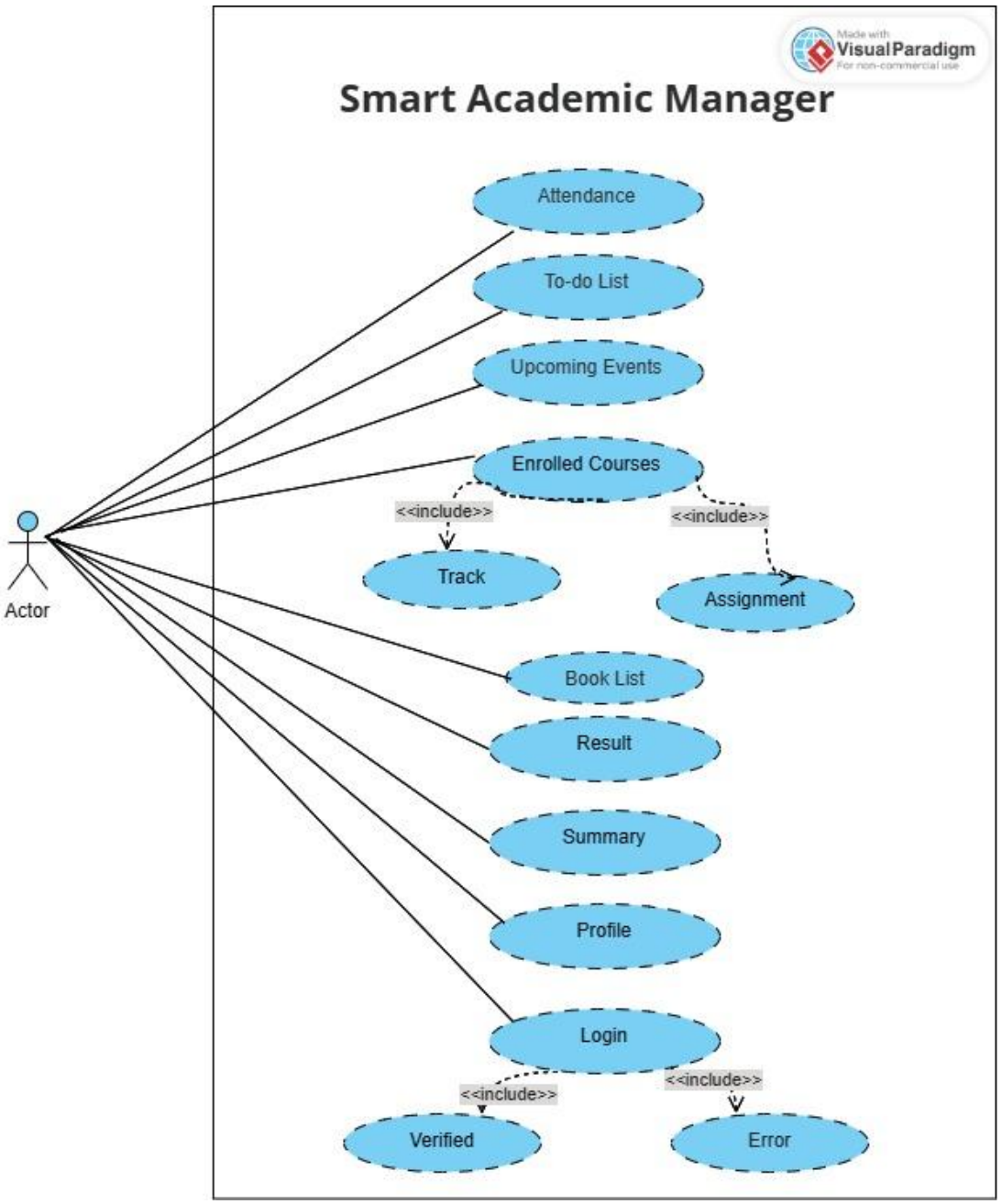

Fig: Complete Use Case Diagram

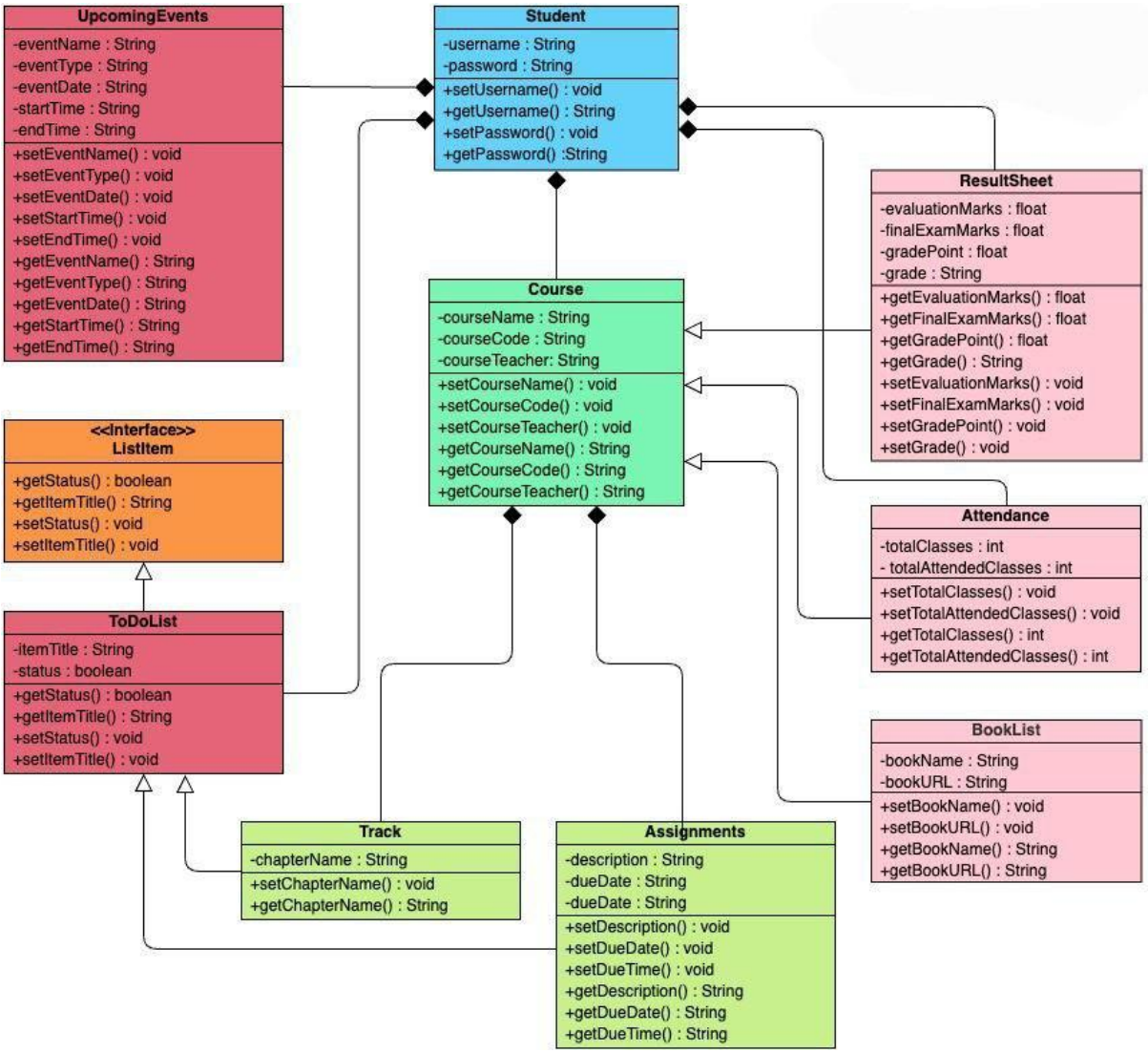

Fig: UML Class Diagram

Class Details:

Student:

It's the superclass of this hierarchy. As this hierarchy is basically used for storing the information, the common information, such as the name and password, is here in this class.

Course:

This class is associated with the Student class. This class is basically used for storing the course details (like course name, course code, and course teacher).

There are also methods to access the fields of the class. The object of this class is mainly used in the CourseSceneController and other classes.

Summary:
This class is associated with the Student class, inherits the Course class, and has a field to store the information of the Course class. It has methods to access its fields. This object of this class is mainly used in the DashboardScreenController class.

Book:
This class inherits the Course class and has a field to store the information of the Course class. This class contains BookName and BookURL. It has methods to access its fields. This object of this class is mainly used in the BookCardController and BookListController classes.

Attendance:
This class inherits the Course class and has a field to store the information of the Course class. This class contains attendedClasses and totalClasses. It has methods to access its fields. This object of this class is mainly used in the AttandenceCardController class.

ResultSheet:
This class inherits the Course class and has a field to store the information of the Course class. It has methods to access its fields. This class contains evaluation, finalExam, gradePoint, grade. This object of this class is mainly used in the ResultScreenController class.

ListItem:
This class is an interface used for Status and ItemTitle. It has methods to access its fields. This object of this class is mainly used in the CourseAssignmentScreenController, CourseTrackScreenController, and ToDoListController classes.

ToDoList:
This class implements ListItem. This object of this class is mainly used in the ResultScreenController class. This object of this class is mainly used in the CourseAssignmentScreenController, CourseTrackScreenController, and ToDoListController classes.

Assignment:

This class inherits the ToDoList class and has a field to store the information of the ToDoList class. This class contains description, dueTime, and dueDate. It has methods to access its fields. This object of this class is mainly used in the CourseAssignmentScreenController, AssignmentCardController, and AssignmentTimeComparator classes.

Track:

This class inherits the ToDoList class and has a field to store the information of the ToDoList class. This class contains chapterName. It has methods to access its fields. This object of this class is mainly used in the CourseTrackScreenController and TrackCardController classes.

Event:

This class is associated with the Student class. This class is basically used for storing the event details (like eventName, eventType, startTime, endTime, and eventDate). There are also methods to access the fields of the class. The object of this class is mainly used in the EventScreenController and other classes.

ImageTextPair:

This class is basically used for storing the image details (like imageView and text). There are also methods to access the fields of the class. The object of this class is mainly used in the AssignmentCardController, CourseCardController, TextItemCardController, ToDoListController, and TrackCardController classes.

ImageTextAnchorTriple:

This class inherits the ImageTextPair class and has a field to store the information of the ImageTextPair class. This class contains anchorPane. It has methods to access its fields. This object of this class is mainly used in the AssignmentCardController, CourseCardController, TextItemCardController, ToDoListController, and TrackCardController classes.

AssignmentTimeComparator:

This class is used to sort the assignments in chronological order according to their due time.

EventTimeComparator:

This class is used to sort the events according to their start and end time in chronological order.

Controller Classes:

All the controller classes are created to manage every fxml screen that is associated with it. For example, *DashboardScreenConroller* controls *DashboardScreen.fxml* file. In every controller classes there is method named *handleEvents* that mainly handles all the inputs from the user.

Database Manager:

It connects the app with the database and then manages the database as the user wants. It has *add, update, delete, currentList* methods dedicated for every screen to handle add, update, delete or retrieving the current list events if the user wants.

User Interface (UI):

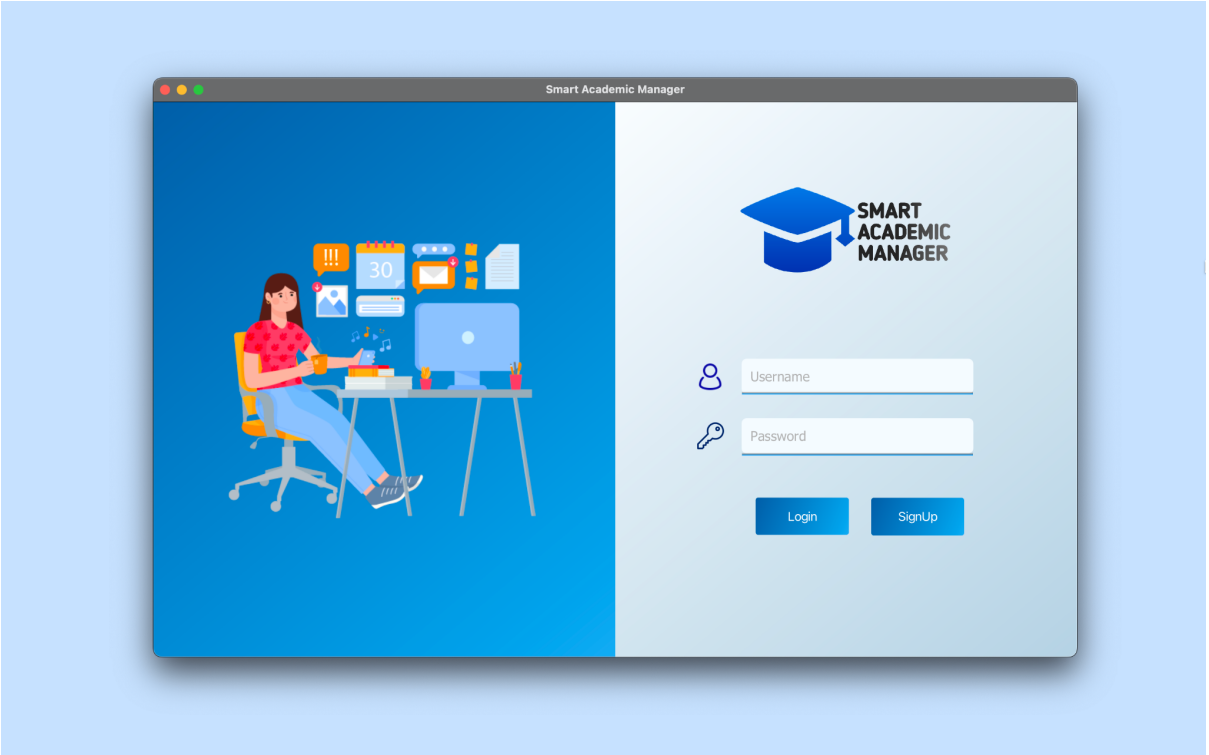

Fig. Log in Screen

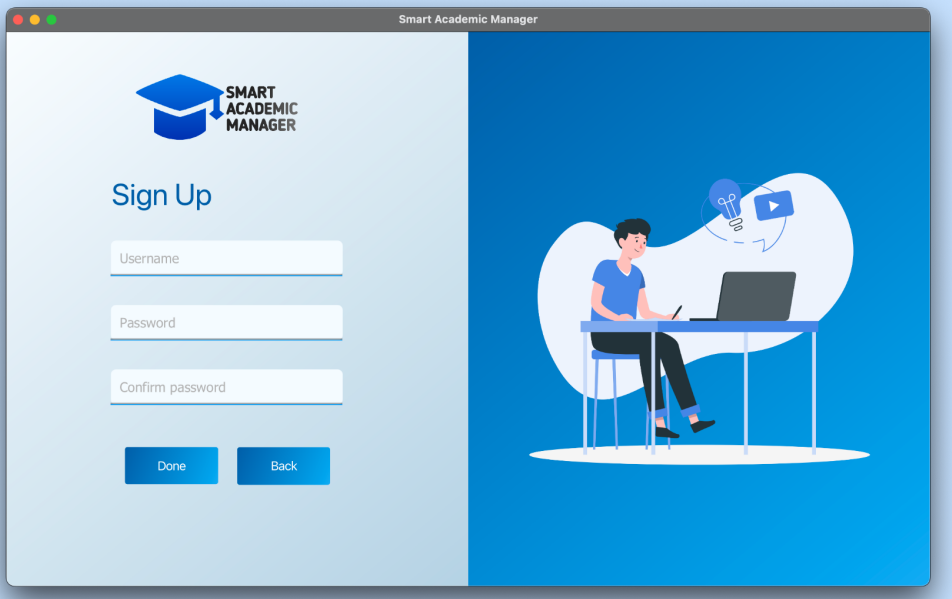

Fig. Sign up Screen

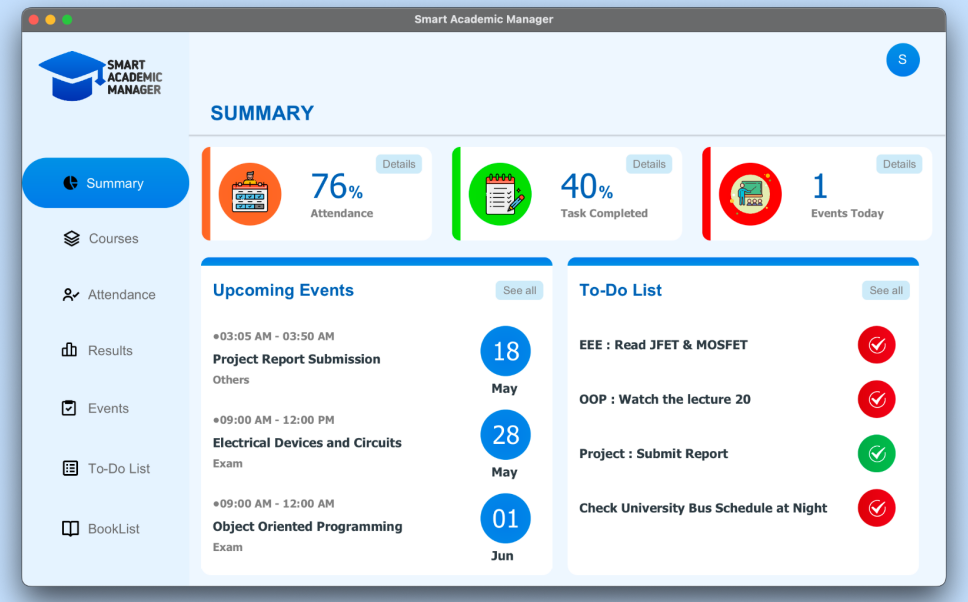

Fig. Summary Screen

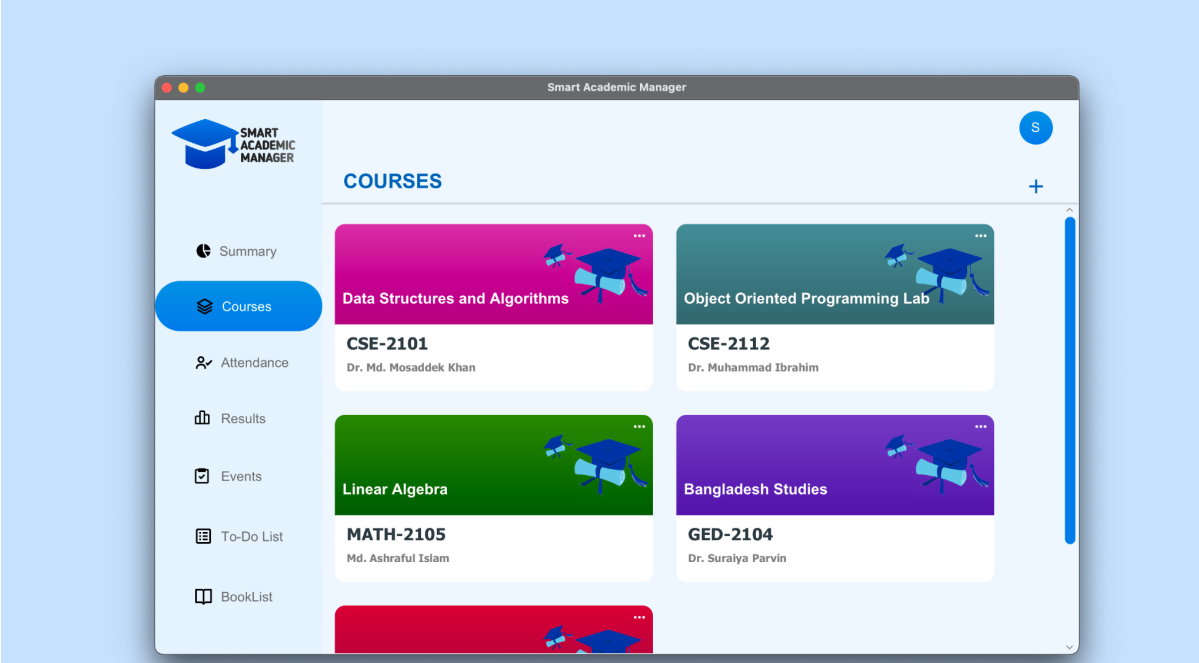

Fig. Course Screen

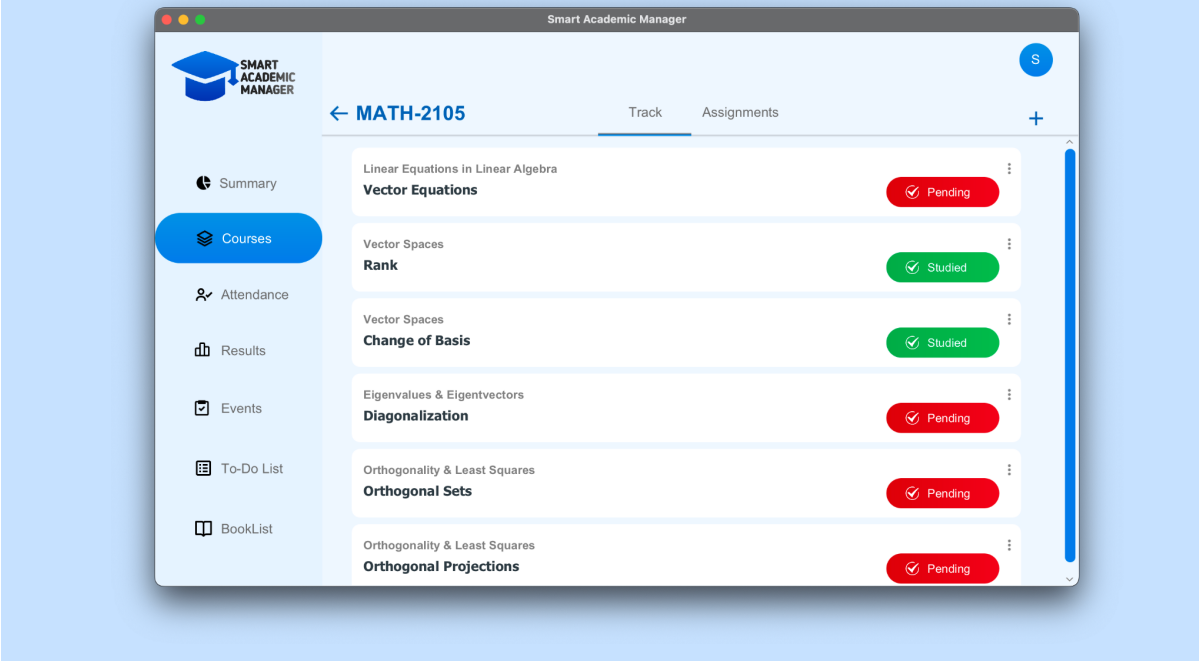

Fig. Track Screen Inside a Course

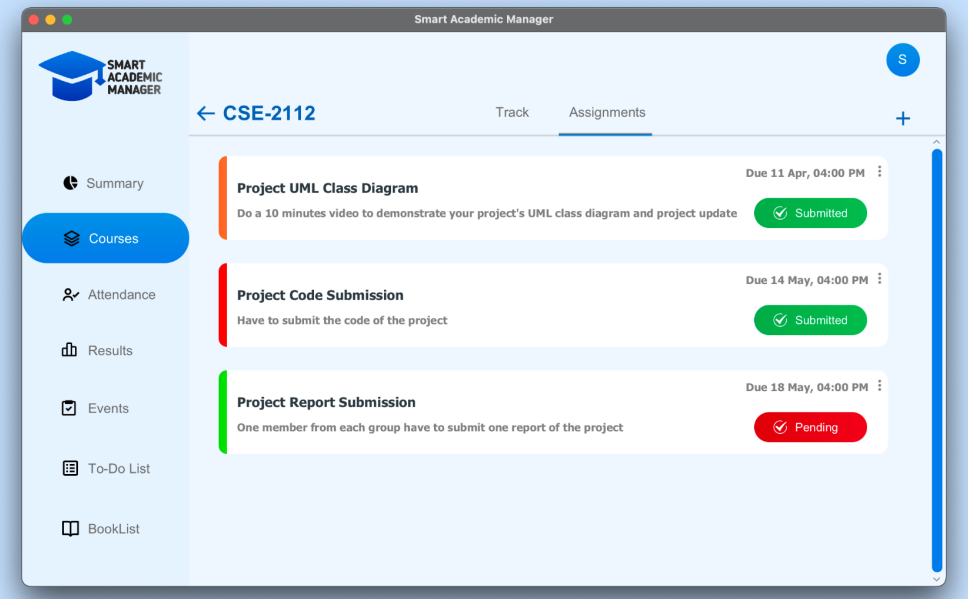

Fig. Assignments Screen Inside a Course

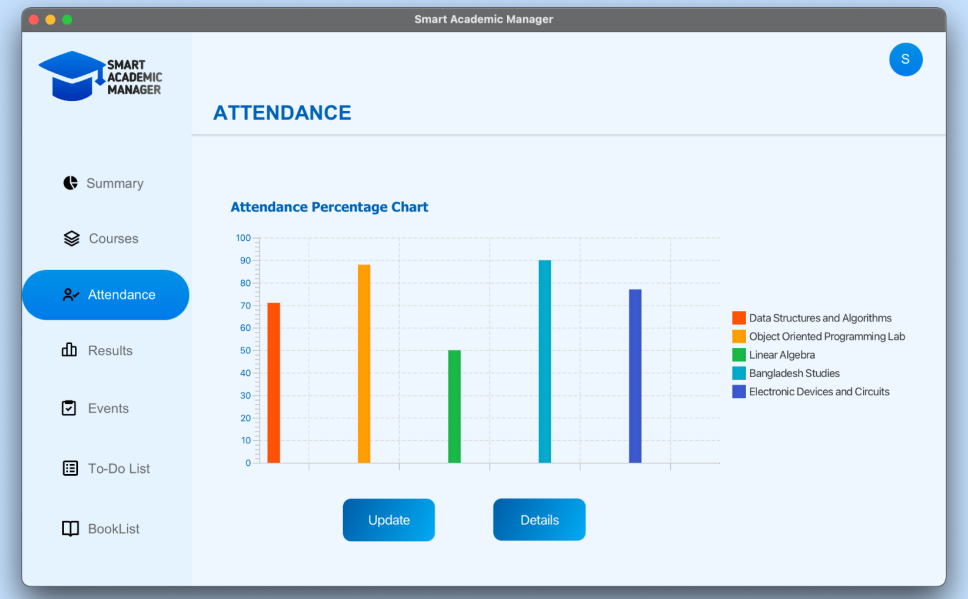

Fig. Attendance Screen

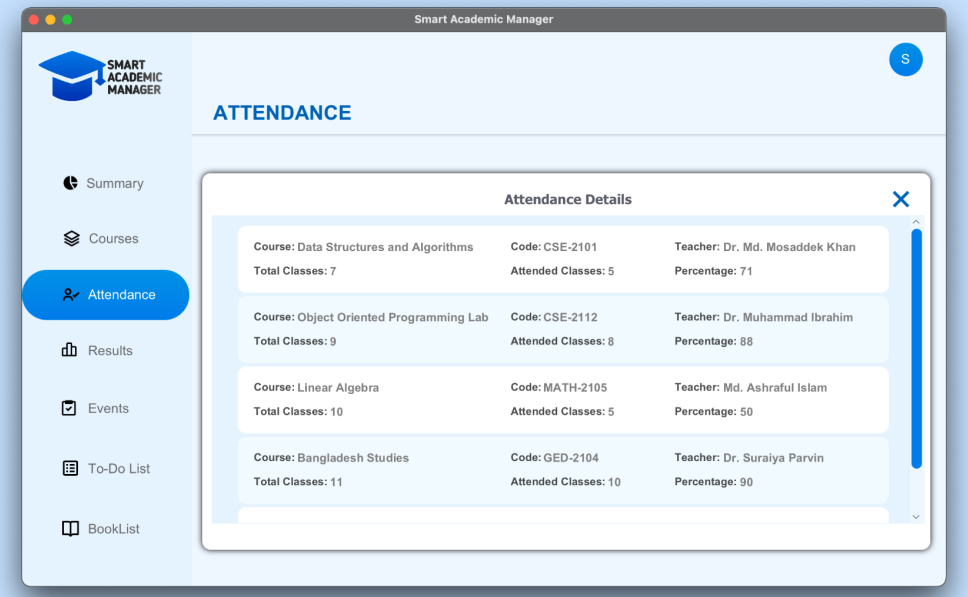

Fig. Attendance Details Screen

Fig. Results Screen

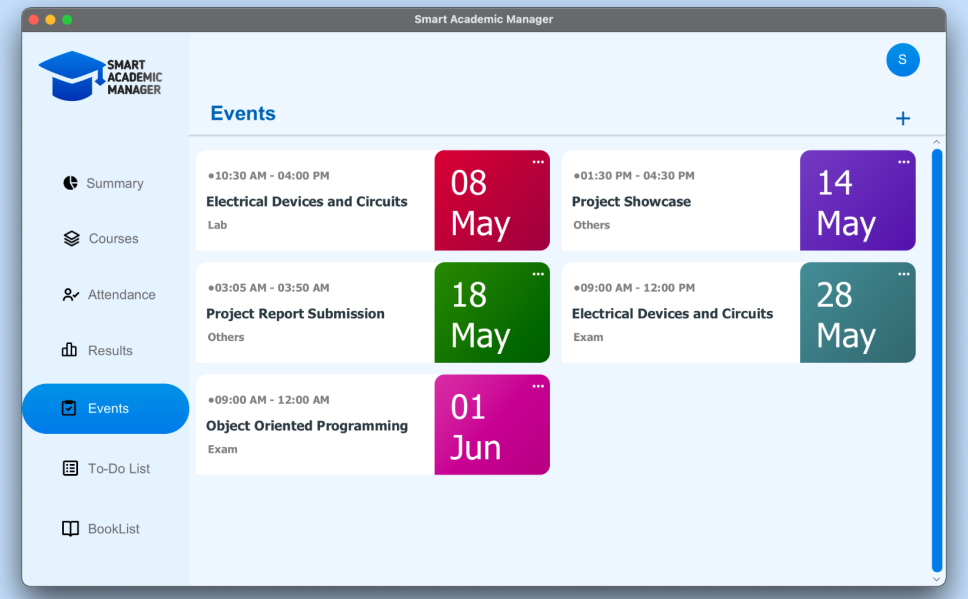

Fig. Events Screen

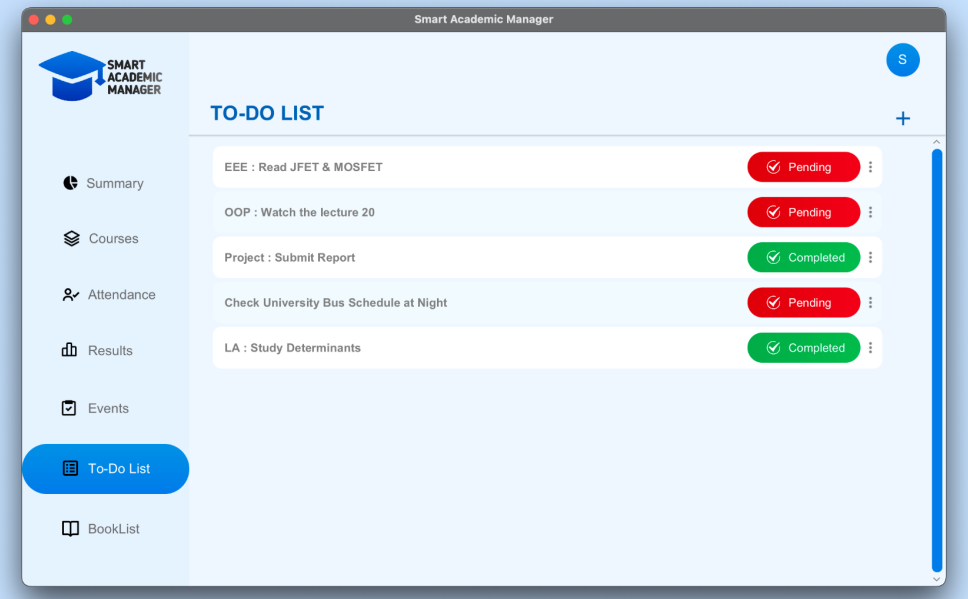

Fig. To-Do List Screen

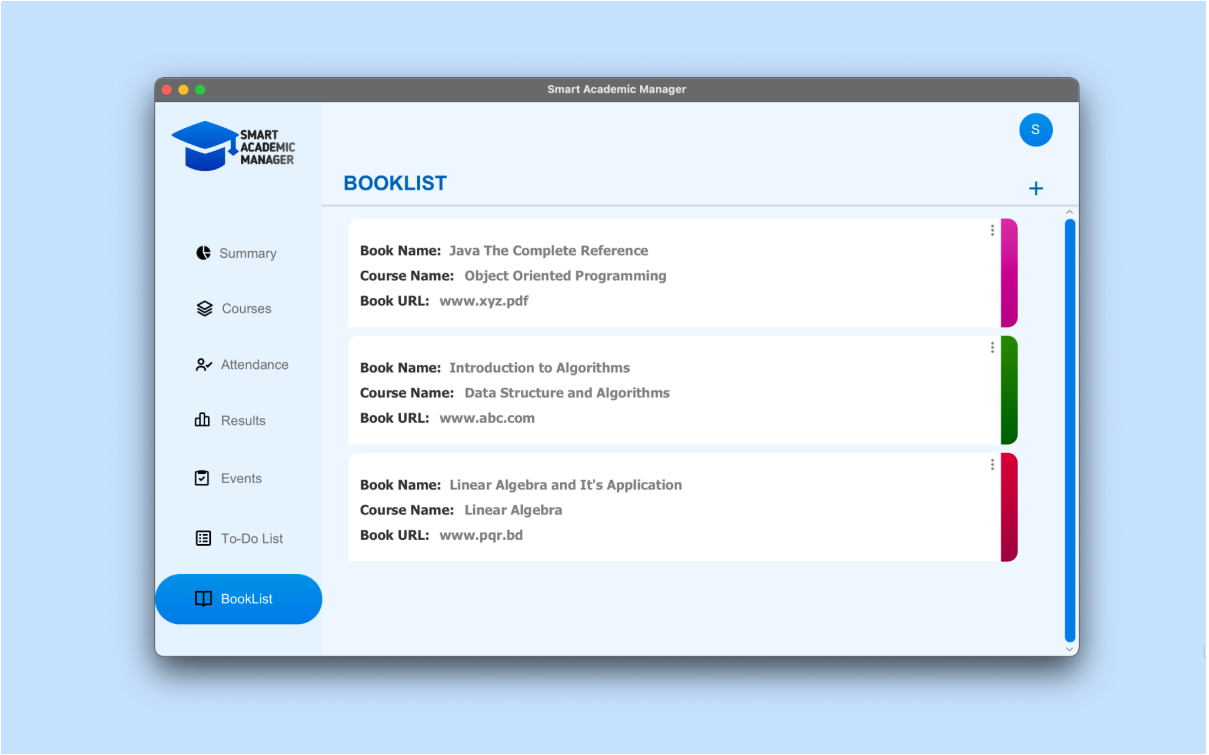

Fig. BookList Screen

Discussion:

In the project, JavaFX is used for designing and demonstration purposes. Besides, many Java features are used. These features are described below with an explanation.

Interface:

An interface in Java is the blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interfaces, and no method bodies. It is used to achieve abstraction and multiple inheritance in Java.

Encapsulation:

Encapsulation is defined as the wrapping of data into a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is that it is a protective shield that prevents the data from being accessed by code outside this shield.

Polymorphism:

Polymorphism is a concept of object-oriented programming that allows us to perform a single action in different forms. It is an OOP design that empowers classes with various functionalities to execute or share a common interface. The helpfulness of polymorphism is that code written in various classes has no impact on which class it has a place in since they are utilized similarly.

Function overriding:

If a subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java. In other words, if a subclass provides the specific implementation of the method that has been declared by one of its parent classes, it is known as method overriding.

Function overloading:

If a class has multiple methods with the same name but different parameters, it is known as method overloading.

Collection framework:

The collection in Java is a framework that provides an architecture to store and manipulate groups of objects. Java Collections can achieve all the operations that you perform on data, such as searching, sorting, insertion, manipulation, and deletion. A Java collection is a single unit of objects. The Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

Exception Handling:

Exception handling in Java is one of the powerful mechanisms to handle runtime errors so that the normal flow of the application can be maintained. In Java, an exception is an event that disrupts the normal flow of the program. It is an object that is thrown at runtime. The java.lang.Throwable class is the root class of the Java exception hierarchy, inherited by two subclasses: Exception and Error.

These are the major features that are used in the project.

Conclusion:

We gained a lot of experience and overcame a lot of challenges while working on this project. For us, creating software is a completely new and practical experience. As a result, we had to start from the beginning. Through this effort, we learned about the JavaFX (special effects in the Java programming language) library. Our ability to think and imagine has grown. We improved our communication abilities by working together as a group. It was, in fact, a fresh experience for us.

This project assisted us in getting significant information and practical experience in a variety of areas, such as using Java features, using the oop principal, CSS, responsive templates, designing reports, and database management using SQL. The entire system has been safeguarded. The project also helped us comprehend project development phases and the software development life cycle. We learned how to test various project aspects.
The challenge we've encountered is that creating software is a whole new experience for us and differs from the programming we're used to. We had to learn things that were completely foreign to us from the start. We had to learn stuff through video lessons, the internet, and study materials. Making software is a highly logical job to accomplish because we aim to make crucial and time-consuming chores manageable for consumers through the program. It is difficult to create a user-friendly model since we must work with each and every point of the model. To a large extent, there is room for further development in our project. This system can be enhanced with a variety of features. Another feature we wanted to include was the option to schedule teachers based on their availability within the time period. Due to schedule constraints, this feature is not included.

Future plan:

- Improve the graphical representation of the Software.
- Feature upgrade for making it more reliable.
- Try to make it online-based.
- Implement multiuser options for students.
- Take user feedback and improve features according to their opinion.
- Introduce a new environment and scenes.
- Build a teacher interface so that teachers can also interact

GitHub Repository:

https://github.com/afia-lubaina/Smart-Academic-Manager/tree/SAM