PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

B.TECH. (CSE)

VI SEMESTER

UE20CS352 – Object Oriented Analysis and Design with Java

Mini-Project Report

on

Evently: Event Management System

SUBMITTED BY

Name	SRN	Roll No
Vibha Murthy	PES1UG20CS495	21
Vibhav Deepak	PES1UG20CS496	15
Vidisha Chandra	PES1UG20CS498	11
Vishakha Hegde	PES1UG20CS506	09

Section I

January – May 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BENGALURU – 560100, KARNATAKA, INDIA

PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

Project Synopsis:

This project aims to create an event management system called Evently that enables organizers to manage events efficiently and attendees to view and participate in events seamlessly. The application will allow speakers to add their speeches and become part of the event. Attendees can view the list of events, register, and attend them. Admins can create, modify, and remove events. Sponsors can sponsor events with their products, and organizers can attract sponsors and promote their products in events. The objective is to develop a user-friendly and intuitive design that streamlines the process of organizing events and enhances the attendee experience. The application will be built using Java and its libraries to ensure robustness and scalability.

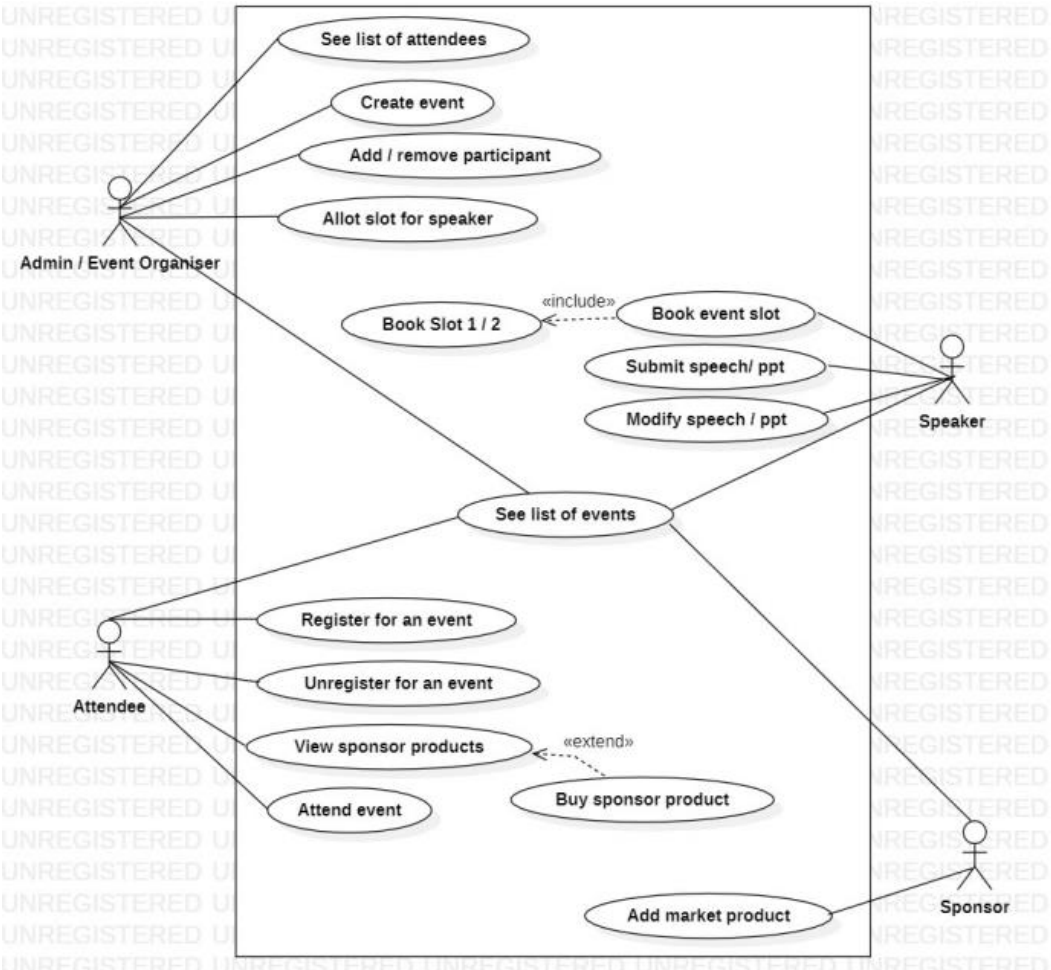
Features:

- ***Speaker Management**: Speakers can add their speeches and become part of the event. This feature allows organizers to manage speakers effectively and streamline the speaker selection process. Attendees can view the list of speakers and their speeches before attending the event, which helps in better event planning and management.*
- ***Attendee Management**: Attendees can view the list of events and attend them. This feature helps attendees to plan and schedule their attendance in events. Attendee management also includes ticket booking and payment processing, which ensures a smooth experience for attendees.*
- ***Event Management**: Admins can create, modify, and remove events. This feature enables organizers to manage events and keep attendees informed about the event schedule and updates. Event management includes creating event pages, setting up registration forms, and managing event logistics.*
- ***Sponsorship Management**: Sponsors can sponsor events with their products. This feature enables organizers to attract sponsors and promote their products in events. Sponsorship management also includes setting up sponsorship packages and managing sponsor communication.*

The Evently application will be built using Java and, with a back-end database to store event information and user details, which can be implemented with JDBC. A client-server architecture model will be implemented to facilitate communication between the application and the devices used by attendees, speakers, sponsors, and organizers. The application will allow for the sharing of event details, attendee registrations, speaker information, and sponsor advertisements between devices. The back-end database will store past event data and user details, allowing for efficient event planning and management.

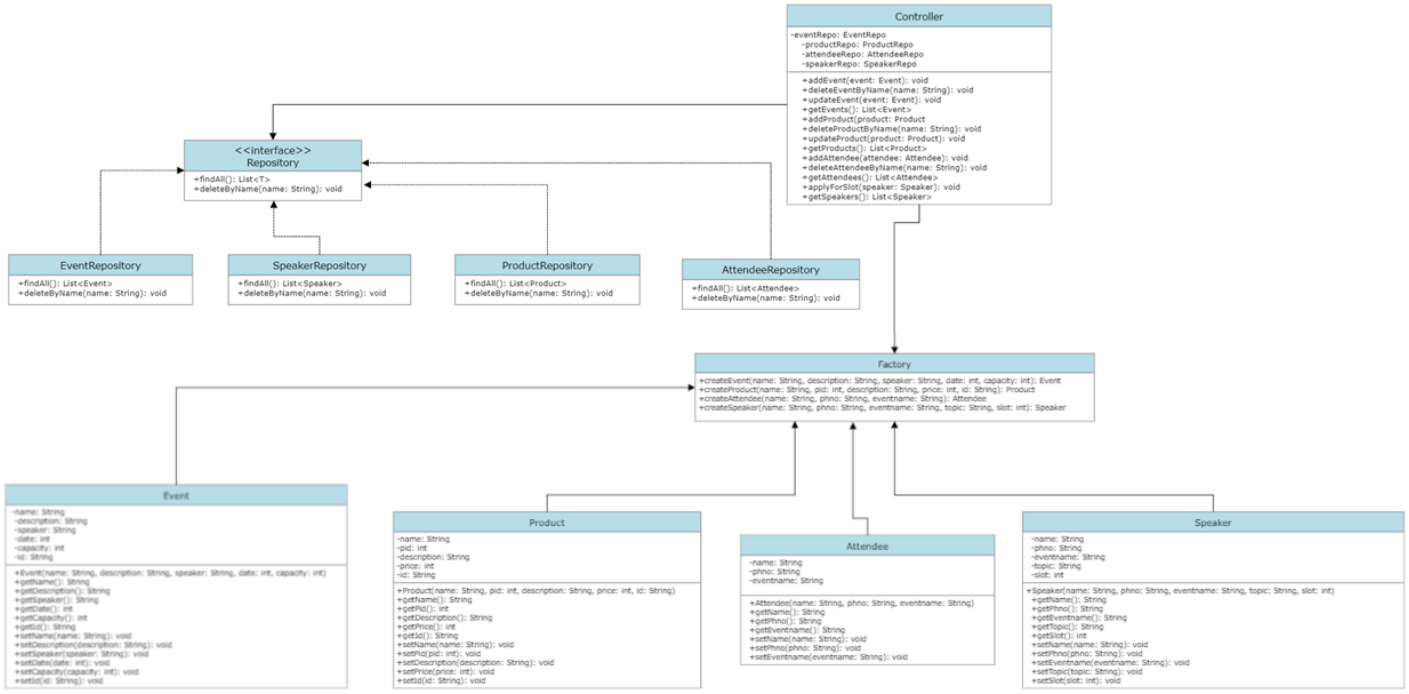
Models

Use Case Model



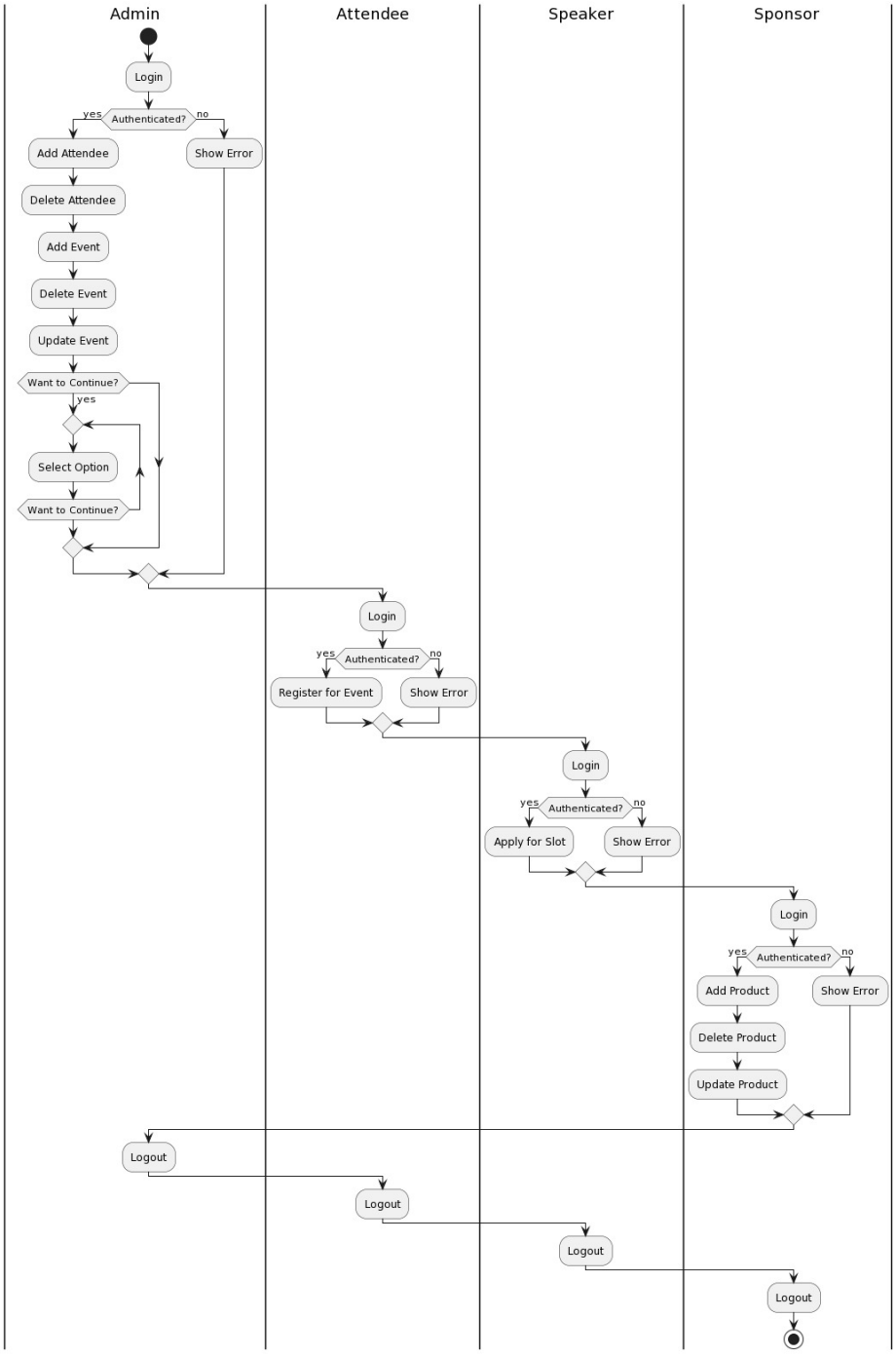
PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

Class Model



PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

Activity Diagram



Architectural Pattern

Model-View-Controller

MVC (Model-View-Controller) is a software architectural pattern commonly used in developing user interfaces for web applications. The pattern separates an application into three interconnected components: the model, the view, and the controller.

The Model component represents the data and the business logic of the application. It is responsible for managing the data and performing operations on it.

The View component represents the user interface of the application. It is responsible for presenting the data to the user in a way that is understandable and meaningful.

The Controller component acts as an intermediary between the Model and the View. It receives input from the user and manipulates the Model to perform the requested operation. It also updates the View with the changes made to the Model.

Using the MVC pattern, developers can create software that is modular, reusable, and easy to maintain. It also allows for easier testing and debugging, as each component can be tested separately.

MVC is widely used in developing web applications, but it can also be used in other software development projects where there is a need to separate the presentation layer from the business logic and data.

The below snapshot gives a glimpse into the directory structure of our project which gives a clear demarcation of MVC models.

Design Principle and Patterns

SOLID

1. **Single Responsibility Principle (SRP)**: Each class has a single responsibility and encapsulates related behavior. For example, the EventRepository class is responsible for managing events, and the ProductRepository class is responsible for managing products.
2. **Open/Closed Principle (OCP)**: The classes are designed to be open for extension but closed for modification. For example, new types of products or events can be added by creating new subclasses that extend the Product and Event classes without modifying the existing code.
3. **Interface Segregation Principle (ISP)**: The Mongo Repository interface provides a set of methods that are common to all repositories, but each specific repository only implements the methods that are relevant to its functionality.
4. **Dependency Inversion Principle (DIP)**: The Controller class depends on the Mongo Repository interface, not on specific repository implementations, which allows for flexibility in changing the repository implementation without affecting the Controller class.

Design Patterns

1. **Factory Pattern**

   The Factory pattern in the Evently project provides a consistent interface for creating objects like events, attendees, sponsors, or admins. It allows subclasses to create specific types of objects, based on specific requirements. This improves modularity and extensibility, reduces coupling between classes, and ensures a consistent user experience. The Factory pattern also helps to avoid code duplication and ensures that objects are created using the same standards and procedures.

2. **Observer Pattern**

   The Observer pattern is a behavioral design pattern that allows an object (called the subject) to notify its dependents (called observers) when it undergoes a change. In the context of the Evently project, the Observer pattern can be used to notify attendees when a new event is created or when an existing event is updated or canceled. The Attendee class can be the observer, and the Event class can be the subject. Whenever a change is made to the Event object, the Attendee objects that are subscribed to it will be notified. This has been implemented in the form of a dropdown menu.

3. **Decorator Pattern**

   The Decorator pattern is a structural design pattern that allows behavior to be added to an individual object, either statically or dynamically, without affecting the behavior of other objects from the same class. In the context of the Evently project, the Decorator pattern can be used to add additional features or functionality to attendees, sponsors, or events. For example, a

speaker has an option to upload their file in addition to the admin editing events.

GitHub link to Code Repository - https://github.com/Vibha-Murthy/Event-Management-System-OOAD

Individual Contribution

Vibha Murthy	Speaker Functionality
Vibhav Deepak	Sponsor Functionality
Vidisha Chandra	Admin Functionality
Vishakha Hegde	Attendee Functionality