XinQiao

Software Architecture Document

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <04/03/2023> | <1.1> | Deployment and logical models | Vahagn Ghazaryan |
| <17/05/2023> | <1.2> | Change of use case model | Vahagn Ghazaryan Simon Krkyasharyan |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 References 4

2. Use-Case View

3. Logical View 4

3.1 Overview 4

3.2 Architecture - Package and Subsystem Layering 4

4. Deployment View 4

Software Architecture Document

# Introduction

## Purpose

This document provides a comprehensive architectural overview of the platform, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the platform.

## 1.2 References

None

# Use-Case View

The Xingqiao event management platform’s use cases are:

- Login

- Manage account

- Create Event

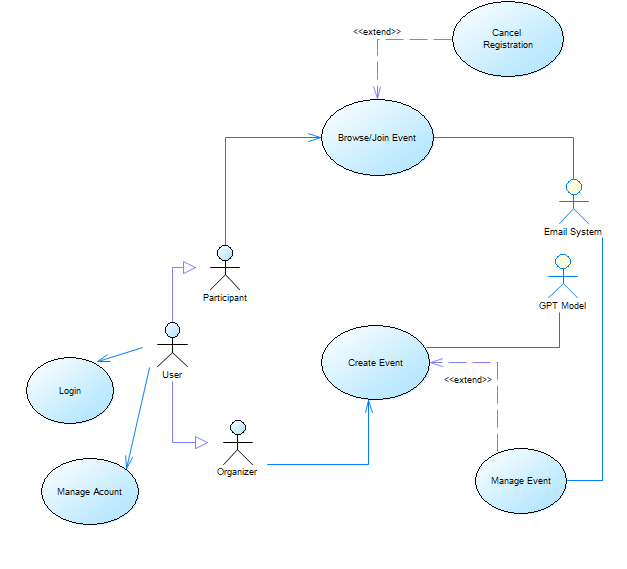
- Manage event

- Brows Events

- Register for Event

- Cancel Registration

These use cases are initiated by the Participant and Organizer actors. In addition, interaction with external actors; Email System and GPT Model.



*2.1 Login*

The Login use case allows all registered users (Organizers and Participants) to access their accounts on the event management website. Users can enter their login credentials such as their username and password to authenticate and gain access to their personal profile pages, event planning tools, and other site features.

*2.2 Manage account*

The event management website allows all users to manage their accounts by changing their photo, name, email, and password. This feature allows users to keep their information up-to-date and secure.

*2.3 Create Event*

The create event use case allows Organizers to create and publish new events on the event management website. Users can input event details such as date, time, location and ticketing information, and can also upload images and descriptions. Additionally they can add tags and ask GPT AI model to generate description.

*2.4 Manage event*

The "Manage Event" feature empowers Organizers to create, update, and control campus events efficiently. It provides event creation, editing, and deletion functionalities, along with registration and ticketing management, ensuring smooth event planning and execution.

*2.5 Browse/Join Event*

The Browse/Join Event use case allows Participants to search and discover new events on an event management website to join. Users can filter events by and other criteria to find events that match their interests. They can also view event details such as time, location, description, and pricing to decide whether to attend. If they are interested in the event they can press join button.

*2.6 Cancel Registration*

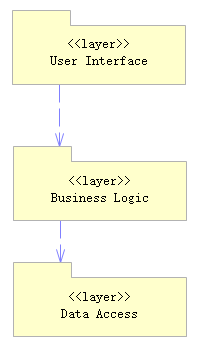
Participants have the flexibility to cancel their event registrations if they are no longer able to attend. This allows for efficient event management and ensures accurate participant counts.

# Logical View

## Overview

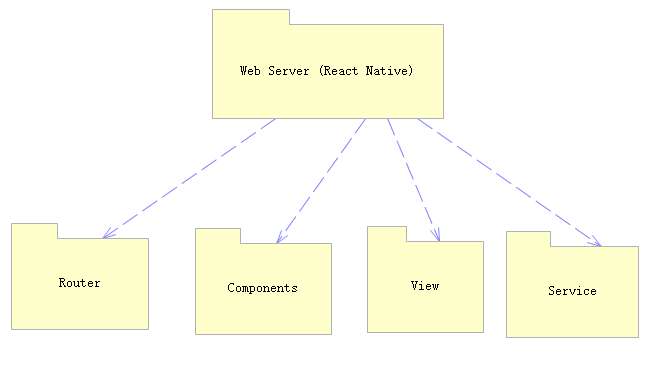
For the event management platform we used 3-tier client-server **layered architectural** style which would provide a scalable and efficient solution for managing events with a clear separation of concerns between different layers.

## Architecture - Package and Subsystem Layering



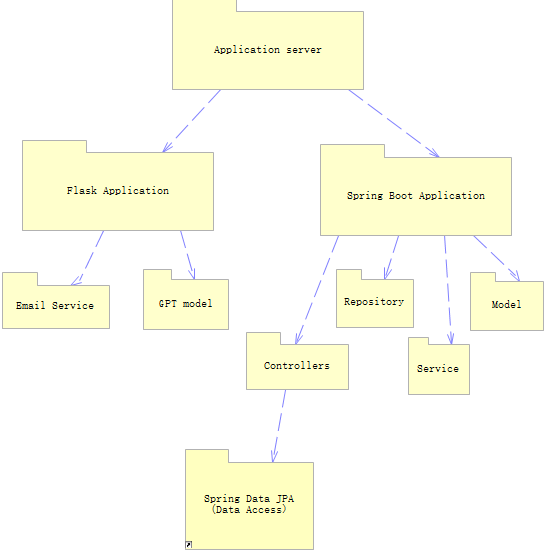
### User Interface layer

### User interface layer is implemented by React Naive Web Application. This includes all the pages and forms that a user interacts with such as the login page, profile page, event catalog page, event creation page, event management page.



### Business logic layer

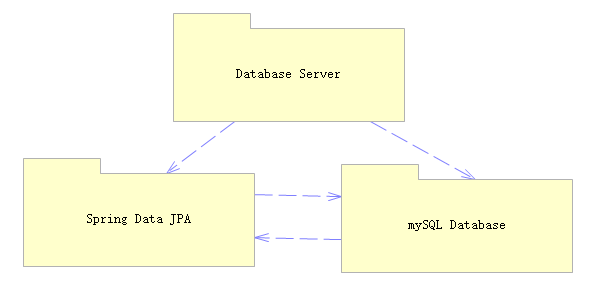
#### The application server would handle all the business logic of the website such as creating and managing events, sending emails and so on. It would also communicate with the database to retrieve and update data. We have two applications in server: Python’s flask app and Java’s spring Boot app. Flask application is responsible with the communication to Email Service and GPT AI model, while Spring boot is responsible with the interaction to Database.



### Data access layer

### Data access layer is realized by creating Database Server using Spring Boot JPA and mySQL database.

This includes all the data related to users and events made on the website. The database has several tables such as users, events, participation, location and so on.



# Deployment View

A description of the deployment view of the architecture describes the various physical nodes for the most typical platform configurations. Also describes the allocation of tasks (from the Process View) to the physical nodes.

This section is organized by physical network configuration; each such configuration is illustrated by a deployment diagram, followed by a mapping of processes to each processor.

# 

*Diagram Name: Deployment View*

## Client

The client can be any student or supervisor accessing the website via desktop PC or phone.

## Web Server

The webserver is responsible for handling all the requests made by the client. It provides a secure and scalable environment for hosting the platform. The server ensures that the website is accessible from any device and browser.

## Application Server

The application Server is responsible for handling all the requests coming from web server and responding them with either database info or micro services (email or gpt prompts).

## Database Server

The database server stores all the information related to the events, clients, and attendees. It provides a reliable and efficient way to store, retrieve and update data. The server also ensures data security, consistency, and availability.