EVENTME

*Design Document*

Alice Han hanalice@usc.edu

Oscar Hong othong@usc.edu

Yoonsoo Nam yoonsoon@usc.edu

CSCI 310 Team 30

*October 10, 2022*

# Contents

[**Contents**](#_546krc2tqt0v) **1**

[**Preface**](#_loj20wo0ofaa) **2**

[**Introduction**](#_7293286obfkg) **3**

[1.1 Motivation and Value](#_entrk7msrixq) 3

[1.2 Product Scope](#_k9lyx7j19p4y) 3

[1.3 Main Functions](#_xo819wnaek77) 3

[1.4 Business Context](#_td2yq46c58ld) 3

[**Architectural Design**](#_iffyg6tj6bul) **5**

[**Detailed Design**](#_rmb8gdaf3qf5) **5**

# Preface

As the global pandemic winds down and the weather is getting better, people have a need to attend different in-person events. But students and other individuals who may be financially insecure or money-conscious don’t particularly want to spend money on every single event being offered. Hence, EventMe is an application that will help users find free and/or cheap local events and register for them.

# 

# Introduction

### 1.1 Motivation and Value

With people going back to in-person normalcy, there is an increase of demand for in-person events. However, the limiting factors for attending events are proximity, money, and time. EventMe is a solution to this that helps users find curated events fitting their needs.

### 1.2 Product Scope

EventMe is a convenient application that will help users find free and/or cheap local events and register for them.

### 1.3 Main Functions

1. Explore the nearby events
   1. Search based on event type (e.g. music, arts, outdoor, etc) through buttons or tabs
   2. Search bar search for name of the event, location, or sponsoring organization
   3. Indicate range of dates to find only those events that fit the timeframe
   4. All of these events can be sorted from in the “Event Box” for
      1. Lowest to highest cost (default)
      2. Proximity
      3. Earliest to latest date
      4. Alphabetical
2. Map
   1. All events shown as markers
   2. When an event is clicked on, it will show in the “Event Box”
   3. Swiping up from the “Event Box” will take user back to map
3. Register
   1. On Events Registration page, users can see all information regarding the events
   2. When clicking on the Register button,
      1. If user not logged in, redirect to Login/Register
      2. If logged in and has not registered, successfully register ONLY if there is no conflicting events
         1. If conflict, warn user
         2. If they want to proceed, they can
      3. If logged in and has registered, ask to unregister
         1. If yes, event removed from the user’s registration
   3. All events will be shown in the Profile Tab

### 1.4 Business Context

The application is similar to Eventbrite. Eventbrite lets you browse, create, and promote local events. Also, a feature in Facebook, events, is a similar context where you can create local events. The difference is that we do not sell tickets but rather allow users to register. Also, we are different from Facebook's function because we do not require a social media connection or login to register for an event.

# Architectural Design

For the architectural design, we will be using the layered design architecture. The major components are Presentation Layer, Business Logic Layer, Data Access Layer, and Data Storing Layer.

The Presentation Layer is responsible for the user interface of the application. This will show the map for the events and present the list of events to the user. It will also provide all user functions such as signing up for events and viewing currently signed up events.

The Business Logic Layer takes care of the user’s request that has been made through the user interface. For example, when a user wants to send a request for signing up for events, the business logic layer would check if there are any conflicts with the other events that the user had already signed up for. Depending on the result of this, the Business Logic Layer would allow for users to send requests to the Data Access Layer.

The Data Access Layer would send requests to the Data Storing Layer. This would include calls to the database to send to the Business Logic Layer. For example, when the user attempts to sign up for an event through the Presentation Layer, the Business Logic Layer will request that the Data Access Layer send a request to check if the user has any conflicting events stored in the Data Storing Layer.

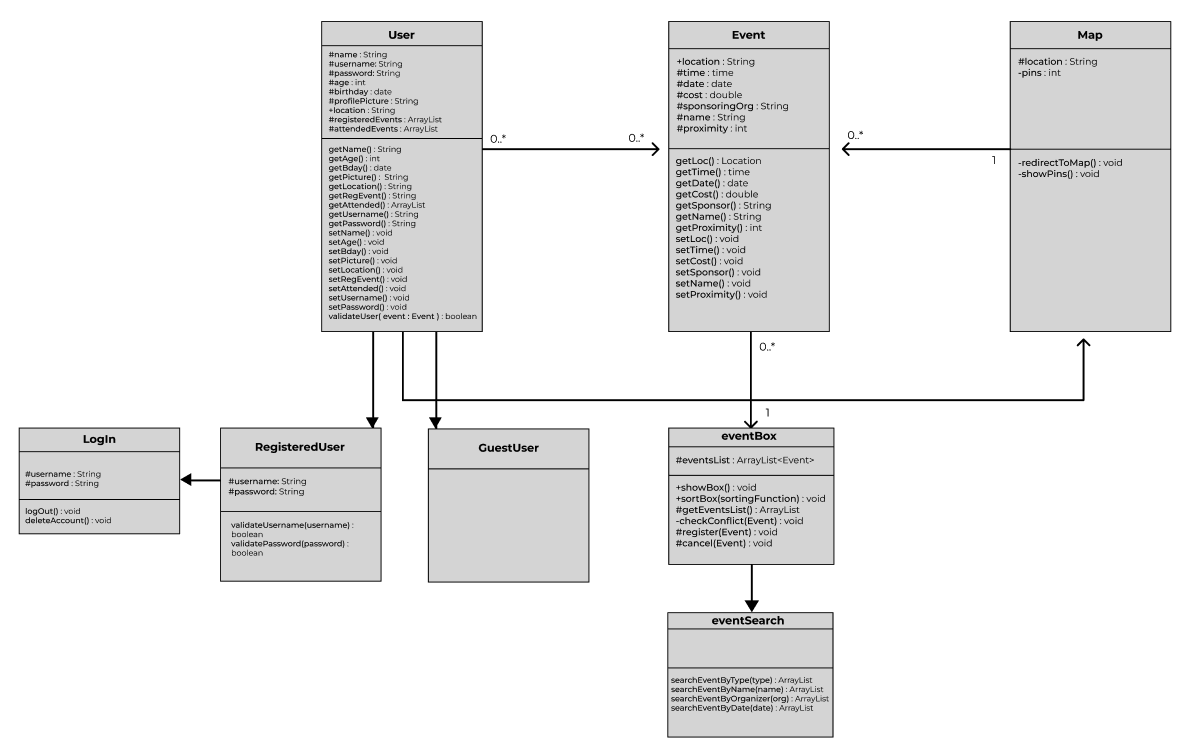
Finally, the Data Storing Layer is the core of all of these layers where we store data. This is where databases, file system, blob storage, documents, and all related data are stored.

We are using the layered design because this is an application that has multiple layers for its system. We need to separate each layer so that the access to the lower layers are validated before giving access. This is conducive to having a secured data processing system. Each of the components of this architectural design is chosen based on the core functionality of each component and the access level. The outermost layer is accessible by all users. However, as we go to the inner layers, we need to send requests to gain access. During the requests, each of the requests that the user makes is turned into another request to the inner layer. This transition occurs because we need to validate the requests made from each of the layers.

# 

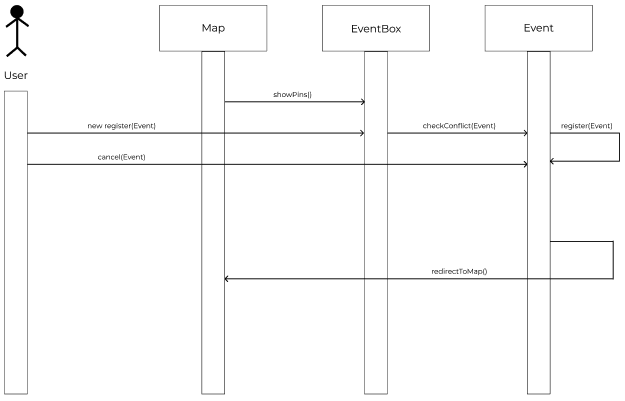
# Detailed Design

### 3.1 UML Class Diagram



### 

### 3.2 UML Sequence Diagrams

****

