

**Addis Ababa Institute of Technology**

**School of Information Technology and Engineering**

Department of SWE Eng.

Ethio tourism promotion

Group-20

**Software Design Specification**

**Group member’s name ID**

1. Biniam markos UGR/6500/15
2. Dibora taye UGR/ 2376/15
3. Eleni abebe UGR/ 2233/15
4. Hilina zemedkun UGR/1331/15
5. Ruth yeshitila UGR/ 8377/15

Submitted To: MR.Aderaw

Submission-Date: Mon,9 DEC,2024

# Table of Contents

## Contents

List of tables………………………………………………………………………………………………..I

List of figures……………………………………………………………………………………………...IV

Definition ,Acronyms ,Abbreviations……………………………………………………………………..V

[1. Introduction](#page7) 1

[1.1 Purpose](#page7) [1](#page7)

[1.2 General Overview](#page7) [1](#page7)

[1.3 Development Methods](#page7) [1](#page7)

1. [System Architecture](#page8) [2](#page8)

[2.1 Subsystem decomposition](#page8) [2](#page8)

[2.2 Hardware/software mapping](#page10) [4](#page10)

[3. Object Model](#page11) 6

[3.1 Class Diagram](#page11) 6

[3.2 Sequence Diagram](#page12) 6

3.3 State Diagram……………………………………………………………………………………….10

[4. Detailed Design](#page29) 11

[References](#page43) 34

# List of Tables

[TABLE 1: Event CLASS](#page29) 11

[TABLE 2: ATTRIBUTES DESCRIPTION FOR EVENT CLASS](#page29) 11

[TABLE 3: OPERATION DESCRIPTION FOR EVENT CLASS](#page30) 12

[TABLE 4: HISTORICAL EVENT CLASS](#page30) 14

[TABLE 5: ATTRIBUTES DESCRIPTION FOR HISTORICAL EVENT CLASS](#page31)  14

[TABLE 6: OPERATION DESCRIPTION FOR HISTORICAL EVENT CLASS](#page31) 15

[TABLE 7: CULTURAL EVENT CLASS](#page32) 15

[TABLE 8: ATTRIBUTES DESCRIPTION FOR CULTURAL EVENT CLASS](#page32)  16

[TABLE 9: OPERATION DESCRIPTION FOR CULTURAL EVENT CLASS](#page32) 17

[TABLE 10: DISCOVER ET CLASS](#page33) 17

[TABLE 11: ATTRIBUTES DESCRIPTION FOR DISCOVER ET CLASS](#page33) 18

[TABLE 12: OPERATION DESCRIPTION FOR DISCOVER ET CLASS](#page33)  18

[TABLE 13: NEWS EMAIL CLASS](#page34) 19

[TABLE 14: ATTRIBUTES DESCRIPTION FOR NEWS EMAIL CLASS](#page35) 19

[TABLE 15: OPERATION DESCRIPTION FOR NEWS EMAIL CLASS](#page35) 20

[TABLE 16: DESTINATION CLASS](#page36) 21

[TABLE 17: ATTRIBUTES DESCRIPTION FOR DESTINATION CLASS](#page36) 21

[TABLE 18: OPERATION DESCRIPTION FOR DESTINATION CLASS](#page37) 21

[TABLE 19: BLOG POST CLASS](#page38) 22

[TABLE 20: ATTRIBUTES DESCRIPTION FOR BLOG POST CLASS](#page38) [22](#page38)

[TABLE 21: OPERATION DESCRIPTION FOR BLOG POST CLASS](#page38) 23

[TABLE 22: TRIP PLAN CLASS](#page38) 24

[TABLE 23: ATTRIBUTES DESCRIPTION FOR TRIP PLAN CLASS](#page39) 24

[TABLE 24: OPERATION DESCRIPTION FOR TRIP PLAN CLASS](#page39) 25

[TABLE 25: BLOG MANAGER CLASS](#page40) 25

[TABLE 26: ATTRIBUTES DESCRIPTION FOR BLOG MANAGER CLASS](#page40) 25

TABLE 2[7: OPERATION DESCRIPTION FOR BLOG MANAGER CLASS](#page40) 26

[TABLE 28: ACCOUNT CLASS](#page41) 26

[TABLE 29: ATTRIBUTES DESCRIPTION FOR ACCOUNT CLASS](#page41) 26

[TABLE 30: OPERATION DESCRIPTION FOR ACCOUNT CLASS](#page42) 27

TABLE 31: USER CLASS……………………………………………………………………………………28

TABLE 32: ATTRIBUTES DESCRIPTION FOR USER CLASS…………………………………………………29

TABLE 33: [OPERATION DESCRIPTION FOR USER CLASS](#page42) 29

TABLE 34: ADMIN CLASS…………………………………………………………………………………30

TABLE 35: ATTRIBUTES DESCRIPTION FOR ADMIN CLASS……………………………………………….30

TABLE 36: [OPERATION DESCRIPTION FOR ADMIN CLASS](#page42)………………………………………………..30

TABLE 37: FEEDBACK CLASS…………………………………………………………………………...…32

TABLE 38: ATTRIBUTES DESCRIPTION FOR FEEDBACK CLASS……………………………………….32

TABLE 39: OPERATION DESCRIPTION FOR FEEDBACK CLASS.………………………………………….32

# List of figures

[FIG 1. TIER 1 DECOMPOSITION](#page8) [2](#page8)

[FIG 2. TIER 2 DECOMPOSITION](#page8) 3

[FIG 3. TIER 3 DECOMPOSITION](#page9) 4

[FIG 4. UML DEPLOYMENT DIAGRAM](#page10) 5

[FIG 5. CLASS DIAGRAM](#page11) 6

[FIG 6. UC-01 SEQUENCE DIAGRAM OF B](#page12)LOG POST 7

[FIG 7. UC 02 - SEQUENCE DIAGRAM OF](#page13) ADD DESTINATION [7](#page13)

[FIG 8. UC 03 - SEQUENCE DIAGRAM OF GET EVENT DETAIL](#page14)  [8](#page14)

[FIG 9. UC 04 - SEQUENCE DIAGRAM OF VIEW DET](#page15)AIL DESCRIPTION OF DESTINATION [9](#page15)

[FIG 10. UC 05 - SEQUENCE DIAGRAM OF S](#page16)EARCH TRIP 9

[FIG 11. - STATE DIAGRAM 1](#page17) 10

[FIG 12. [- STATE DIAGRAM 2](#page17) [10](#page17)](#page18)

Figure 17- state diagram of update and save trip plan 13

# Definitions, Acronyms, Abbreviation

* **HTML -** Hyper Text Mark Up language
* **CSS -** Cascading Styles Sheet
* **SDS -** Software Design Specification
* **API -** Application Programming Interface

# 1. Introduction

## 1.1 Purpose

The System Design document's objective is to translate system requirements of our website into a technical design that will be utilized to construct the application.

The software architecture and design for the Ethio tourism promotion website application will be detailed in this document. It includes details about the intended input, output, classes, and methods.

## 1.2 General Overview

The website we are building in our group is Ehio Tourism Promotion website application that could potentially promote our country and its cultural and historical sites. We decided to use use three-tier architectural style for our Ethio tourism promotional website involving REST API.

This architecture separates the system into three logical layers:

#### ****1. Presentation Tier****

This is the user-facing layer where users interact with the system.

* **Purpose**: Provide an intuitive and engaging interface for users to explore Ethiopian tourist destinations, cultural events, and travel services.
* **Technologies**: HTML, CSS, JavaScript, and possibly frameworks like **Bootstrap** for responsiveness or **React** for dynamic content.

#### ****2. Application Layer****

This is the intermediary layer that processes user requests, applies business rules, and communicates with the data layer.

* **Purpose**: Handle the core functionality of the system, including user authentication, search algorithms, and booking systems.
* **Technologies**:
* **Backend Frameworks** :node js, express js
* **API Development**: RESTful APIs for communication between the front-end and the back-end.

#### ****3. Data Tier****

This layer stores and manages all the system's data.

* **Purpose**: Provide a centralized, consistent, and secure storage for system data.
* **Technologies** :MySQL or Mongo DB(for structured and unstructured data)

## 1.3 Development Methods

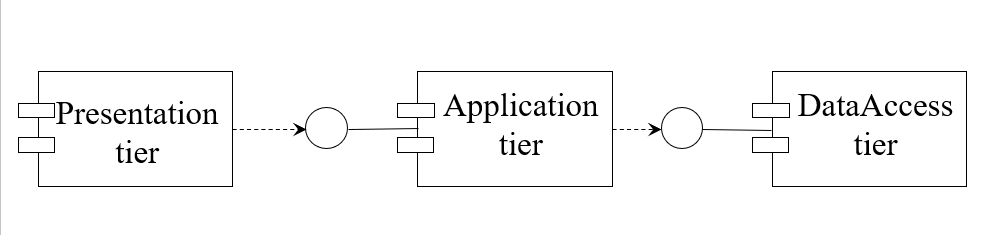
The backend part of our project will be developed using Node framework of java script . Where as,for the frontend part of our project,we will use HTML,CSS,JS AND REACT.We also will be using mysql or mongo db for data storage of our platform.

# 2. System Architecture

System architecture refers to the conceptual model that defines the structure, behavior, and more views of a system. It outlines how components interact, their relationships, and the principles guiding their design and evolution over time

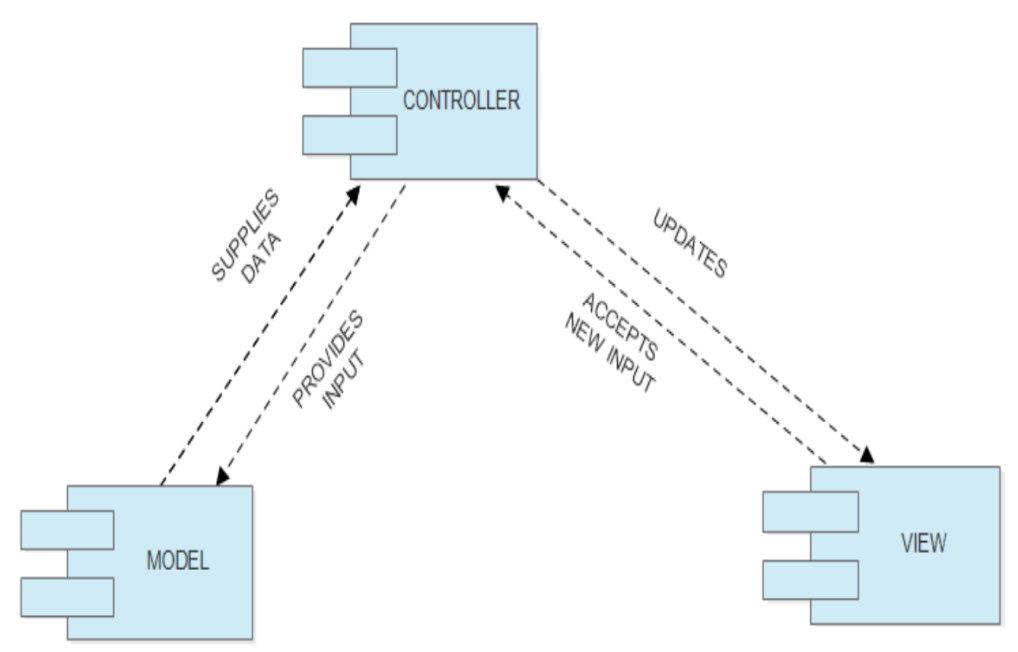
In our case we are using the popular Three-Tier architecture.

**Three-tier architecture** is a software design pattern that organizes an application into three distinct layers: the presentation tier, which handles user interaction; the application tier, which contains the business logic and processes user requests; and the data tier, responsible for data storage and retrieval

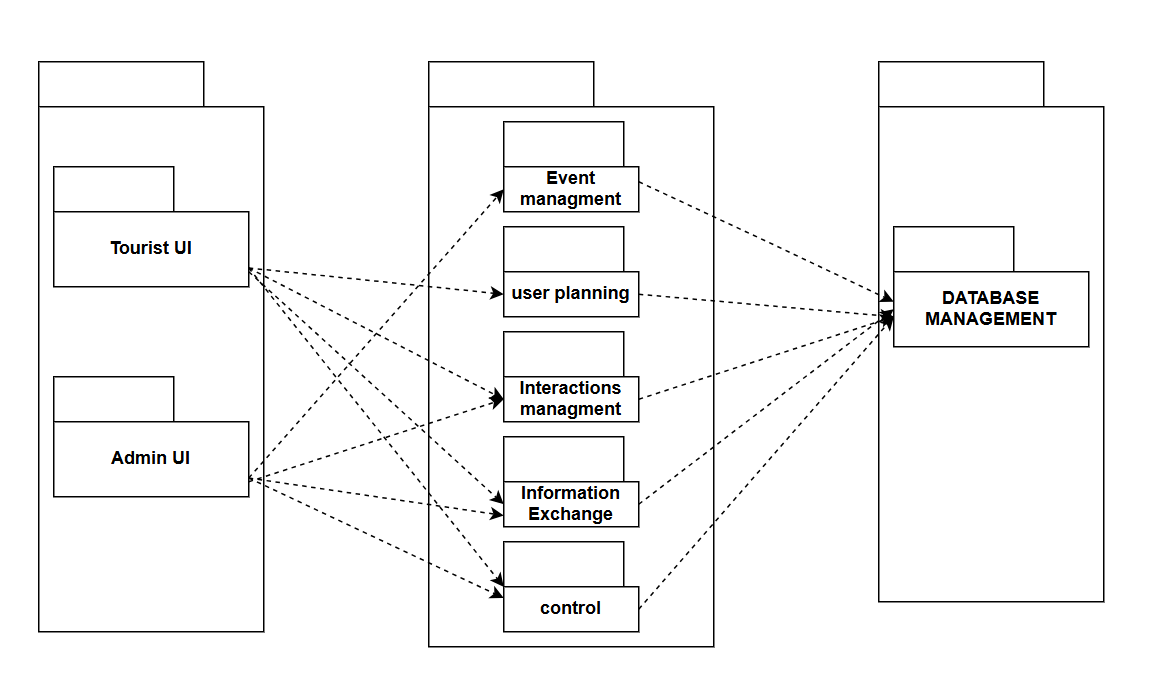


* 1. Three-Tier architecture representation

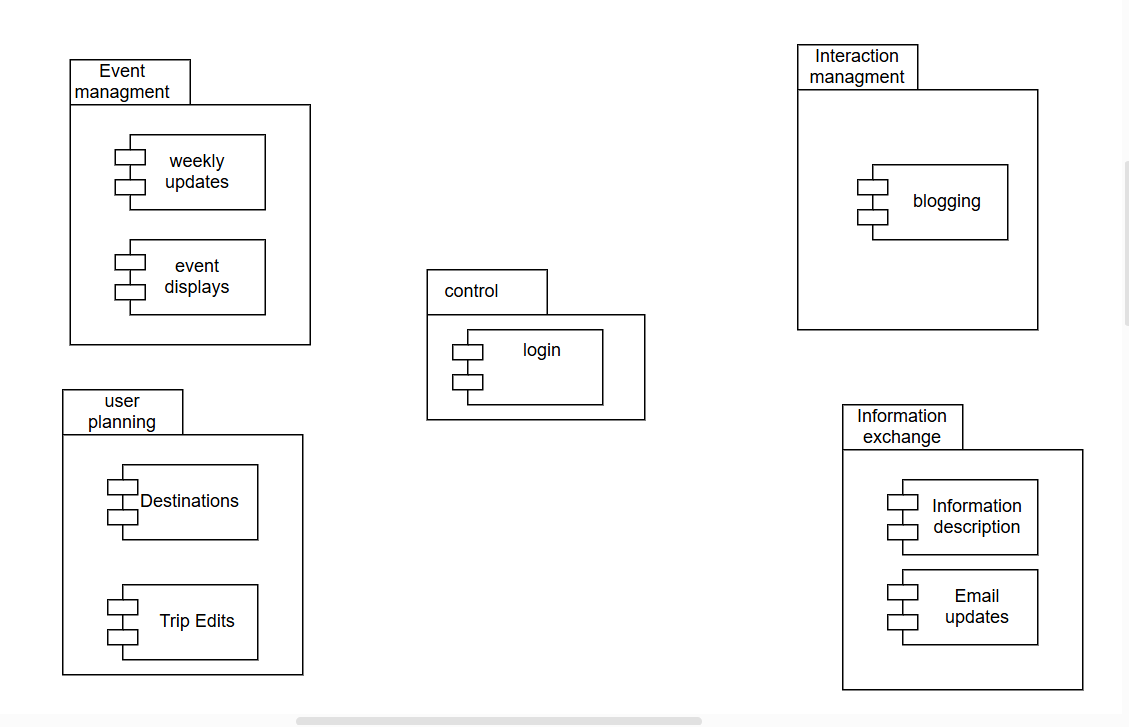
## 2.1 Subsystem decomposition



**Figure 1. Layer 1/Tier 1 Decomposition**

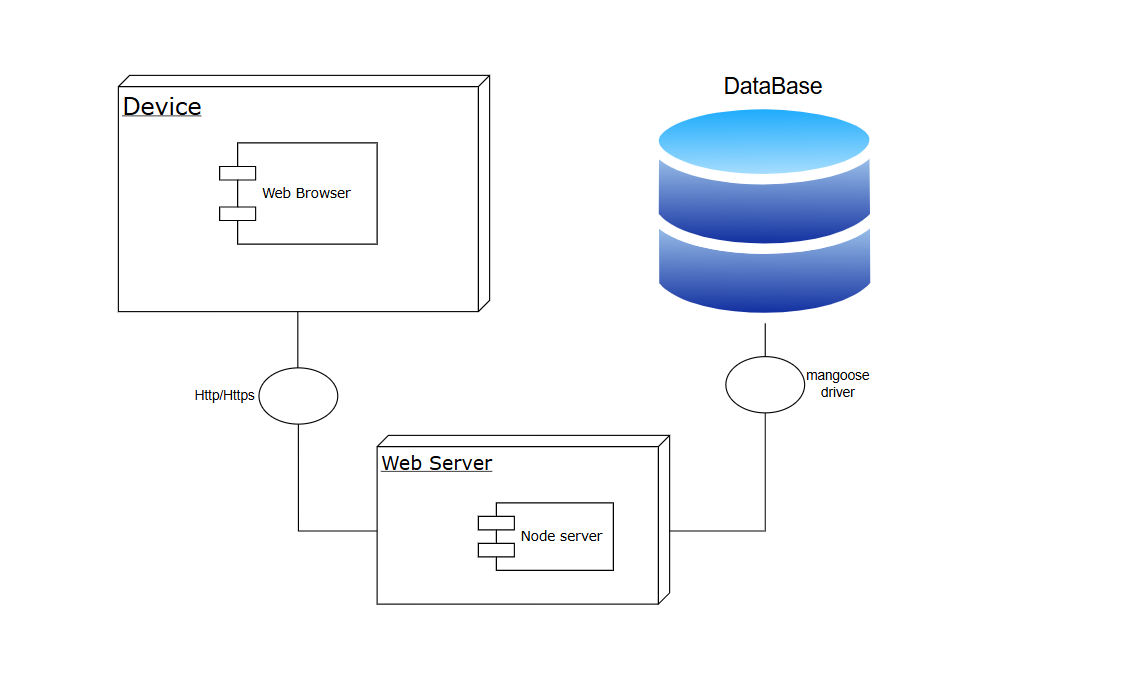


**Figure 3. Tier 2 Decomposition**

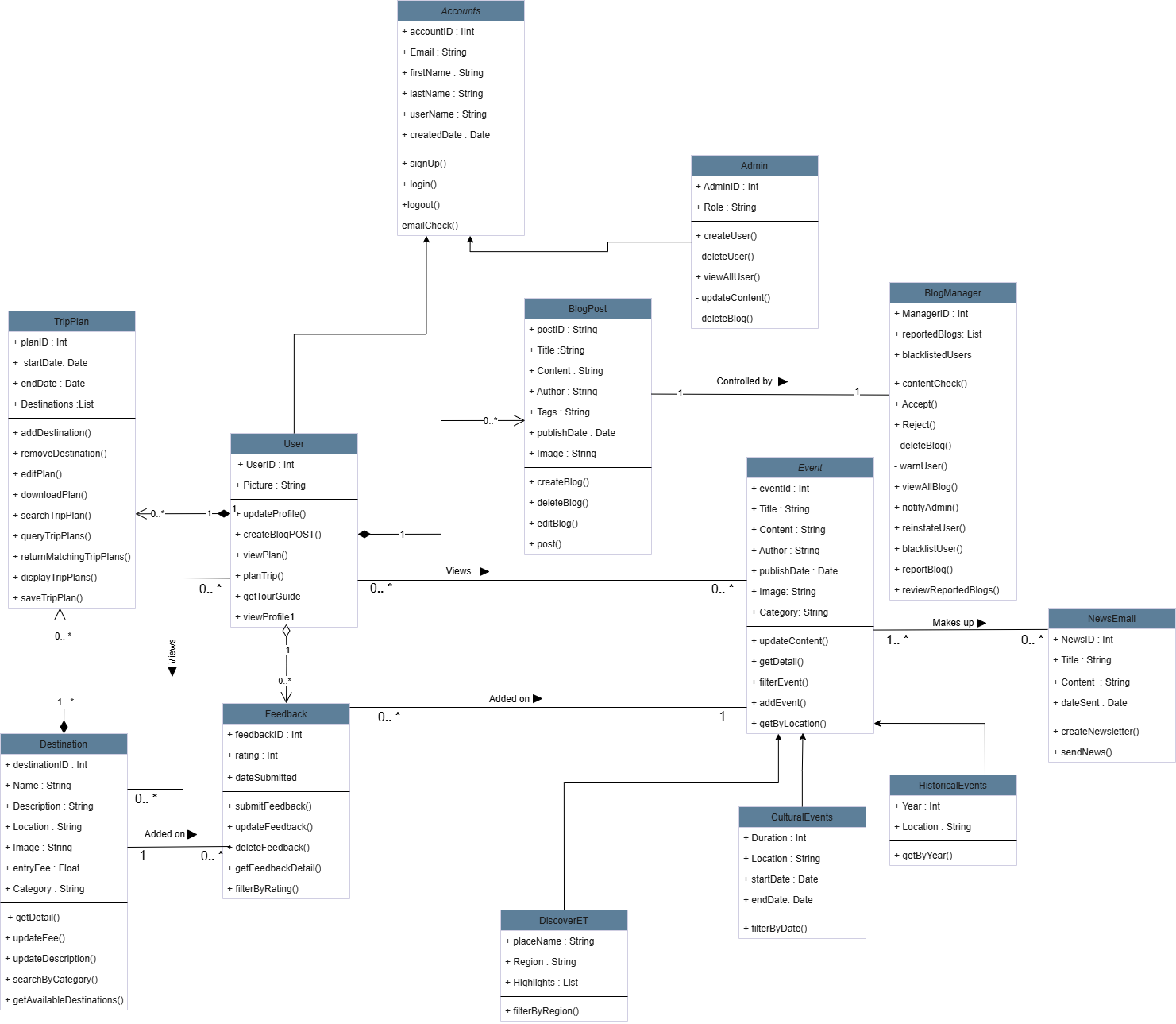


**Figure 4. Tier 3 decomposition**

## 2.2 Hardware/software mapping

**UML DEPLOYMENT DIAGRAM** 

**Figure 5. UML deployment diagram**



# 3.Object Model

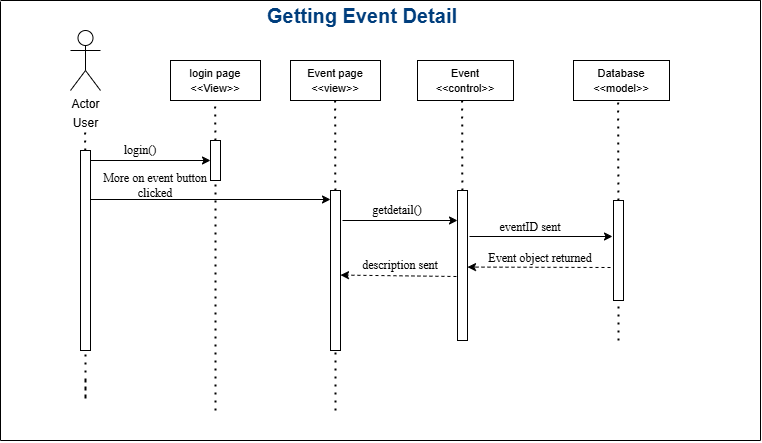
## 3.1 Class Diagram

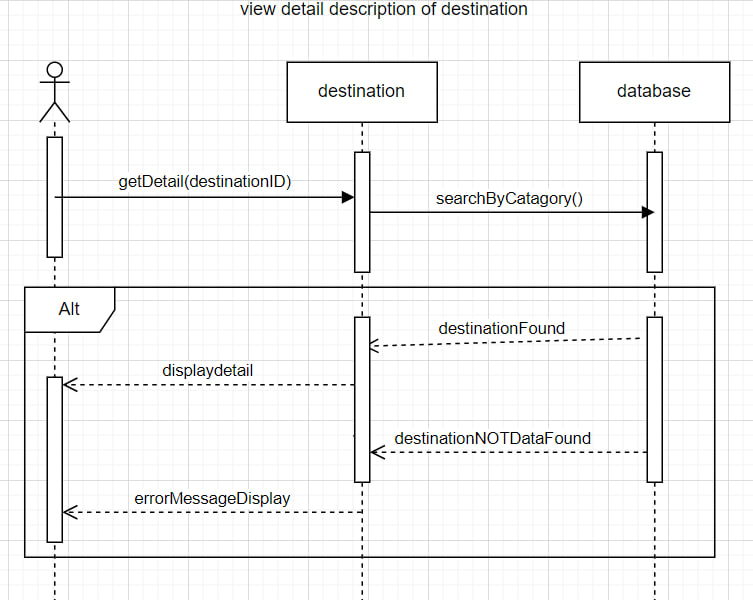
**Figure 6- Class Diagram**

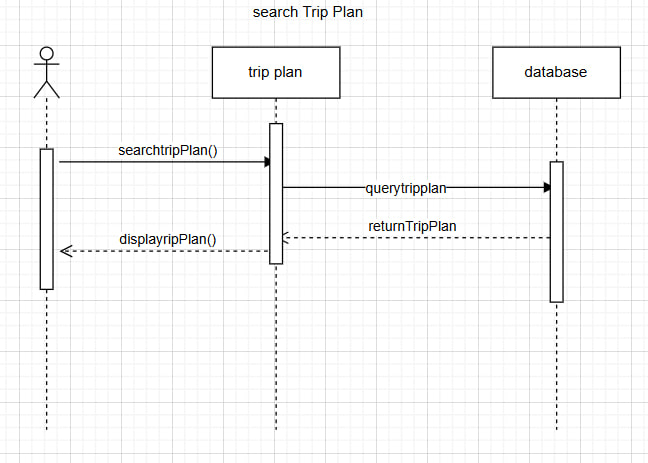
## 3.2 Sequence Diagram

#### Figure-7 - Sequence diagram of Adding a Destination

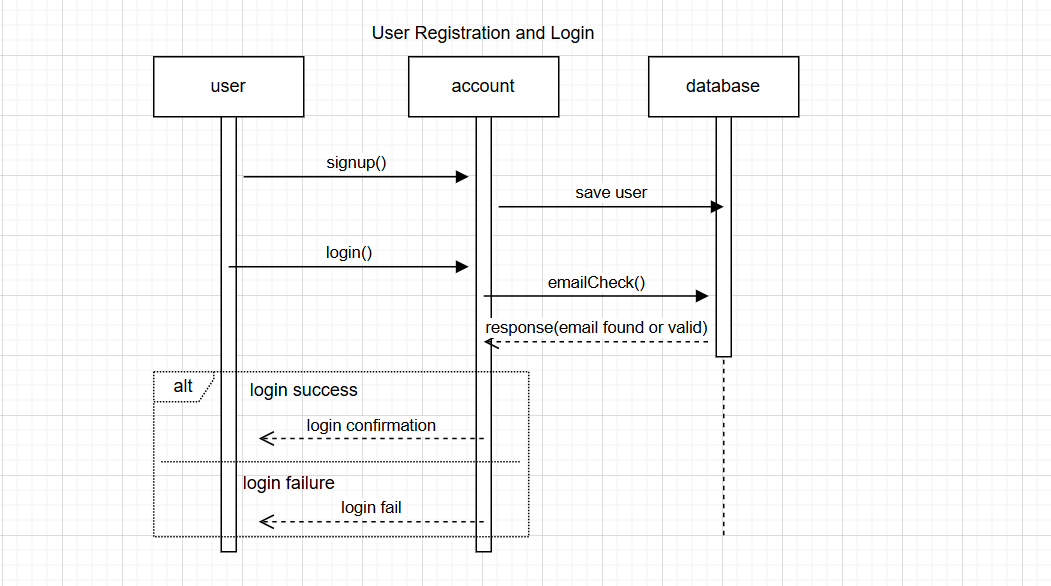
**Figure 8 - Sequence diagram of Blog post**

**Figure 9 - Sequence diagram of Getting Event Detail**

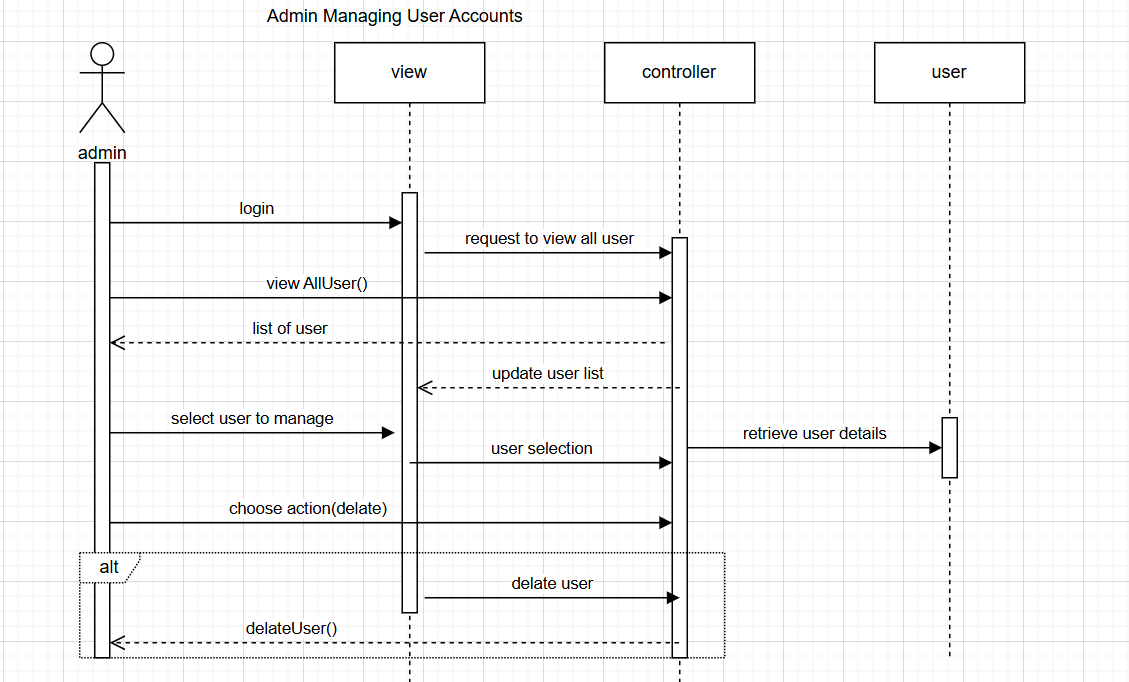
 **Figure 10- Sequence diagram of View detail description of destination diagram**



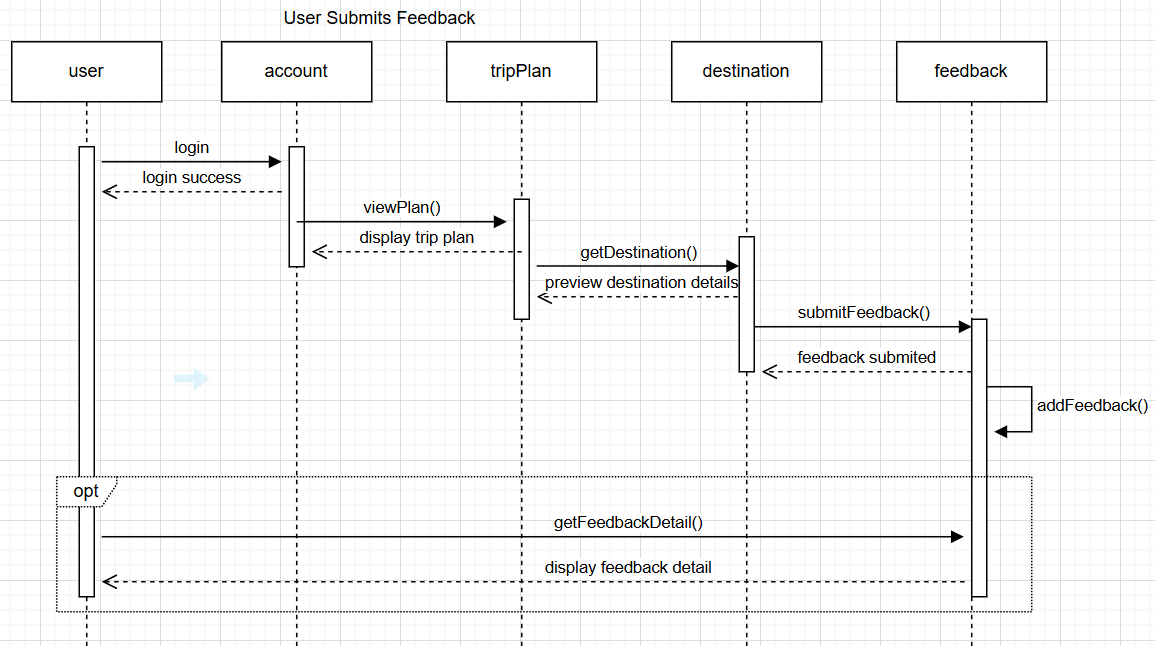
**Figure 11 - Sequence diagram of search trip plan**

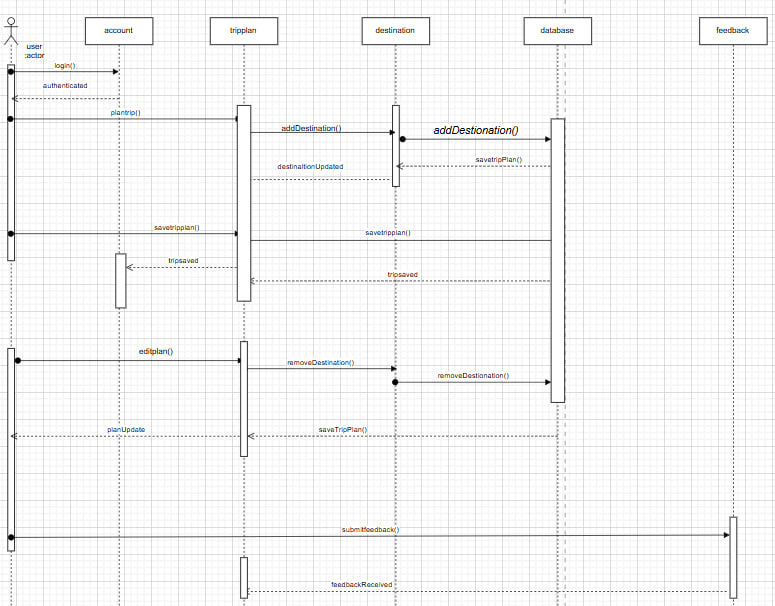


**Figure 12 - Sequence diagram of user registration and Login**



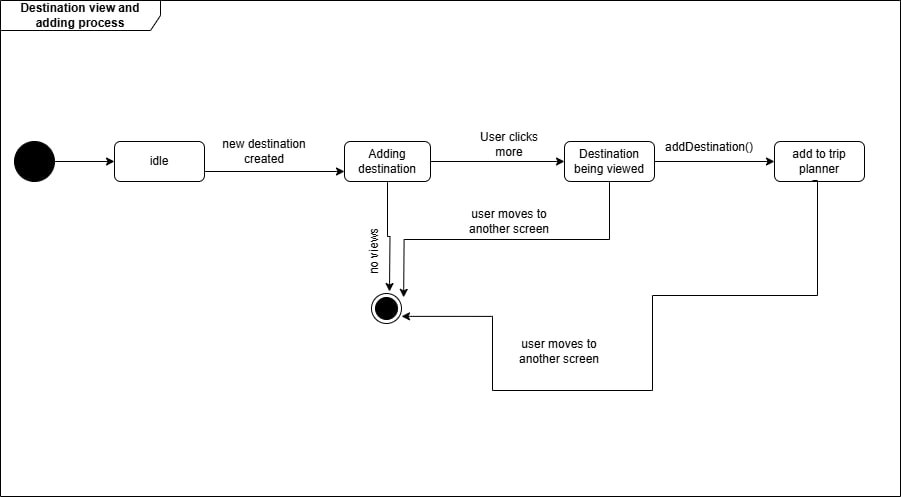
**Figure 13 - Sequence diagram of admin managing user accounts**

 **Figure 14 - Sequence diagram of user submits feedback**

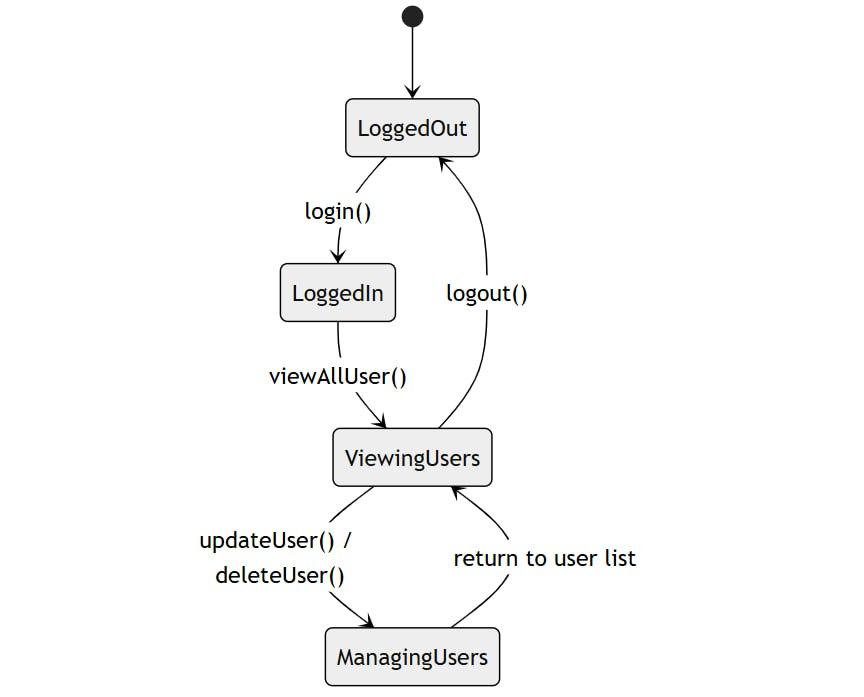


**Figure 15 - Sequence diagram of save and customize and give feedback on trip plan**

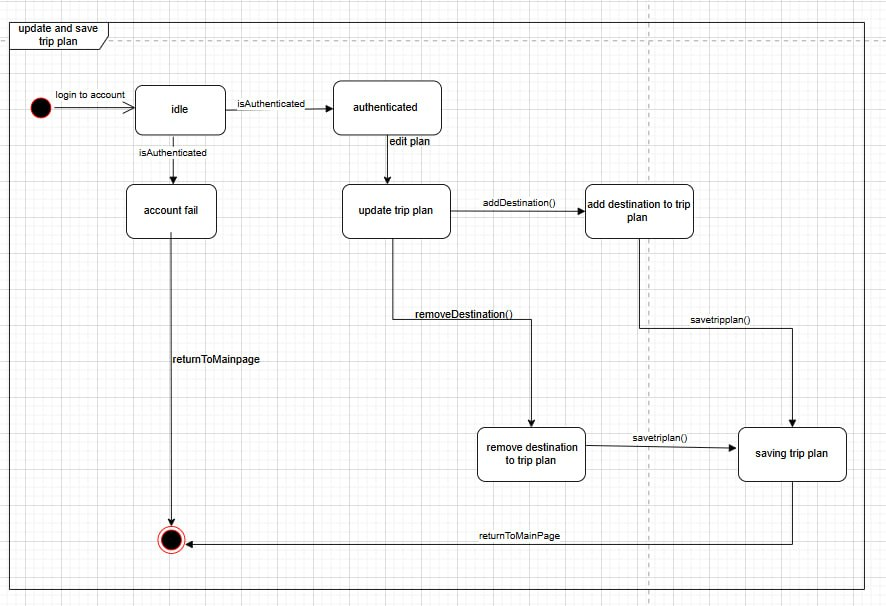
## 3.3 State Diagram



**Figure 15- State diagram of destinstion view and adding process**



**Figure 16- State diagram of admin managing user accounts**

**Figure 17- state diagram of update and save trip plan**

# 3.Detailed design

**Table 1: Event Class**

|  |
| --- |
| **Event** |
| + eventId : Int  + Title : String  + Content : String  + Author : String  + Category: String  + publishDate : Date |
| + updateContent()  + getDetail()  + filterEvent()  + addEvent()  + getByLocation() |

**Table 2 : Attribute Description for Events class**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| eventId | Int | Public | eventId <> NULL, it must be unique for each event and its auto-generated |
| Title | String | Public | Title <> NULL and must have at least 3 characters. |
| Content | String | Public | Content <> NULL and must have at least 20 characters. |
| Author | String | Public | Author <> NULL and must match a valid user. |
| Category | String | Public | Category <> NULL ,Must match predefined categories |
| Image | String | Public | Image can be null (Image == NULL), or if provided, it must be a valid URL or file path pointing to an image resource. |
| publishDate | Date | Public | publishDate <> NULL , Must be a valid date and not in the future. |

**Table 3 :Operation Description for Events class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| updateContent | Public | Void | eventId , content | The event with the given eventId must exist. | The content of the specified event is updated. |
| getDetail | Public | String | eventId | The event with the given eventId must exist. | Returns detailed information about the specified event. |
| filterEvent | Public | List | criteria(e.g category, publish date,  ) | Valid filter criteria must be provided. | Returns a list of events matching the filter criteria. |
| addEvent | Public | Event | eventId  Title  Content  Author  Category  publishDate | Valid event data must be given as input | A new event is added to the system and returned. |
| getByLocation | Public | List | location | Location parameter must be provided. and the location must exist in the system | Returns a list of events occurring in the specified location. |

**Table 4: HistoricalEvent class**

|  |
| --- |
| **HistoricalEvent** |
| + Year : Int  + Location : String  + Image : String |
| + getByYear() |

**Table 5 : Attribute Description for HistoricalEvent class**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| Year | Int | Public | Year <> NULL , Year must be a valid integer  and must represent a historical year meaning it must be less than current year |
| Location | String | Public | Location <> NULL , It must be a valid, non-empty string representing a place. |

**Table 6 :Operation Description for HistoricalEvent class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| getByYear | Public | List | year | year must be a valid integer and less than current year. | Returns a list of historical events that occurred in the specified year. |

**Table 7: CulturalEvent class**

|  |
| --- |
| **CulturalEvent** |
| + Duration : Int  + Location : String  + startDate : Date  + endDate: Date |
| + filterByDate() |

**Table 8 : Attribute Description for CulturalEvent class**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| Duration | Int | Public | Duration <> NULL ,Duration must be greater than 0 (Duration > 0) and represent the number of days. |
| Location | String | Public | Location <> NULL and must be a valid, non-empty string. |
| startDate | Date | Public | startDate <> NULL and must be a valid date object representing a date on or after the current date. |
| endDate | Date | Public | endDate <> NULL and endDate >= startDate |

**Table 9 :Operation Description for CulturalEvent class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| filterByDate | Public | List<CulturalEvent> | startDate | startDate must be a valid date and not null. | Returns a list of cultural events that start on the specified date. |

**Table 10: DiscoverET class**

|  |
| --- |
| **DiscoverET** |
| + placeName : String  + Region : String  + Highlights : List |
| + filterByRegion() |

**Table 11 : Attribute Description for DiscoverET class**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| placeName | String | Public | placeName <> NULL , it must be a non-empty string with a minimum length of 2 characters. |
| Region | String | Public | Region <> NULL , it must be a non-empty string with a minimum length of 2 characters. |
| Highlights | List<String> | Public | Highlights must be a non-null, non-empty list with at least one item, and each item must be a non-empty string. |

**Table 12 :Operation Description for DiscoverET class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| filterByRegion | Public | List<Places> | Region | Region must be a non-empty string and match an existing region in the system | returns any hosted news on the defined region |

**Table 13: NewsEmail class**

|  |
| --- |
| **NewsEmail** |
| + NewsID : Int  + Title : String  + Content : String  + dateSent : Date |
| + createNewsletter()  + sendNews() |

**Table 14 : Attribute Description for NewsEmail class**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| NewsID | Int | Public | eventId <> NULL ,must be unique and is auto-generated by the system or database. |
| Title | String | Public | Title <> NULL ,must be a non-empty string with a minimum length of 3 characters. |
| dateSent | Date | Public | dateSent <> NULL, must be a valid date, either the current date or a future date (cannot be a past date). |
| Content | String | Public | Content <> NULL, must be a non-empty string with a minimum length of 10 characters |

**Table 15 :Operation Description for NewsEmail class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| createNewsletter | Public |  | NewsID,  Titele,  Content, | NewsID must be unique; Title and Content must meet their respective invariants. | A new newsletter is created with the provided data and is stored in the system. |
| sendNews | Public | void | NewsID | NewsID must correspond to an existing newsletter in the system. | The newsletter with the specified NewsID is sent to all subscribed users. |

*Table 16: destination class*

|  |
| --- |
| *Destination* |
| *+destinationID: int +Name: String +Description: String +Location: String +Image: String +entryFee: Float +Category: String* |
| *+getDetail() +updateFee() +updateDescription() +searchByCategory() +getAvailableDestinations()* |

*Table 17: attribute description of destination class*

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | type | Visibility | Invariant |
| Destination ID | int | public | destinationID <> NULL, must be unique and auto-generated by the database. |
| Name | String | public | name <> NULL, must not be empty. |
| Description | String | public | description <> NULL, must not be empty |
| Location | String | public | |  | | --- | |  |  |  | | --- | | location <> NULL, must not be empty. | |
| Image | String | public | image <> NULL, must represent a valid image URL. |
| Entry\_ fee | Float | Public | Entry Fee <> NULL, must be greater than or equal to 0. |
| category | String | public | category <> NULL, must be a valid category (e.g. Historical, Scenic). |

Table 18: operation description of destination class

| **Method** | **Visibility** | **Return Type** | **Arguments** | **Pre-condition** | **Post-condition** |
| --- | --- | --- | --- | --- | --- |
| getDetail() | public | String | None | None | Returns destination details |
| updateFee() | public | void | fee: float | fee must be >= 0 | Updates the entry Fee |
| updateDescription() | public | void | description: String | description cannot be null | Updates the description |
| searchByCategory() | public | List<Destination> | category: String | category cannot be null | Returns destinations matching the category |
| getAvailableDestinations() | public | List<Destination> | None | None | Returns all destinations |

*Table 19: blog post class*

|  |
| --- |
| *Blog post* |
| *+postID: Int +Title: String +Content: String +Author: String +Tags: String +publishDate: Date +image: String* |
| *+createBlog() +deleteBlog() +editBlog()*  *+Post()* |

| **Attribute** | **Type** | **Visibility** | **Invariant** |
| --- | --- | --- | --- |
| Post ID | String | Public | postID <> NULL, auto-generated by the database, and must be unique |
| title | String | public | title <> NULL and must not be empty |
| content | String | public | content <> NULL and must not be empty |
| author | String | public | author <> NULL and must match a valid User ID |
| tags | String | public | Optional, can be NULL |
| Publish Date | Date | public | publishDate <> NULL and must represent a valid date |
| image | String | public | image <> NULL and must represent a valid image URL |

*Table 20: attribute description of blog post class*

| **Method** | **Visibility** | **Return Type** | **Arguments** | **Pre-condition** | **Post-condition** |
| --- | --- | --- | --- | --- | --- |
| createBlog() | public | void | content: String, title: String | Title and content are valid | Adds a new blog to the system |
| deleteBlog() | public | void | blogID: String | postID exists | Deletes the blog post |
| editBlog() | public | void | blogID: String, content: String, title: String | postID exists, content valid | Updates blog post content |
| Post() | public | Void | userID: int, blogContent: String, title: String, image: String | userID <> NULL and matches author. blogContent and title <> NULL and must not be empty cannot be null | Publishes the blog and sets publishDate to the current date |

*Table 21: operation description of blog post class*

*Table 22: trip plan class*

|  |
| --- |
| *Trip plan* |
| *+planID: int +startDate: Date +endDate: Date +Destinations: List<string>* |
| *+addDestination() +RemoveDestination() +editPlan() +downloadPlan()* |

*Table 23: attribute description of trip plan class*

| **Attribute** | **Type** | **Visibility** | **Invariant** |
| --- | --- | --- | --- |
| planID | int | public | planID <> NULL, must be unique and auto-generated by the database |
| startDate | Date | public | startDate <> NULL, must be a valid date. |
| endDate | Date | Public | endDate <> NULL, must be after startDate. |
| destinations | List<String> | Public | destinations <> NULL, must contain valid destination IDs. |

*Table 24: operation description of trip plan class*

| **Method** | **Visibility** | **Return Type** | **Arguments** | **Pre-condition** | **Post-condition** |
| --- | --- | --- | --- | --- | --- |
| addDestination() | public | void | destinationID: integer | destinationID <> NULL, must exist in the system. | Adds destination to plan |
| removeDestination() | public | void | destinationID: integer | destinationID <> NULL, must exist in the plan. | Removes destination |
| editPlan() | public | void | startDate: Date, endDate: Date, destinations | planID <> NULL, startDate < endDate. | Updates plan details |
| downloadPlan() | public | void | None | PlanID <> NULL , must exist | Returns and generates downloadable version |

*Table 25: blog manager class*

|  |
| --- |
| *Blog manager* |
| -managerID: int  -reportedBlogs: List<String> |
| +reviewReportedBlogs()  +reportBlog() +warnUser()  +deletBlog() |

*Table 26: attribute description of blog manager class*

| **Attribute** | **Type** | **Visibility** | **Invariant** |
| --- | --- | --- | --- |
| managerID | int | private | managerID <> NULL and must be a unique positive integer |
| reportedBlogs | List<String> | private | reportedBlogs <> NULL |

*Table 27: operation description of blog manager class*

| **Method** | **Visibility** | **Return Type** | **Arguments** | **Pre-condition** | **Post-condition** |
| --- | --- | --- | --- | --- | --- |
| reportBlog() | Public | Void | blogID: String | blogID <> NULL | Adds blogID to reportedBlogs |
| reviewReportedBlogs() | Public | List<String> | None | reportedBlogs <> NULL | Returns the list of all reported blogs |
| warnUser() | Public | Void | userID: Integer, message: String | userID <> NULL | Sends a warning to the specified user |
| deleteBlog() | Public | Void | blogID: String | blogID <> NULL | Removes the specified blog from the system |

Table 28: Account class

|  |
| --- |
| Account class |
| +accountID : int  +Email : string  +firstName : string  +lastName : string  +username: string  -Password : string  -createdDate : date |
| +signUp()  +logIn()  +logout()  +emailCheck() |

Table 29: Attribute Description for Account class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| AccountID | int | Public | Id<>NULL, is auto generated by the database and is unique |
| Email | string | Public | Email<>NULL  email.contains("@") AND email.contains(".")  Position of @ > 1  Position of (dot) > position of @ + 2  Position of (dot) + 3 <= total length of email address and the total character of the Email is at least 5 characters |
| firstName | string | Public | firstName<>NULL  firstName.matches("[A-Za-z]+")) |
| lastName | string | Public | lastName<>NULL  lastName.matches("[A-Za-z]+")) |
| userName | string | private | userName<>NULL, is unique and not already taken by another account |
| Password | string | public | Password<>NULL, Password should contain at least 8 characters |
| createdDate | Date | private | createdDate<>NULL, assigned the current date when the account is created and cannot change afterward |

Table 30: Operation Description for Account class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| signUp | public | boolean | account information | The account information must not already exist (username and email should be unique) | The account information should exist (a new account is successfully created) |
| logIn | public | boolean | Email/userName password | The account information must exist (username must be registered)  The provided password must match the stored password | The user should be logged in (the session is active) |
| logOut | public | void | None | The user must be logged in (a session must be active) | The user should be logged out (the session is terminated) |
| emailCheck | public | boolean | Email | The email must not be empty | The email validity should be confirmed (returns true if valid and not already used) |

Table 31: User class

|  |
| --- |
| User class |
| +userID : int  +userName : string  +picture : string |
| +updateProfile()  +createBlogPost()  +viewPlan()  +planTrip()  +getTourGuide()  +viewProfile() |

Table 32: Attribute Description for user class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| userID | integer | public | ID<>NULL, is auto generated by the database and is unique |
| picture | string | public | picture is valid, picture can be empty |

Table 33: Operation Description for user class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Type | Visibility | Argument | Pre-Condition | Post Condition |
| updateProfile | boolean | public | firstName  lastName  Email  Picture(optional) | The user’s information shouldn’t meet the requirements | The user’s information should meet the requirements |
| createBlogPost | boolean | public | Title  Content  Tags(optional) | The blog post’s title and content shouldn’t be empty | The blog post should be created successfully |
| viewPlan | plan | public | planID | The specified plan should exist | Details of the specified plan are retrieved |
| planTrip | boolean | public | Destination startDate endDate | The trip’s destination should not be empty; the start date should be before the end date | The trip should be planned successfully |
| getTourGuide | TourGuide | public | None | The user must be logged in (userID must exist) | The tour guide information should be retrieved successfully |
| viewProfile | user | public | None | The user must be logged in (userID must exist) | The user’s profile information should be retrieved successfully |

Table 34: Admin class

|  |
| --- |
| Admin class |
| +adminID  +role |
| +createUser()  -deleteUser()  +viewAllUser()  -updateContent()  -deleteBlog() |

Table 35: Operation Description for Admin class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| adminID | Integer | Public | ID<>NULL, is auto generated by the database and is unique |
| role | String | Public | role<>NULL  role is valid |

Table 36: Operation Declaration for Admin class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Type | Visibility | Argument | Pre-Condition | Post Condition |
| createUser | Boolean | Public | firstName  lastName  Email  username  password | Admin must be authenticated; email and username must not already exist | A new user account should be created successfully |
| deleteUser | Boolean | Private | userID | The admin must be authenticated and have permission to delete users | The user account should be deleted successfully |
| viewAllUser | List<user> | Public | None | The admin must be authenticated and have permission to view user accounts | A list of all user accounts should be retrieved successfully |
| updateContent | Boolean | private | contentID  newContent | The admin must be authenticated and have permission to update content | The specified content should be updated with newContent |
| deleteBlog | Boolean | private | blogID | The admin must be authenticated and have permission to delete blogs | The specified blog should be deleted successfully |

Table 37: feedback class

|  |
| --- |
| Feedback class |
| +feedbackID  +rating  +dataSubmitted |
| +submitFeedback()  +updateFeedback()  +delateFeedback()  +getFeedbackDetail()  +filterByRating() |

Table 38: Operation Description for FeedBack class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| feedbackID | Integer | Public | ID<>NULL, is auto generated by the database and is unique |
| rating | Integer | Public | rating >= 1 AND rating <= 5 |
| dateSubmitted | Date | Public | dateSubmitted <= currentDate |

Table 39: Operation Description for FeedBack class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Type | Visibility | Argument | Pre-Condition | Post Condition |
| submitFeedback | Void | Public | Rating  feedbackText | The feedback does not already exist in the system for the associated entity | Creates a new feedback entry |
| updateFeedback | Void | Public | feedbackID newRating  newFeedbackText | The user has permission to modify the feedback | Updates the specified feedback entry |
| delateFeedback | Void | Public | feedbackID | The user has permission to delete the feedback | Deletes the specified feedback entry |
| getFeedbackDetail | Feedback | Public | feedbackID | Feedback exists | Returns the details of the specified feedback entry |
| filterByRating | List<Feedback> | Public | rating | Feedback with the specified rating exists | Returns feedback entries matching the specified rating |

**References**

[**https://www.geeksforgeeks.org/three-tier-client-server-architecture-in-distributed-system/**](https://www.geeksforgeeks.org/three-tier-client-server-architecture-in-distributed-system/)**,geeks for geeks.**

[**https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/**](https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/)**,geeks for geeks.**

[**https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/**](https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/)**,geeks for geeks.**

[**https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/**](https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/)**,geeks for geeks.**