CS-476 PROJECT

MARCH 20, 2022

University of Regina

Meet Our Team !!

- Abhishek
- Lalith Souri
- Jovanjeet
- Husanpreet
- Ritika

Outline

- Problem definition
- Use Case Diagram-1
- Use Case Diagram-2
- MVC Architecture Diagram
- Code within Web framework
- Factory Design Pattern
- Observer Design Pattern
- Our Website (Demo)
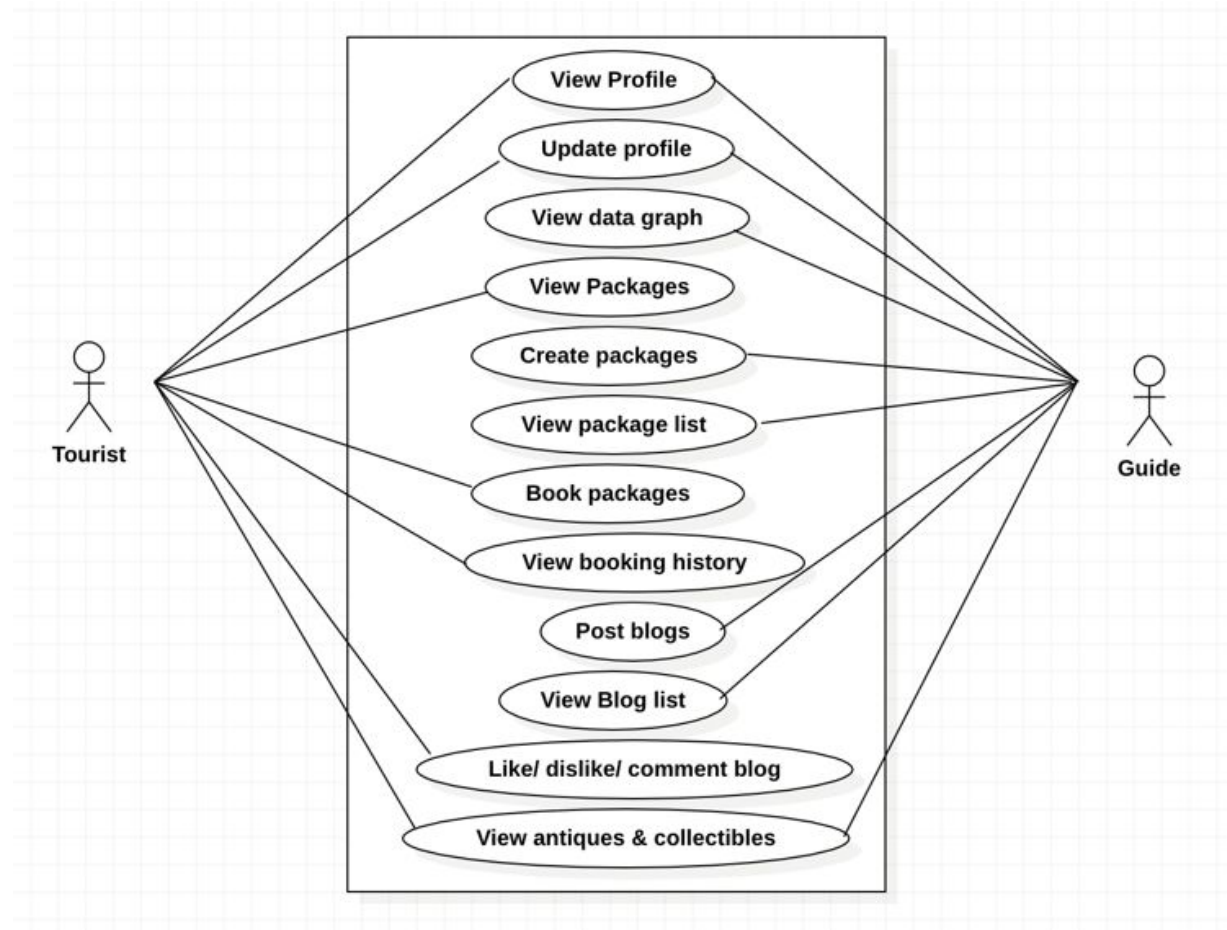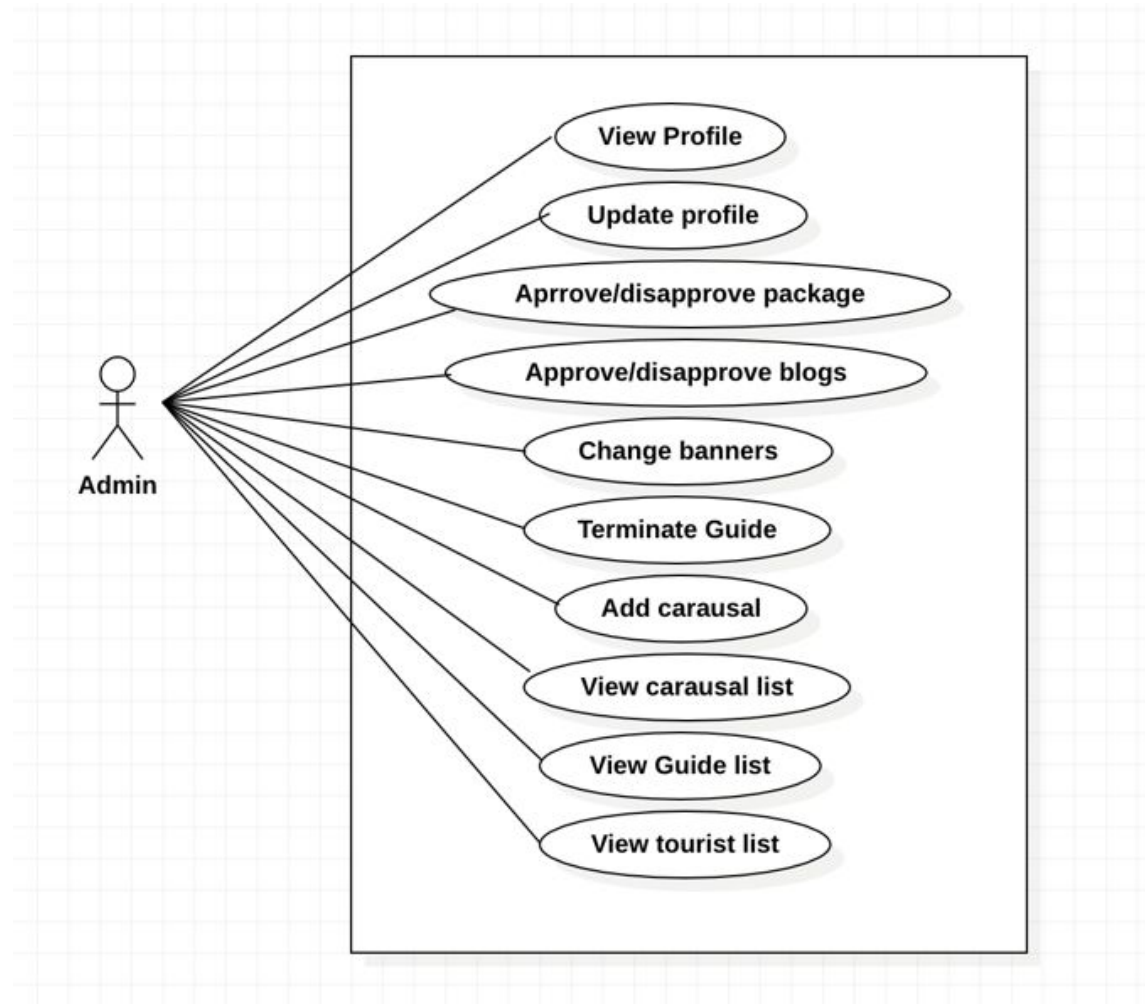- Questions?
- The end

Problem definition

- Provides comprehensive tourist information on alberta.
- User-friendly design, that is intuitive and straightforward.
- Features reviews and recommendations from experts
- Regularly updated website with updated information acting as a relevant and useful resource for travellers.
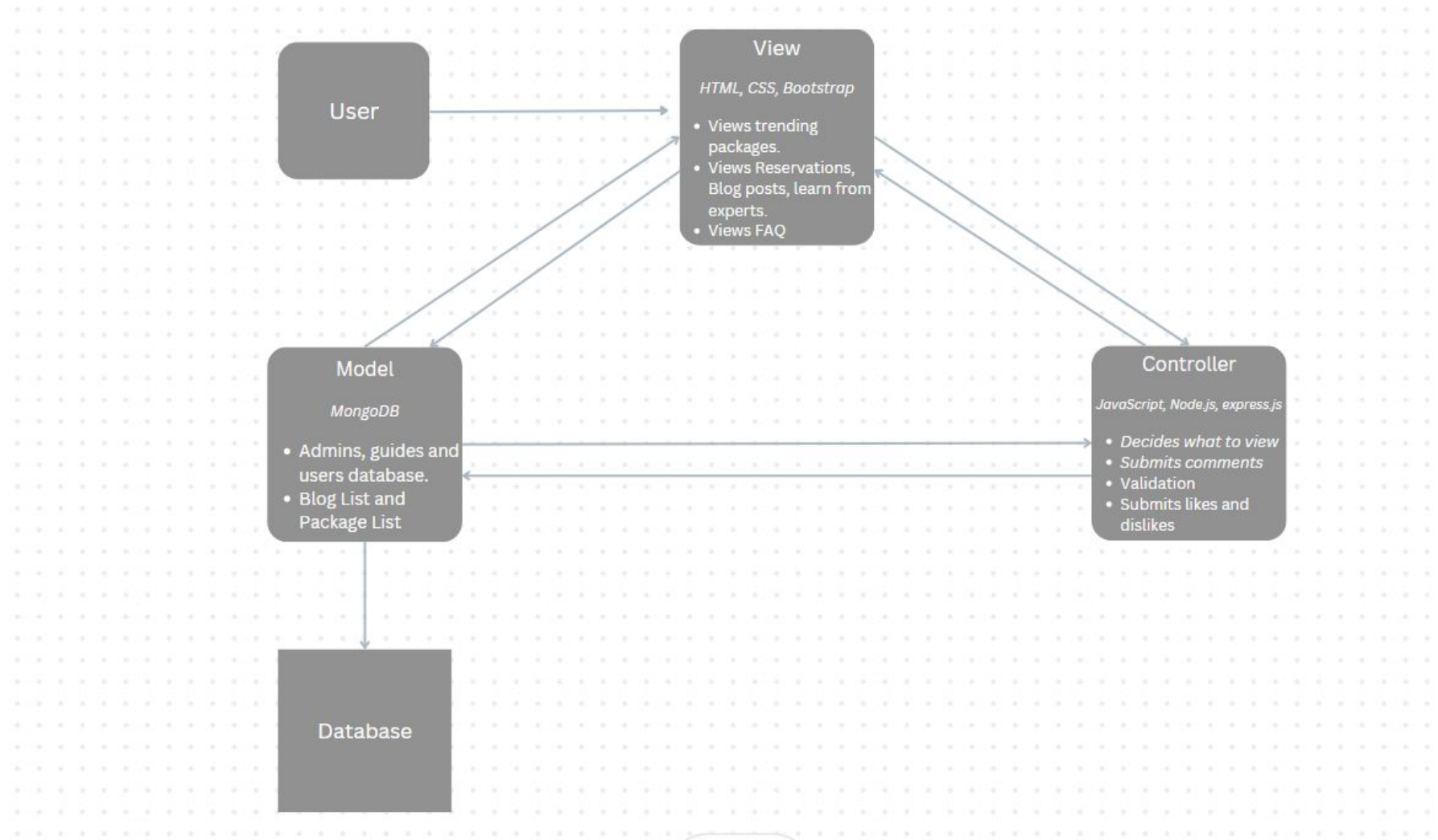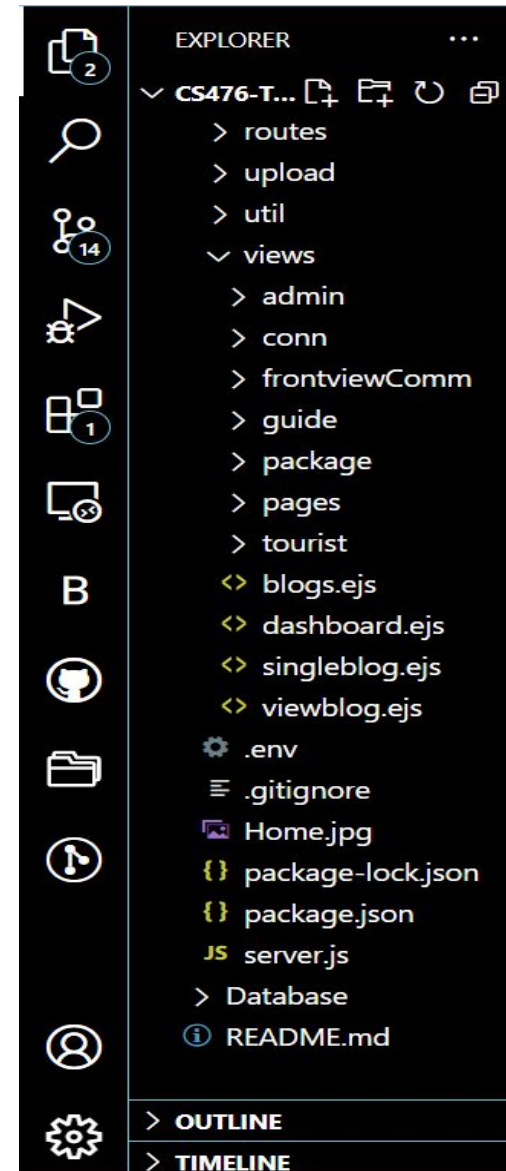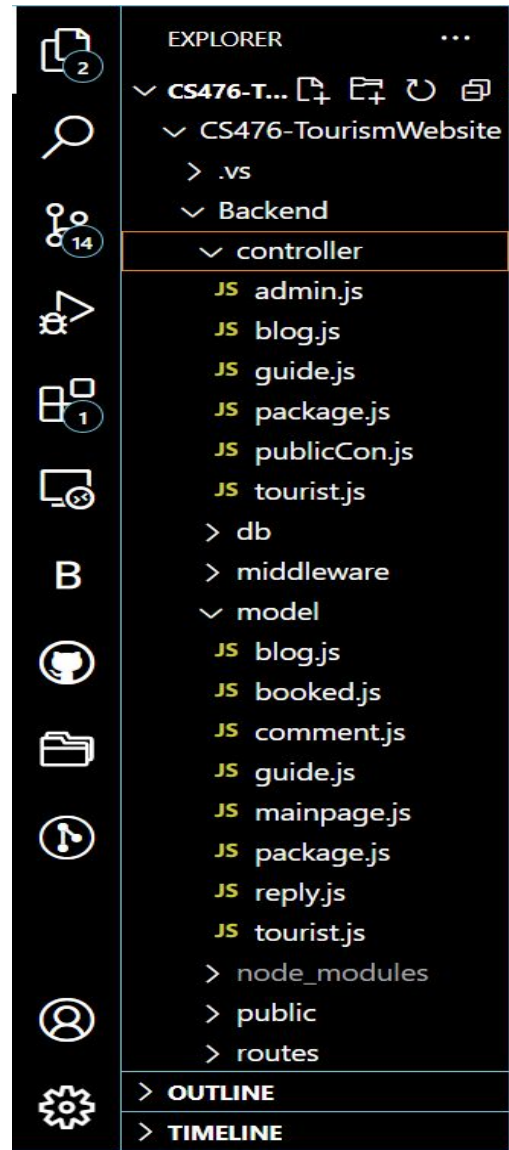
Use Case Diagram-1
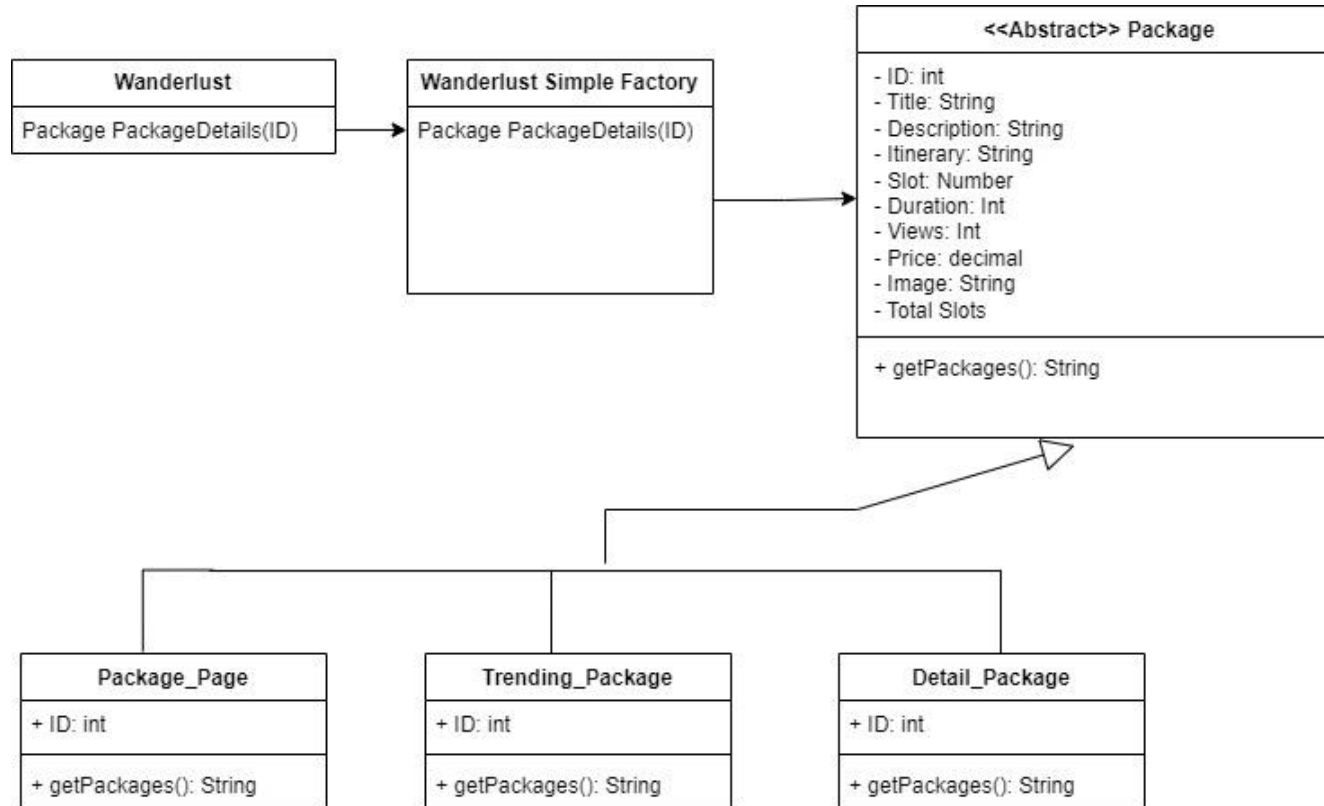
Use Case Diagram-2

MVC Architecture Diagram

Code within Web framework

EXPLORER   ...

∨ CS476-T... 📄 📁 ↻ ⊟

  ∨ CS476-TourismWebsite
    > .vs
    ∨ Backend
      ∨ controller
        JS admin.js
        JS blog.js
        JS guide.js
        JS package.js
        JS publicCon.js
        JS tourist.js
      > db
      > middleware
      ∨ model
        JS blog.js
        JS booked.js
        JS comment.js
        JS guide.js
        JS mainpage.js
        JS package.js
        JS reply.js
        JS tourist.js
      > node_modules
      > public
      > routes

> OUTLINE
> TIMELINE

EXPLORER   ...

∨ CS476-T... 📄 📁 ↻ ⊟

  > routes
  > upload
  > util
  ∨ views
    > admin
    > conn
    > frontviewComm
    > guide
    > package
    > pages
    > tourist
    <> blogs.ejs
    <> dashboard.ejs
    <> singleblog.ejs
    <> viewblog.ejs
  ⚙ .env
  ≡ .gitignore
  🖼 Home.jpg
  {} package-lock.json
  {} package.json
  JS server.js
  > Database
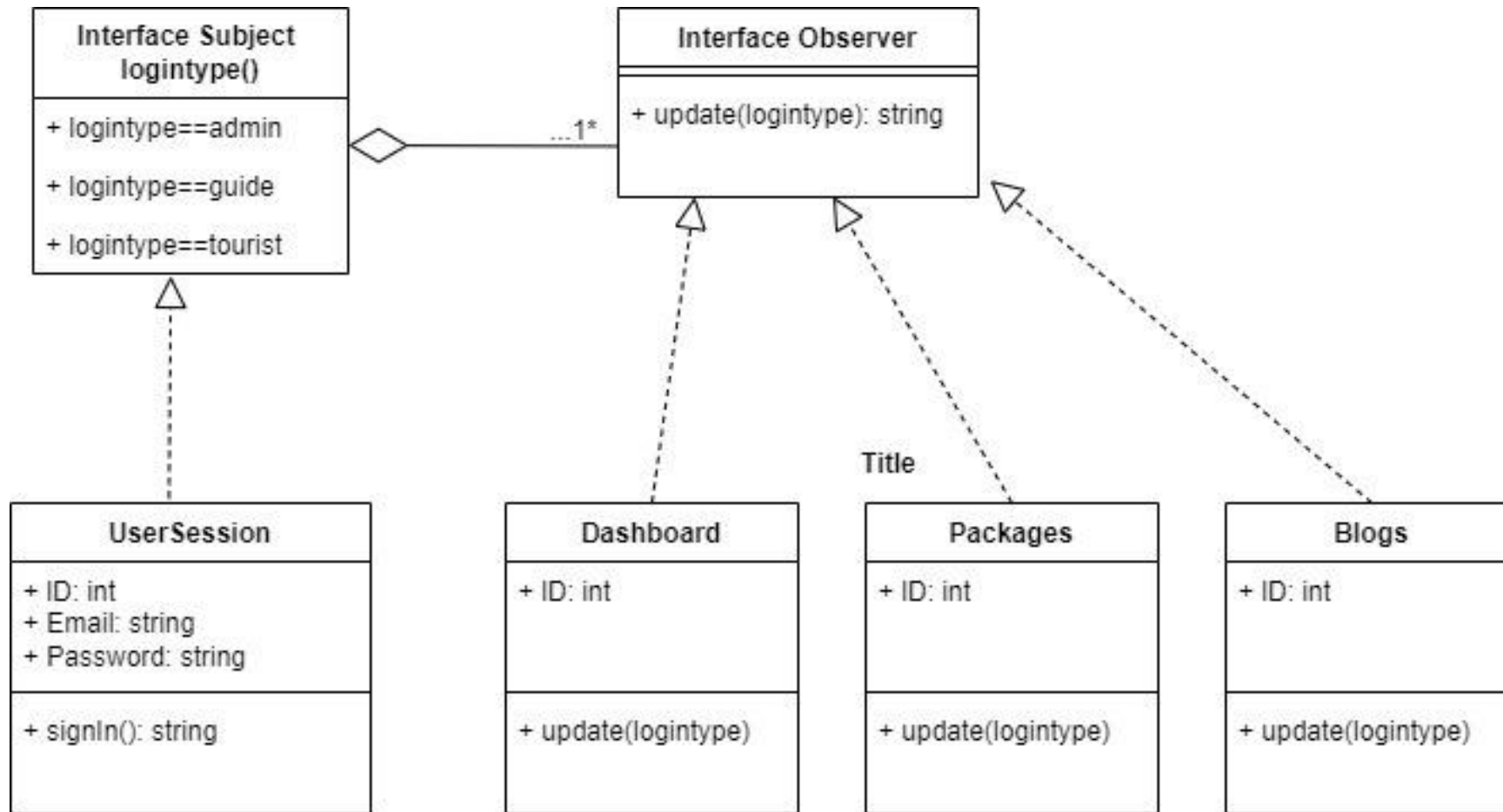 ⓘ README.md

> OUTLINE
> TIMELINE

Factory Design Pattern

Factory Design Pattern

- **Usability:** We created methods to save, update, view and delete the packages. These packages are the methods and we used incremental loop objects to retrieve the data from the backend to display on the page.
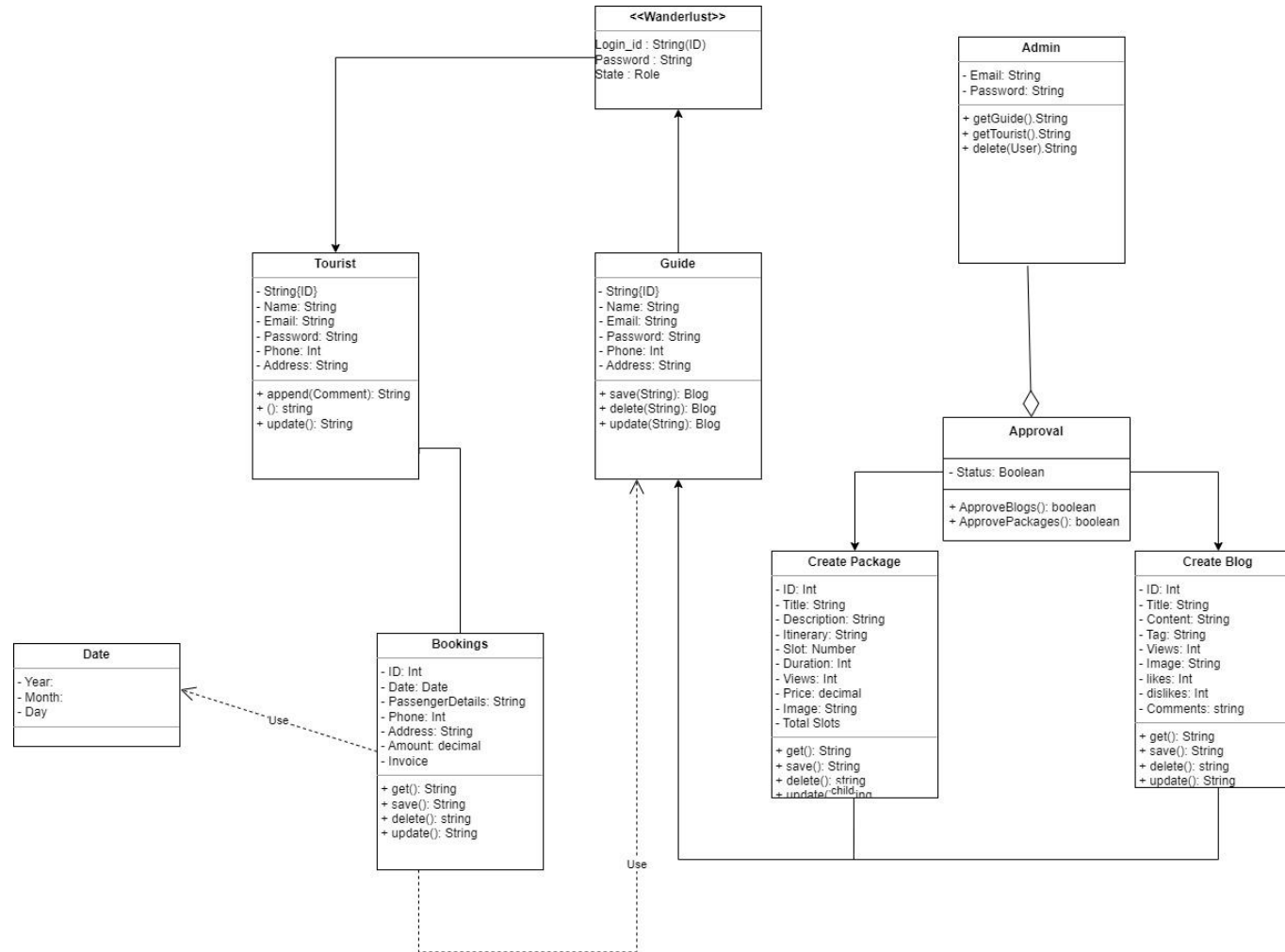
Observer Design Pattern
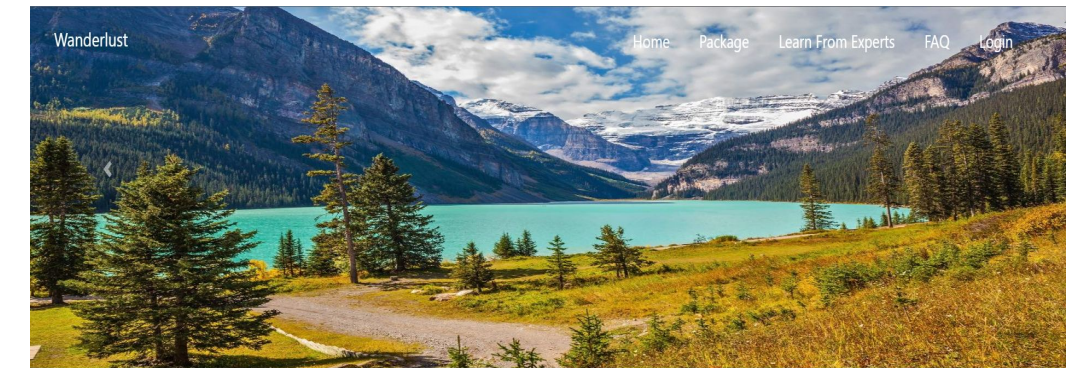
Observer Design Pattern

- **Usability:** Admin, Tourist and Guide all have different views of their dashboards. The functionality of all the dashboards is different for all the user roles. The administrator of the website has been given more access to maintain and keep up to the security standards of the website. Guide can post the packages and blogs. Tourists can view their reservations and invoices. In this way, the observer pattern gives the flexibility to all the user roles to perform the actions as per their requirements.
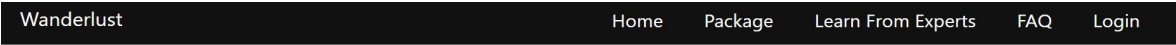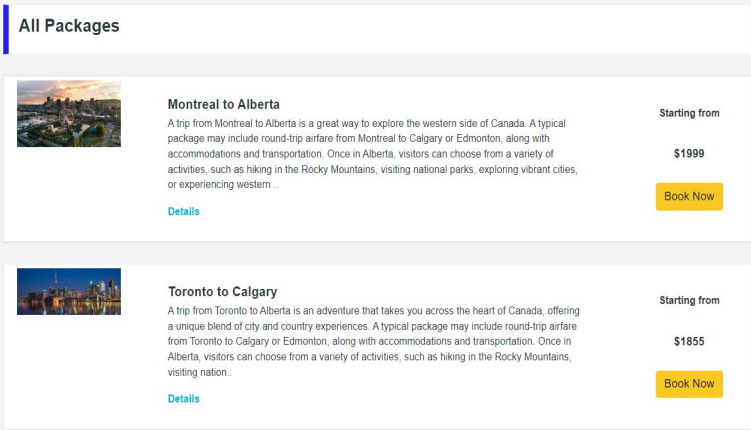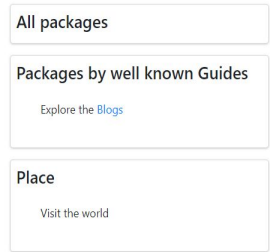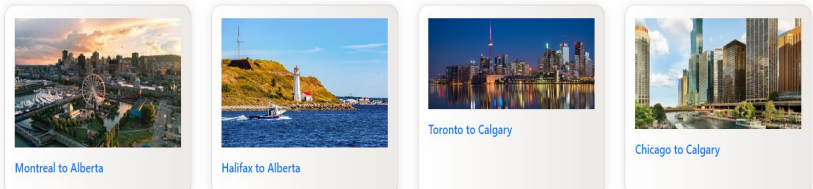
Class Diagram

<<Wanderlust>>

Login_id : String(ID)
Password : String
State : Role

Admin

- Email: String
- Password: String

+ getGuide().String
+ getTourist().String
+ delete(User).String

Tourist

- String{ID}
- Name: String
- Email: String
- Password: String
- Phone: Int
- Address: String

+ append(Comment): String
+ (): string
+ update(): String

Guide

- String{ID}
- Name: String
- Email: String
- Password: String
- Phone: Int
- Address: String

+ save(String): Blog
+ delete(String): Blog
+ update(String): Blog

Approval

- Status: Boolean

+ ApproveBlogs(): boolean
+ ApprovePackages(): boolean

Date

- Year:
- Month:
- Day

Use

Bookings

- ID: Int
- Date: Date
- PassengerDetails: String
- Phone: Int
- Address: String
- Amount: decimal
- Invoice

+ get(): String
+ save(): String
+ delete(): string
+ update(): String

Create Package

- ID: Int
- Title: String
- Description: String
- Itinerary: String
- Slot: Number
- Duration: Int
- Views: Int
- Price: decimal
- Image: String
- Total Slots

+ get(): String
+ save(): String
+ delete(): string
+ update(): string

Create Blog

- ID: Int
- Title: String
- Content: String
- Tag: String
- Views: Int
- Image: String
- likes: Int
- dislikes: Int
- Comments: string

+ get(): String
+ save(): String
+ delete(): string
+ update(): String

Use

Our Website (Demo)

Questions?

Thank you!!