**CS 491 SDD**

**CepGuide**

Submitted by:

*Serhat Tetik*

*Ridvan Akbulut*

*Orhan Taylan Sawaf*

*Özkan Özkan*

Table of Contents

[**1. INTRODUCTION** 2](#_Toc154519212)

[**2. SOFTWARE DESIGN** 2](#_Toc154519213)

[**2.1.  SOFTWARE ARCHITECTURE** 2](#_Toc154519214)

**2.2 TOPOLOGY** ……………………………………………………………………………………………………….2

[**2.3.  MODULE STRUCTURE** 3](#_Toc154519215)

[**2.4.  INTERFACE DESIGN** 4](#_Toc154519216)

**2.5 CLASS DIAGRAMS** ……………………………………………………………………………………………5

[**2.6.  SEQUENCE DIAGRAMS** 6](#_Toc154519217)

[**2.7.  USER INTERFACE (UI) MOCKUPS** 7](#_Toc154519218)

[**2.8. DATA DESIGN** 9](#_Toc154519222)

[**2.9. TOOLS and TECHNOLOGY**](#_Toc154519223) 9-10

[**3. PROJECT SCHEDULE** 11](#_Toc154519224)

**1. INTRODUCTION**

The travel and tourism industry, while a source of adventure and cultural exchange, often presents challenges in terms of navigation, understanding local history, and finding authentic experiences. Each year, millions of tourists worldwide face difficulties in accessing reliable and comprehensive information about their destinations, leading to missed opportunities and a superficial understanding of the places they visit.

Furthermore, the global tourism market, valued at over $8 trillion, indicates a massive demand for enhanced travel experiences. However, this demand is often met with fragmented information and a lack of integration between historical insights and local activities. Current solutions do not fully address the need for a cohesive and enriching travel experience that combines ease of access to local knowledge with the convenience of booking experiences.

These observations have inspired the development of PocketGuide, a mobile application designed to transform the way travelers interact with their destinations. Recognizing that over 60% of travelers express a desire to immerse themselves in local culture and history, our app aims to fill this gap by providing a comprehensive, user-friendly platform. With features like detailed historical information and easy booking options, the app is positioned to enhance the travel experience significantly.

Our approach is grounded in the understanding that travel is more than just visiting a place; it's about experiencing and understanding the essence of the destination. CepGuide leverages advanced technologies such as AI-driven personalization and location-based services to offer a unique and immersive travel guide. Our goal is to empower travelers with knowledge and options, making each journey an enriching and memorable experience.

**2. SOFTWARE DESIGN**

**2.1.  SOFTWARE ARCHITECTURE**

Our system employs a three-tier Client-Server Architecture that offers excellent scalability; this allows us to effortlessly adjust our computing resources to meet demand. Its modular design ensures maintainability, as each tier can be managed and debugged independently. The architecture's inherent flexibility means that we can implement upgrades or changes in one tier without disrupting the others. Moreover, the layered approach enhances security, significantly diminishing the likelihood of cyber-attacks.

**2.2.  TOPOLOGY**  
  
A diagram of a server

Description automatically generated with low confidence

**2.3.  MODULE STRUCTURE**

A screenshot of a diagram

Description automatically generated

**2.4.  INTERFACE DESIGN**

1- We will have a User Rest:

Methods provided by this rest:

- PostUser(Information): Save the given user information to the database.

- PostUser(Username, Password): If there is a user matching the given username and password, return success.

- GetUser(UserID): Return user information matching the given UserID.

- GetUsers(): Return a list of all users.

2- Adoption List Rest:

Methods provided by this rest:

- GetAdoption(): Return the entire adoption list in list format.

- GetAdoption(UserID): Return all adoption lists for the given userID.

- PostAdoption(UserId): Add an adoption list to the database with the given userID.

- DeleteAdoption(UserId, AdId, AdoptionId): Delete information from the adoption list and ad list based on the parameters.

3- Lost List Rest:

Methods provided by this rest:

- GetLost(): Return the entire Lost list in list format.

- GetLost(UserID): Return all Lost lists for the given userID.

- PostLost(UserId): Add a Lost list to the database with the given userID.

- DeleteLost(UserId, AdId, AdoptionId): Delete information from the Lost list and ad list based on the parameters.

4- Cities Rest:

Methods provided by this rest:

- GetCities(): Return all cities in list format.

5- Districts Rest:

Methods provided by this rest:

- GetCities(CityId): Return districts associated with the given CityId.

6- Admin Rest:

Methods provided by this rest:

- PostAdmin(Information): Save the given admin information to the database.

- PostAdmin(Username, Password): If there is an admin matching the given username and password, return success.

- GetAdmin(UserID): Return admin information matching the given AdminID.

- GetAdmin(): Return a list of all admins.

7- Pet Rest:

Methods provided by this rest:

- PostPet(Information): Save the given pet information to the database.

- GetPet(AdID): Return pet information matching the given AdID.

- GetPet(PetID): Return pet information matching the given PetID.

- DeletePet(AdId, PetId): Delete information from the Pet data based on the parameters.

8- AI Rest:

Methods provided by this rest:

-PostImage(Image):Save the given pet’s photo and the return breed of pet .

**2.5.  CLASS DIAGRAMS**

A diagram of a computer

Description automatically generated with medium confidence

**2.6.  SEQUENCE DIAGRAMS**

A diagram of a diagram

Description automatically generated with medium confidence

**2.7.  USER INTERFACE (UI) MOCKUPS**

**HomePage:**

****

Web -Adoption Page:

Web-Admin Page:

**2.8. DATA DESIGN**

A diagram of a computer program

Description automatically generated

**2.9. TOOLS and TECHNOLOGY**

**Web Page**

The language we will use to develop the website is React. To mention the reasons why we chose this language, we can create scripts in a simplified way, thus making your code simpler and cleaner and making it work faster. With its component-based architecture, React.js uses a component-based architecture that allows creating reusable components for the user interface. This makes applications easier to maintain and scale. Additionally, the tools we are considering using are React Bootstrap and Material UI because we can find a lot of resources and documents.

**Mobil App**

Flutter allows you to create cross-platform apps that provide native performance. Apps created with Flutter feature beautiful and intuitive design and are able to run animations smoothly. Flutter also increases mobile development speed, helping lower costs.Also, we will use the State Management Library which is named GetX. The reason we use GetX is to facilitate state management and routing, providing a fast development process.

**Database**

We will use firebase to store data.Advantage of Firebase is its secure & fast hosting services. Firebase hosting supports all content types, including web applications, dynamic and static content. Moreover, whether you want to host your Express.js microservices, HTML, CSS, or APIs, the hosting support of Firebase is always there. It means Firebase hosts a diverse variety of content.

**Server**

We will use Node.js technology on the server side. We chose Node.js because our members who will be writing the backend have previous experience with this technology.Also, Node.js is the go-to choice for developers seeking a high-performance, scalable, and unified development experience. Built on Chrome’s V8 JavaScript engine, Node.js ushers JavaScript into the realm of server-side programming.

**AI**

We will use Python technology on the AI side. We saw Python as the most ideal programming language for artificial intelligence due to its simplicity. We researched that Python, which we think is easier to learn and has a simple syntax, is also extremely suitable for the implementation of artificial intelligence algorithms. It also supports object-oriented, functional, and procedure-oriented programming. And another of its features is the large number of libraries, we think it will make our job easier in this regard, for this reason we found Python more ideal and suitable in our choice of language.

**3. PROJECT SCHEDULE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Oğuzhan** | **Beyza** | **Yusuf** | **Özge** | **Toygun** |
| **0-2 weeks** | **Investigation of AI education** | **Firebase Research** | **Flutter Research** | **React and react tools research** | **React and react tools research** |
| **2-4 weeks** | **Firebase Research** | **Investigation of AI education** | **Integration of API with Flutter**  **Research** | **Home Page**  **page development** | **Adoption page development** |
| **4-6 weeks** | **Integration of adoption and writing the API** | **Integration of user and writing the API** | **Adoption Page**  **UI development** | **Home Page**  **API development** | **Adoption API development** |
| **6-8 weeks** | **Investigation of AI education** | **Investigation of AI education** | **Lost Page UI Development** | **Home Page**  **page development** | **Admin Page**  **And API development** |
| **8-10 weeks** | **Integration of Lost List and writing the API** | **Integration of Cities and writing the API** | **Profile Page UI Development** | **Lost Page development** | **Profile page development** |
| **10-12 weeks** | **Extraction of AI model and writing the API** | **Extraction of AI model and writing the API** | **Add Ad Page UI Development** | **Lost Page and API development** | **Profile page and API development** |
| **12-14 weeks** | **Integration of Pet and writing the API** | **Integration of Admin and writing the API** | **Integration of API’s and UI’s** | **Validation of Admin and Home pages** | **Validation of adoption and profile pages** |
| **14-16 weeks** | **TEST** | **TEST** | **TEST** | **TEST** | **TEST** |