
5CLCsCan

Gogle
Software Architecture Document

Version 1.1

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

Revision History

Date	Version	Description	Author
26/07/2024	1.0	Infomation and diagram fill in	Tuan Kiệt, Khuong Duy
28/07/2024	1.1	Review & Identify	Duy Anh

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

Table of Contents

1. Introduction	4
1.1 Purpose	4
1.2 Scope	4
1.3 References	4
2. Architectural Goals and Constraints	4
Architectural Constraints	4
Additional Constraints	4
3. Use-Case Model	6
4. Logical View	7
4.1 Architecture diagram	7
4.2 Component: Model	7
4.3 Component: View	13
4.4 Component: Controller	16
5. Deployment	20
6. Implementation View	20

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

Software Architecture Document

1. Introduction

The introduction of the Software Architecture Document provides an overview of the entire document, ensuring that readers understand the purpose, scope, and context of the architecture being described. This section sets the stage for the detailed architecture descriptions that follow.

1.1 Purpose

The purpose of this Software Architecture Document is to outline the architectural design and structure of the Gogle Platform, a website that leverages AI technology to find, suggest, and create local trips. This document serves as a guide for stakeholders, developers, and other team members to understand the system's high-level design, components, and interactions. It ensures that the architecture aligns with the overall business objectives, requirements, and constraints of the Gogle Platform.

1.2 Scope

This document covers the architectural aspects of the Gogle Platform, including its key components, modules, and their interactions. It provides an abstract view of the system, highlighting the main architectural decisions, patterns, and frameworks used. The scope includes:

- High-level architecture overview of the Gogle Platform
- Detailed component descriptions and their responsibilities
- Inter-component relationships and data flow within the platform
- Technology stack and frameworks utilized
- Deployment and operational considerations specific to the Gogle Platform

1.3 References

A comprehensive list of all documents referenced within the Software Architecture document is presented in this subsection. Each reference is meticulously cataloged, including its title, report number (if applicable), date of publication, and the publishing organization. Additionally, sources for obtaining the referenced documents are provided to facilitate accessibility and further exploration, if necessary.

2. Architectural Goals and Constraints

Architectural Constraints

- **Running Environment:**
 - **Browser:** Google Chrome, version 116.0.5845.14 or newer.
 - **System:** Windows 10–11 with no more than 1GB of RAM in use.
- **Performance:** The website must serve at least 50 users at any given time, with services responding in no more than 10 seconds.
- **Programming Languages:** HTML, CSS, JavaScript.
- **Data:** The system uses MongoDB to securely store user account and product information.
- **User Interface Design:** Designs must be simple and clear to ensure a comfortable user experience.

Additional Constraints

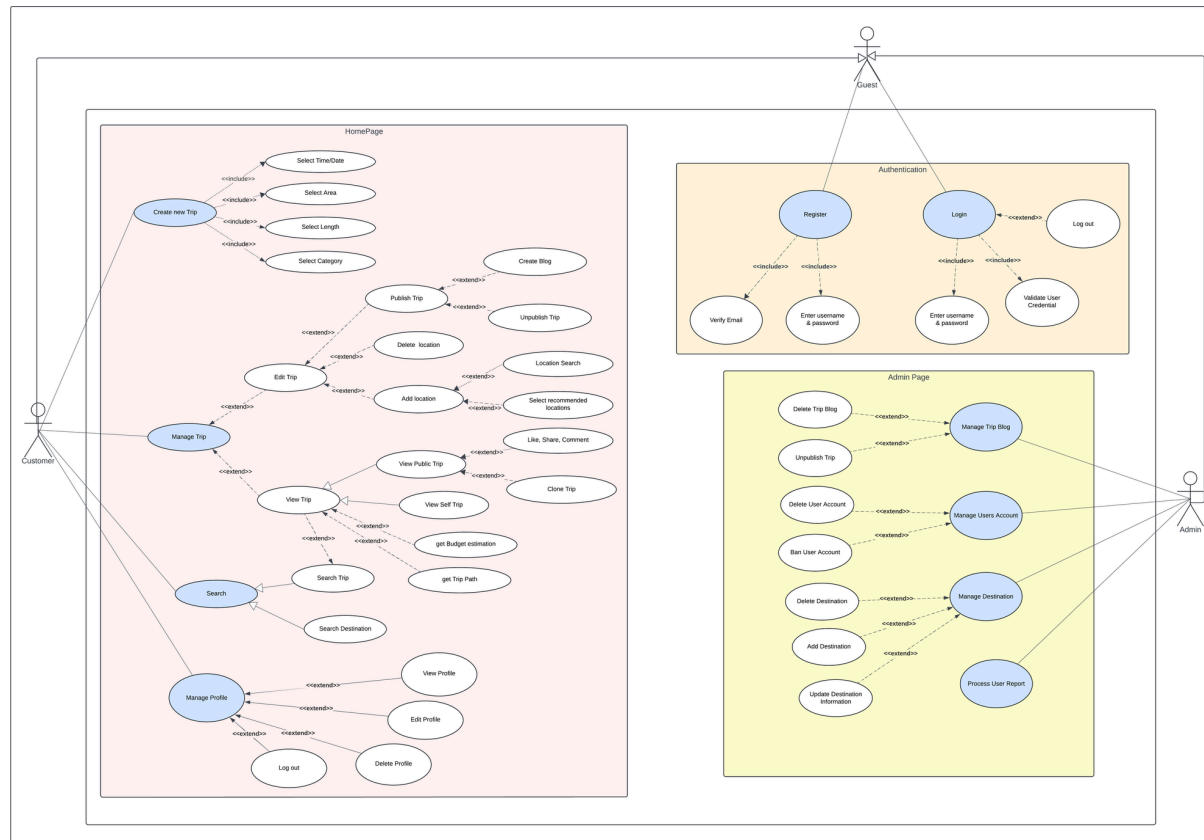
- **Security:** Must protect user data through encryption and secure authentication methods.

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

- **Maintainability:** The design must facilitate easy updates, bug fixes, and enhancements.
- **Compatibility:** The system must be compatible with major web browsers (Chrome, Firefox, Safari, Edge) and support responsive design for various devices (desktops, tablets, smartphones).
- **Portability:** The system should be easily transferable and deployable across different environments and cloud platforms without significant modifications.
- **Compliance:** The system must comply with relevant industry standards and regulations, such as GDPR for data protection and PCI-DSS for payment processing.
- **Backup and Recovery:** The system must implement automated daily backups with a recovery point objective (RPO) of 24 hours and a recovery time objective (RTO) of 2 hours in case of data loss.
- **Logging and Monitoring:** The system must have comprehensive logging and monitoring capabilities to track user activities, system performance, and errors, providing alerts for unusual or critical conditions.
- **Localization and Internationalization:** The system should support multiple languages and regional settings, including date formats, currency symbols, and time zones, allowing users to interact in their preferred language and format.
- **Energy Efficiency:** The system should be optimized for energy efficiency, minimizing server and resource usage to reduce environmental impact.
- **Accessibility:** The system must adhere to accessibility standards (e.g., WCAG 2.1) to ensure usability by individuals with disabilities, including support for screen readers and keyboard navigation.

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

3. Use-Case Model

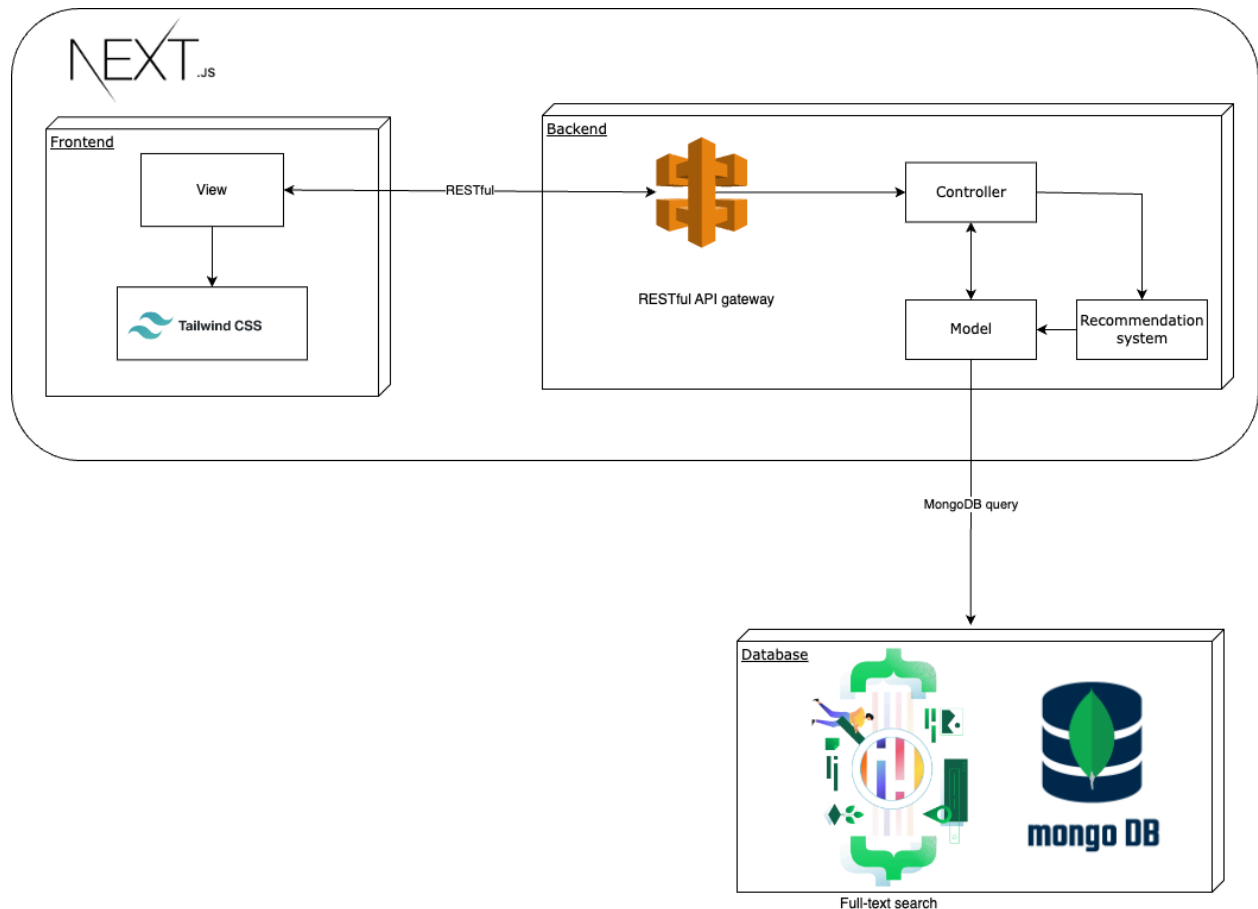


The Gogle platform is designed to streamline local trip planning through AI-powered recommendations, offering a range of functionalities to different user roles. Customers can create, manage, and search for trips, utilizing features such as selecting time, area, length, and category, as well as publishing and editing trips. Guests can register and log in to access these features, while Admins have robust control over managing trip content, user accounts, and content web pages.

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

4. Logical View

4.1 Architecture diagram



4.2 Component: Model

Description:

The Model component is responsible for managing and storing data within an application. It acts as a bridge between the user interface (UI) and the database, handling data retrieval, creation, modification, and deletion.

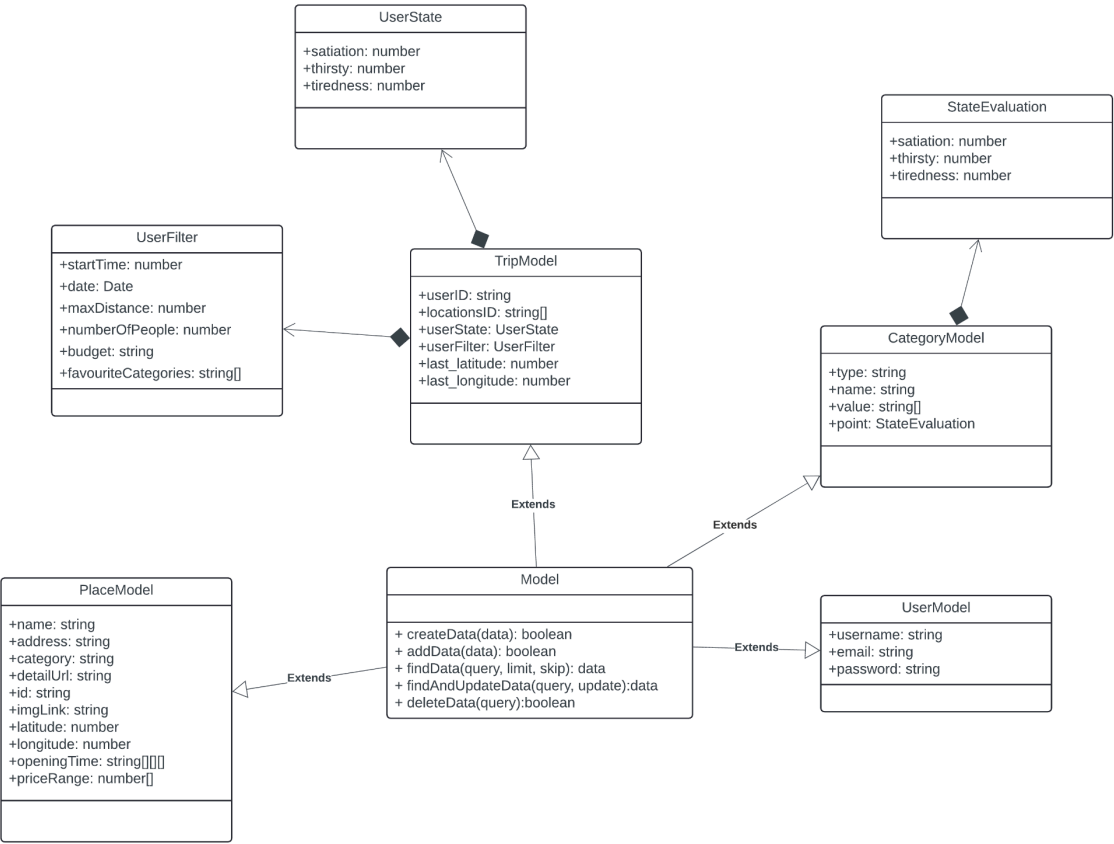
- UserState: Captures user's current state, including satiation, thirstiness, and tiredness levels.
- UserFilter: Defines user preferences for a trip, including start time, date, maximum distance, number of people, budget, and preferred categories.
- TripModel: Represents a trip, containing user ID, locations visited, user state, user filter, and potentially other details.
- CategoryModel: Defines categories for locations, with name, value, and an associated StateEvaluation.
- PlaceModel: Represents a place, including name, address, category, details URL, ID, image link,

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

latitude, longitude, opening time, and price range.

- UserModel: Stores user information, such as username, email, and password.
- Model: An abstract class providing common CRUD (Create, Read, Update, Delete) operations for data management.
- StateEvaluation: Likely used to assess how well a location matches a user's current state.

Class Diagram:



Key-class explanation:

- **Model Class**
 - Purpose
 - The Model class serves as a generic base class for data management operations, providing common functionalities for creating, adding, finding, updating, and deleting data. It acts as an abstraction layer, separating the core data operations from specific data models.
 - Methods
 - createData(data): boolean
 - Creates a new data record based on the provided data.

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

- Returns a boolean indicating success or failure.
 - addData(data): boolean
 - Adds an existing data record to a collection or database.
 - Returns a boolean indicating success or failure.
 - findData(query, limit, skip): data
 - Retrieves data based on the specified query, limit, and skip parameters.
 - Returns the found data.
 - The **query** parameter defines the search criteria.
 - The **limit** parameter specifies the maximum number of results to return.
 - The **skip** parameter determines the offset for pagination.
 - findAndUpdateData(query, update): data
 - Finds data based on the query and updates matching records with the provided update data.
 - Returns the updated data.
 - deleteData(query): boolean
 - Deletes data matching the specified query.
 - Returns a boolean indicating success or failure.
- **UserState Class**
 - **Purpose:**
 - The UserState class is designed to encapsulate a user's current physical and psychological condition. It serves as a data structure to store and manage attributes related to a user's state, such as hunger, thirst, and tiredness levels. This information can be used by other components of the system to personalize recommendations, experiences, or interactions based on the user's current needs.
 - **Attribute:**
 - Satiation:
 - Represents the user's level of fullness or satisfaction.
 - Typically a numerical value (e.g., integer or float).
 - Range might be defined (e.g., 0-100) to indicate a scale from extremely hungry to completely full.
 - Thirsty:
 - Represents the user's level of thirst.
 - Similar to satiation, it's typically a numerical value.
 - Define a suitable range (e.g., 0-100) to indicate thirst level.
 - Tiredness:
 - Represents the user's level of fatigue or exhaustion.
 - Again, a numerical value is common.
 - Define a suitable range (e.g., 0-100) to indicate energy level.
- **UserFilter class:**
 - **Purpose**
 - The **UserFilter** class encapsulates a set of criteria used to filter and refine search results or recommendations based on user preferences. It defines parameters such as desired start time, date, maximum distance, number of people, budget, and preferred categories.
 - **Attributes**
 - **startTime:**
 - Represents the preferred start time for an activity or trip.
 - Typically a numerical value representing time in a specific format (e.g.,

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

milliseconds since epoch, hours in a day).

- **date:**
 - Represents the desired date for an activity or trip.
 - Typically a Date object or a string representing a date in a specific format.
- **maxDistance:**
 - Represents the maximum distance the user is willing to travel.
 - A numerical value representing distance (e.g., meters, kilometers, miles).
- **numberOfPeople:**
 - Represents the number of people involved in the activity or trip.
 - A numerical value (integer).
- **budget:**
 - Represents the user's budget for the activity or trip.
 - A string value to allow for flexible budget representations (e.g., "low", "medium", "high").
- **favoriteCategory:**
 - Represents a list of categories the user prefers.
 - A string array to store category names..
- **TripModel Class:**
 - **Purpose**
 - The **TripModel** class represents a user's trip or journey, encapsulating essential information about the trip, including the user who initiated it, the locations visited, the user's state during the trip, the filters applied to plan the trip, and the user's last known location.
 - **Attributes**
 - **userID:**
 - A unique identifier for the user associated with the trip.
 - Typically a string or a user object reference.
 - **locationsID:**
 - An array of identifiers for the locations visited or planned to be visited during the trip.
 - The data type depends on the specific location identifier used (e.g., string, integer, object reference).
 - **userState:**
 - A reference to a UserState object, capturing the user's physical and psychological condition at a specific point in the trip.
 - This allows for tracking changes in user state during the journey.
 - **userFilter:**
 - A reference to a UserFilter object, representing the criteria used to plan the trip.
 - This provides insights into the user's preferences and constraints.
 - **last_latitude:**
 - The latitude of the user's last known location.
 - A numerical value representing the latitude coordinate.
 - **last_longitude:**
 - The longitude of the user's last known location.
 - A numerical value representing the longitude coordinate.
- **PlaceModel Class:**
 - **Purpose**

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

- The **PlaceModel** class represents a physical location, providing essential information about the place, including its name, address, category, details, unique identifier, image, geographic coordinates, opening hours, and price range.
- **Attributes**
 - **name:**
 - The name of the place.
 - Typically a string.
 - **address:**
 - The physical address of the place.
 - Typically a string.
 - **category:**
 - The category or type of the place (e.g., restaurant, hotel, attraction).
 - Typically a string.
 - **detailUrl:**
 - A URL linking to more detailed information about the place.
 - Typically a string.
 - **id:**
 - A unique identifier for the place.
 - Typically a string or a numerical value.
 - **imgLink:**
 - A URL to an image of the place.
 - Typically a string.
 - **latitude:**
 - The latitude coordinate of the place.
 - A numerical value.
 - **longitude:**
 - The longitude coordinate of the place.
 - A numerical value.
 - **openingTime:**
 - A complex data structure representing the place's opening hours.
 - A three-dimensional array where:
 - The first dimension represents days of the week (e.g., 0 for Monday, 1 for Tuesday).
 - The second dimension represents different opening time periods (e.g., morning, afternoon, evening).
 - The third dimension represents the start and end times for each period (e.g., ["10:00", "12:00"]).
 - **priceRange:**
 - An array of numerical values representing the price range of the place.
 - The array can contain two or more values to indicate the minimum and maximum prices.
- **CategoryModel Class**
- **Purpose**
 - The **CategoryModel** class defines a category or classification for places, providing information about the category type, name, associated values, and a potential evaluation point related to user state.
- **Attributes**

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

- **type:**
 - The type or category of the category itself.
 - Typically a string.
- **name:**
 - The name of the category.
 - Typically a string.
- **value:**
 - An array of string values representing subcategories or specific options within the category.
 - Can be used for more filtering or categorization.
- **point:**
 - A reference to a StateEvaluation object, potentially indicating how well places in this category match a user's current state.
 - This could be used for personalized recommendations or filtering.
- **UserModel Class**
- **Purpose**
 - The **UserModel** class represents a user of the system, storing essential information for user identification and authentication.
- **Attributes**
 - **username:**
 - A unique identifier for the user.
 - Typically a string.
 - **email:**
 - The user's email address.
 - Typically a string.
 - **password:**
 - The user's password.
 - **Important:** For security reasons, passwords are never stored in plain text. They are hashed using a cryptographic algorithm.
- **StateEvaluation Class**
- **Purpose**
 - The **StateEvaluation** class represents the changes that affect the UserState if a place's category is chosen.
- **Attributes**
 - **satiation:**
 - **Type:** int
 - **Description:** Represents the level of hunger satisfaction provided by the place's category. A positive value indicates an increase in satiation (decrease in hunger), while a negative value might indicate a decrease in satiation.
 - **thirsty:**
 - **Type:** int
 - **Description:** Represents the level of thirst satisfaction provided by the place's category. A positive value indicates an increase in thirst satisfaction (decrease in thirst), while a negative value indicates an increase in thirst.
 - **tiredness:**
 - **Type:** int
 - **Description:** Represents the impact on tiredness after visiting the place. A

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

positive value indicates an increase in tiredness (making the user more tired), while a negative value indicates a reduction in tiredness (making the user feel more rested).

4.3 Component: View

The view component displays related information in a structured and logical way so that the user can understand and interact with the whole system. It acts as a connector between the end user and the backend systems and services. In our application, this component also supports serializing data to match with other systems, providing a better user experience.

View Included:

1. Home Page

This view serves as a landing page for our application. It displays information about the project, including its aims and objectives. There are also other sections such as team introduction, project images, in-app examples, and testimonials. This page acts as a starting point for new users when they find out our project and have some questions about its functionality.

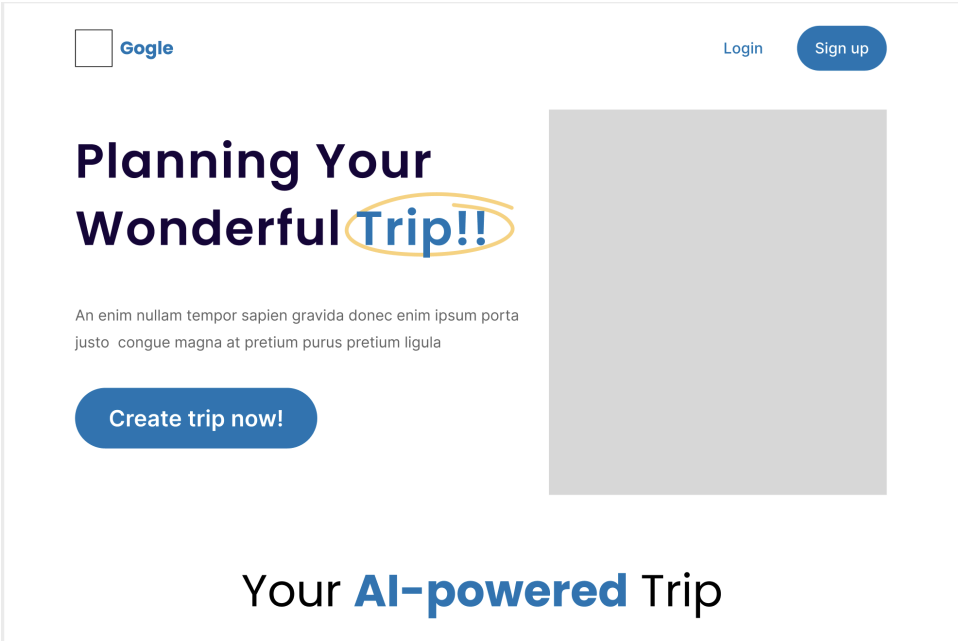


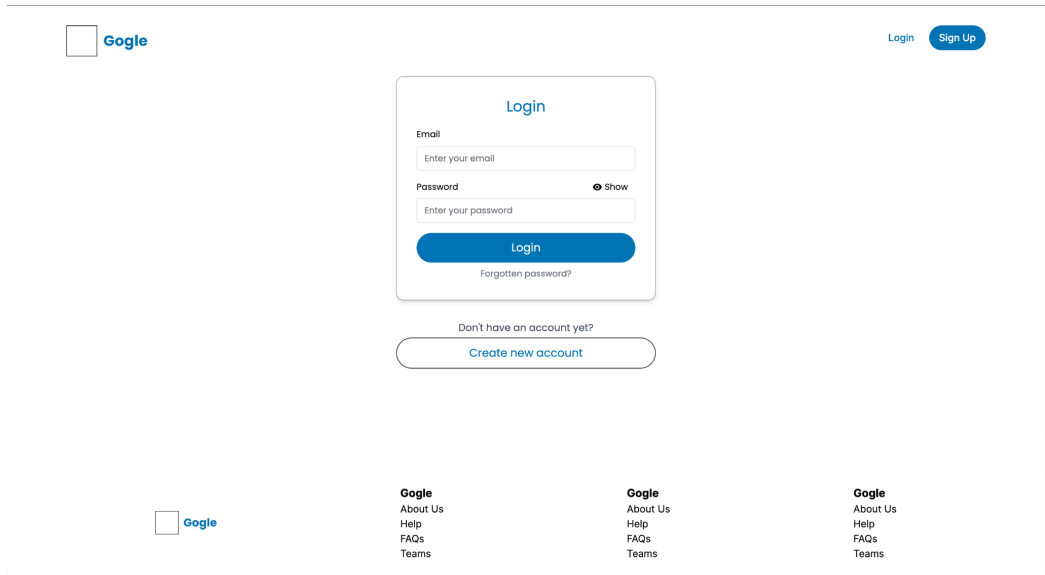
Image 1. Homepage screenshot

2. Login Page

This view displays a curated form containing different fields and labels to direct users for the login process. It handles input validation: minimum characters for password, valid email,..., and allows users to view their password (only if needed). The page supports automatic redirection after successful login, and displays proper messages on errors.

This page also contains some handy links in case the users don't have an account yet, or they have forgotten their password. These links enhance the overall user experience, and create a complete flow for the authentication process.

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	



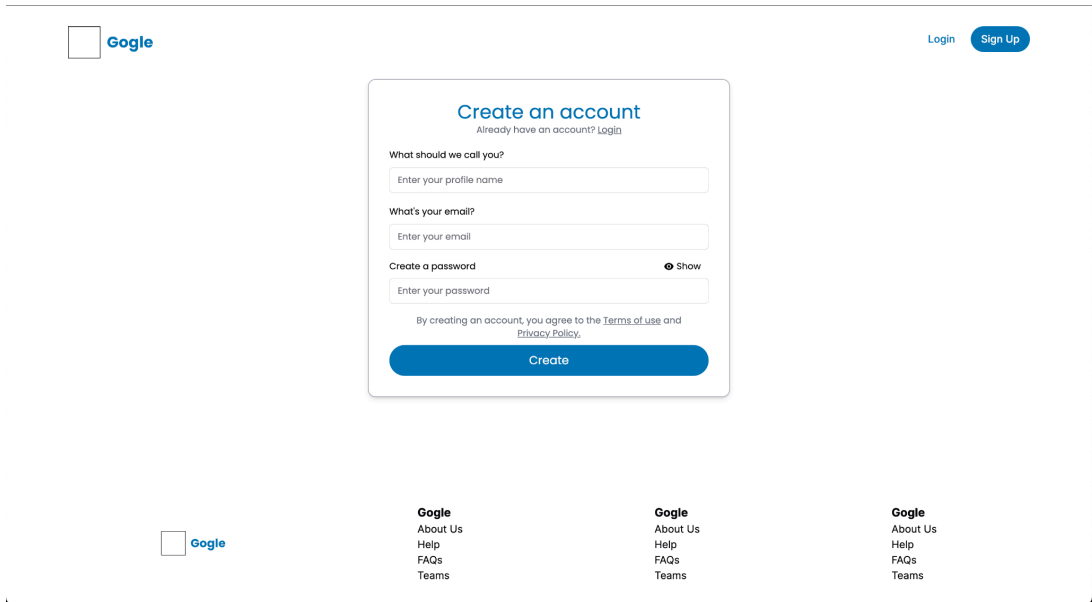
The screenshot shows a web page for signing in. At the top left is a logo consisting of a square icon and the word "Gogle". At the top right are links for "Login" and "Sign Up". The main content area features a "Login" form with fields for "Email" (labeled "Enter your email") and "Password" (labeled "Enter your password" with a "Show" toggle). Below the password field is a "Login" button and a link for "Forgotten password?". Below the form is a link for "Don't have an account yet?" and a "Create new account" button. The footer contains three columns, each with a "Gogle" logo and links for "About Us", "Help", "FAQs", and "Teams".

Image 2. Sign In Page

3. Signup Page

This view displays a form that contains different fields and labels to direct users for the signup process. It handles input validation: minimum characters for password, valid email,..., and allows users to view their password (only if needed). The page supports automatic redirection after successful signup, and displays proper messages on errors.

This page also contains some handy links in case the users already have an account. These links enhance the overall user experience, and create a complete flow for the authentication process.



The screenshot shows a web page for creating an account. At the top left is a logo consisting of a square icon and the word "Gogle". At the top right are links for "Login" and "Sign Up". The main content area features a "Create an account" form with the subtext "Already have an account? Login". The form has fields for "What should we call you?" (labeled "Enter your profile name"), "What's your email?" (labeled "Enter your email"), and "Create a password" (labeled "Enter your password" with a "Show" toggle). Below the password field is a "Create" button and a link for "By creating an account, you agree to the Terms of use and Privacy Policy.". The footer contains three columns, each with a "Gogle" logo and links for "About Us", "Help", "FAQs", and "Teams".

Image 3. Sign Up Page

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

4. Dashboard Page

This page allows authenticated users to view their created trips, and some ongoing plans in a logical way. There're also some options on each trip information, helping users to perform actions on their existing trips. Moreover, there's also a link to the user settings page, which allows users to change their credentials, as well as some public information.

5. Create Trip Page

This page contains step-by-step forms, each step allows users to fill in the trip information, as well as filter out unwanted suggestions. There's also a map to give users an overview of their upcoming trip. It also supports automatically saving progress, even without Internet connection. This feature significantly increases the user experience, allowing them to come back and continue their trip's plan at any time.

6. Trip Detail Page

This page allows users to have a summary on the planned trip. In this view, users can quickly identify places, and detailed routes from their starting point to the first destination, and so on. Users can also edit selected places, and remove places to fit their plan better.

Note: there're more pages to be developed, and has been included in the below diagram.

Class Diagram:

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

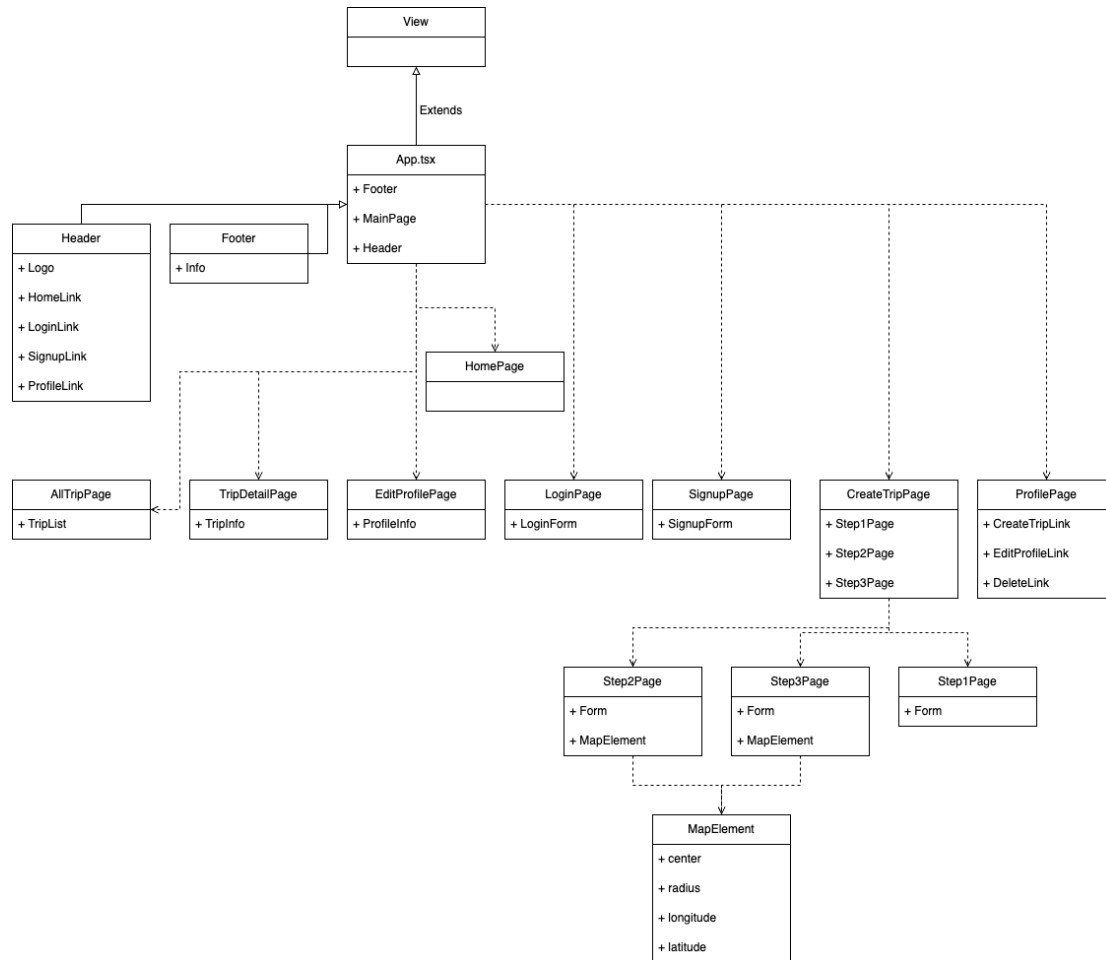


Image 4. Overview of View component

4.4 Component: Controller

Description:

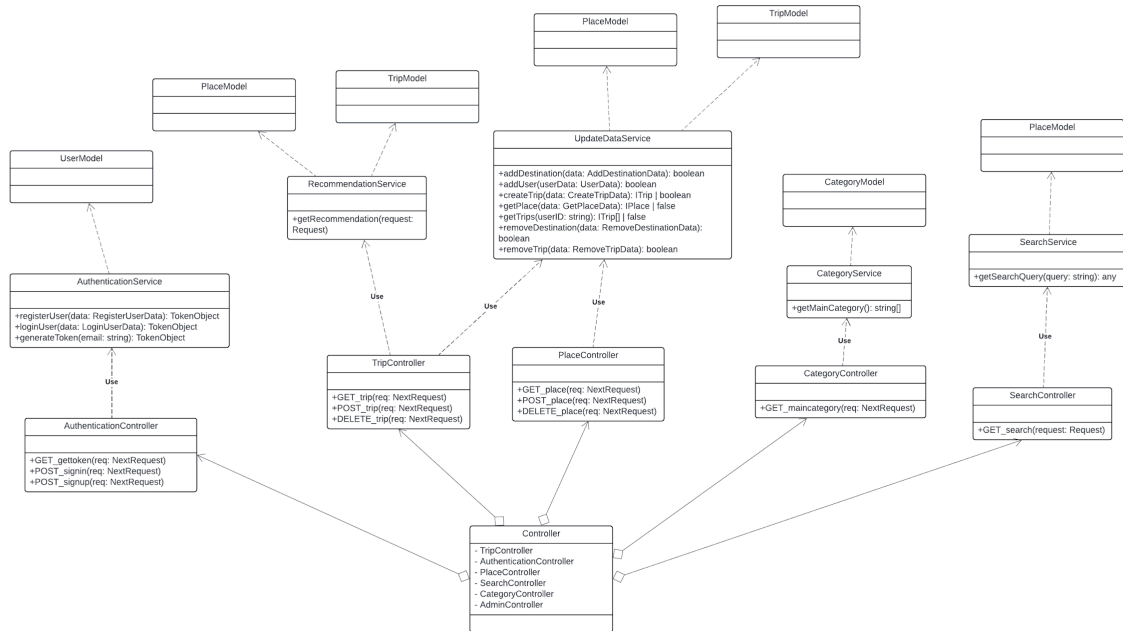
The controller component manages the interactions between the client requests and the services that handle the business logic. It ensures that the appropriate services are called and that the correct data is returned to the client. This component is composed of multiple controllers, each responsible for handling specific types of requests.

Controllers Included:

1. AuthenticationController
2. TripController
3. PlaceController
4. CategoryController
5. SearchController

Class Diagram:

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	



Key-class Explanation:

1. AuthenticationController:

Handles all authentication-related requests, such as user login, signup, and token generation.

- Methods:
 - **+GET_gettoken(req: NextRequest)**: Generates and returns a new authentication token.
 - **+POST_signin(req: NextRequest)**: Authenticates a user based on provided credentials.
 - **+POST_signup(req: NextRequest)**: Registers a new user in the system.

2. TripController:

Manages trip-related requests, such as creating, retrieving, and deleting trips.

- Methods:
 - **+GET_trip(req: NextRequest)**: Retrieves details of a specific trip.
 - **+POST_trip(req: NextRequest)**: Creates a new trip based on provided data.
 - **+DELETE_trip(req: NextRequest)**: Deletes an existing trip.

3. PlaceController:

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

Handles requests related to places, including retrieving, adding, and removing places.

- Methods:
 - `+GET_place(req: NextRequest)`: Retrieves details of a specific place.
 - `+POST_place(req: NextRequest)`: Adds a new place to the system.
 - `+DELETE_place(req: NextRequest)`: Removes an existing place from the system.

4. CategoryController:

Manages category-related requests, primarily retrieving the main categories.

- Methods:
 - `+GET_maincategory(req: NextRequest)`: Retrieves the main categories available.

5. SearchController:

Handles search queries within the system.

- Methods:
 - `+GET_search(request: Request)`: Processes search queries and returns relevant results.

Integration with Services:

In this section, we detail the specific roles and functionalities of each service that the controllers interact with. Each service encapsulates a distinct set of business logic and operations, ensuring a clean separation of concerns and promoting maintainability.

AuthenticationService

Role: Manages user authentication, registration, and token generation.

- `registerUser(data: RegisterUserData): TokenObject`
 - Registers a new user and returns a token object.
- `loginUser(data: LoginUserData): TokenObject`
 - Authenticates a user with the provided credentials and returns a token object.
- `generateToken(email: string): TokenObject`
 - Generates a new authentication token for the given email.

Interacts With:

- `AuthenticationController`: For handling requests related to user sign-in, sign-up, and token management.

UpdateDataService

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

Role: Manages updates to trip and place data, including adding, removing, and retrieving destinations and trips.

- addDestination(data: AddDestinationData): boolean
 - Adds a new destination to a user's trip.
- addUser(userData: UserData): boolean
 - Adds a new user to the system.
- createTrip(data: CreateTripData): ITrip | boolean
 - Creates a new trip with the provided data and returns the trip object or a boolean status.
- getPlace(data: GetPlaceData): IPlace | false
 - Retrieves place information based on the provided data.
- getTrips(userID: string): ITrip[] | false
 - Retrieves all trips associated with the specified user ID.
- removeDestination(data: RemoveDestinationData): boolean
 - Removes a destination from a user's trip.
- removeTrip(data: RemoveTripData): boolean
 - Deletes an existing trip based on the provided data.

Interacts With:

- TripController: For creating, retrieving, and deleting trips.
- PlaceController: For retrieving, adding, and removing places.

CategoryService

Role: Manages operations related to categories, including retrieving the main categories.

- getMainCategory(): string[]
 - Retrieves the main categories available in the system.

Interacts With:

- CategoryController: For handling requests to retrieve main categories.

SearchService

Role: Handles search queries within the system, ensuring that relevant data is retrieved based on the search parameters.

- getSearchQuery(query: string): any
 - Processes the provided search query and returns the relevant results.

Interacts With:

- SearchController: For handling search requests and returning the results to the client.

Gogle	Version: 1.1
Software Architecture Document	Date: 28/07/2024
SAD	

5. Deployment

[Leave this section blank for PA4.]

In this section, describe how the system is deployed by mapping the components in Section 4 to machines running them. For example, your mobile app is running on a mobile device (Android, iOS, etc), your server runs all components on the server side including the database]

6. Implementation View

[Leave this section blank for PA4.]

In this section, provide folder structures for your code for all components described in Section 4.]