

# Background

## Problem:

As part of Singapore's smart nation initiative, we are trying to promote better living through our app TripGenie, an app which helps you to plan your overseas trip itinerary. In the post-pandemic world, everyone is getting back to travelling but many of us lack the time or the will to properly plan an efficient itinerary for the trip to maximise the time spent there to explore as much of our destination as possible. Planning a holiday is a hassle as there are many factors to take into account, the best places to visit based on rating, the travelling time between places coupled with when these places are even open throughout the day. TripGenie takes all these into account and gives the user an optimised route, minimising the hassle and amplifying all the fun.

## Proposed Solution:

We have come up with an app that will do all that for you. TripGenie, takes into account the place you are travelling to, the number of days you are going to be spending there and the places you want to visit to deliver an optimised schedule for your trip. This schedule allows you to spend your holiday efficiently by clustering the places you are visiting based on their distances from each other to separate days so you spend less time travelling each day and more time doing the activity or enjoying the views. It also perfectly plans the day so that whatever is better to be done in the morning as it's opening hours are earlier will be pushed to the front of the day so that you won't miss out on anything due to it being closed. This app removes the worries of planning a trip and instead makes you look forward to your trip with no hesitations on your itinerary.

## Description of resources used in our project:

1. Google Maps API
2. Google Places API
3. Google Login Authentication
4. Firebase realtime Database

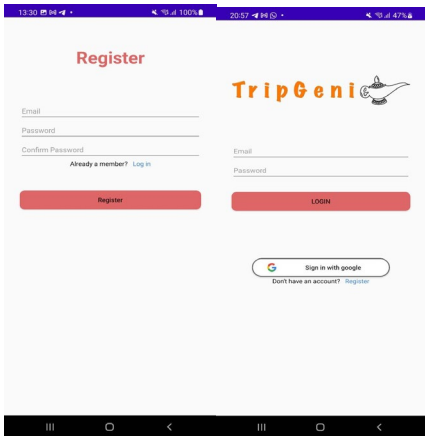
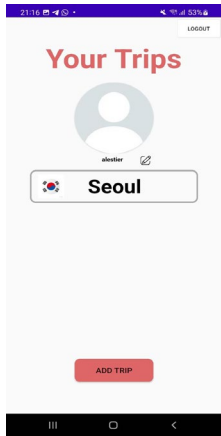
# System Design and Implementation:

## System Architecture:

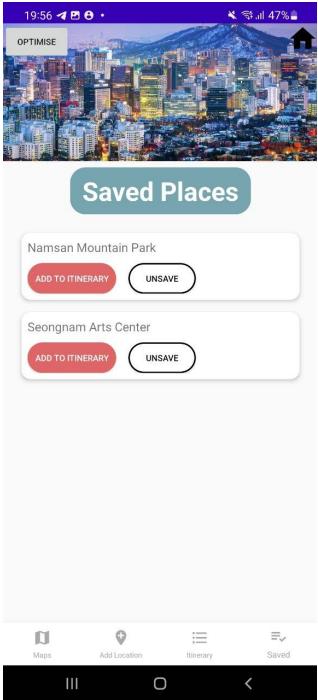
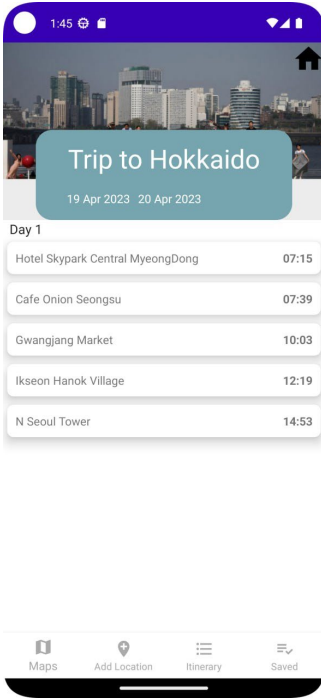
The system architecture of TripGenie can be broken down into the following components:

1. User Interface (UI): Provides an intuitive interface for users to interact with the app.
2. User Authentication: Handles user login and registration using Google Login Authentication.
3. API Integration: Integrates Google Maps API and Google Places API for location data and place information.
4. Back-end Server: Get Request using Places APIs, google authentication with firebase to manage user data, storage of user data including trips created and saved places, and processes itinerary optimization requests via k means clustering and greedy algorithm.
5. Database: Stores user information, preferences, and generated itineraries.

## App Screenshots

Register/Login	Homepage
	
<p>Users can login using 2 methods:</p> <ol style="list-style-type: none"><li>1) Email and password login</li><li>2) Google login</li></ol> <p>The users google username will be stored in firebase database which will allow them to change their username in the homepage</p>	<p>Users can view their past trips or add new trips by clicking on the add trip button</p>

<p>Add new trip</p>	<p>Add new location</p>
	
<p>Users will input country and duration of travel and click on the start planning button</p>	<p>Users can either search for a location or get suggestions to add a location to the saved places.</p>
<p>Maps</p>	<p>Recommendations</p>
	
<p>Users can use the map to search for locations</p> <p>Using Google Maps API to display map</p>	<p>From the get recommendations button, the places of interest shows the attraction and restaurant near their search location</p> <p>We integrated Google Places API to search for nearby places, such as restaurant or shops</p>

Saved Places	Itinerary
	
<p>Users can click on the optimise button to get their optimised schedule based on their saved places.</p>	<p>The trips json file are stored in sharedpreferences and is displayed in the itinerary page</p>

## Design Patterns:

1. Model-View-Controller (MVC): This design pattern separates the application logic into three interconnected components: Model, View, and Controller. The Model (Place, PlaceAdapter, SavedPlacesAdapter, and RecommendationAdapter classes) represents the data and business logic. The View (layout XML files) is responsible for displaying the data and receiving user input. The Controller classes (NewLocations, SavedLocations, and RecommendationsActivity) manage the flow of data between the View and the Model and handle user input. This pattern helps to organise the code, making it easier to maintain and extend.
2. Adapter Pattern: This pattern is used to bridge the gap between two incompatible interfaces. In the NewLocations, SavedLocations, and RecommendationsActivity classes, PlaceAdapter, SavedPlacesAdapter, and RecommendationAdapter act as adapters that convert the data (e.g., lists of Place objects) into a format that can be displayed by UI components such as RecyclerView and AutoCompleteTextView. This pattern ensures seamless communication between the data and UI components.

3. **Observer Pattern:** This pattern defines a one-to-many dependency between objects, where a change in one object (the subject) automatically triggers a change in the dependent objects (the observers). In the NewLocations and SavedLocations classes, listener interfaces such as PlaceAdapter.OnAddPlaceClickListener, OnClickListener, OnItemSelectedListener, and TextWatcher serve as observers that are triggered when specific events occur in the UI components (subjects). In RecommendationsActivity, the RecyclerView and RecommendationAdapter implement the Observer pattern, where the adapter observes changes in the data set and updates the RecyclerView accordingly.
4. **Design Principles:** The Place and PlacesApiHelper classes follow several design principles:
  - a. **Single Responsibility Principle (SRP):** Each class has a single responsibility, focusing on one aspect of the application logic. This principle helps to create modular, maintainable code. We separate the job of storing places in a single class for attributes like opening hours, latitude, longitude, address, while the places api helper's job is to communicate with the respective google apis. Each class only has one reason to change.
  - b. **Encapsulation:** This principle restricts access to an object's internal state, allowing access only through public getter methods. Encapsulation protects the integrity of the object's state and ensures that the object is used only as intended.
  - c. **Composition over Inheritance:** Instead of inheriting from multiple classes, the Place class composes several nested classes (Geometry, Location, and OpeningHours) to represent different aspects of a place. This approach helps to keep the code modular and maintainable.
5. **Template Method Design and Abstraction:** In this design pattern, there is an algorithm with a fixed structure, but the implementation of some steps are done by the subclasses. In order to create the itinerary in Savedlocations.java, we use an abstract class Algorithm.java which has 2 abstract methods Clustering() and GreedyAlgorithm(). These 2 methods are abstract as they are subject to change for future plans. In future we hope to try out other kinds of clustering methods as well as other kinds of optimisation algorithms. We have CustomAlgorithm.java class which extends this Algorithm class and implements the k++ means for clustering and implements the custom greedy algorithm which takes into account opening hours, time spent and eating places.

## Other Key technical Features used:

1. **Google Maps API:** Used for displaying maps and calculating routes between locations.
2. **Google Places API:** Provides information about various points of interest, including ratings, opening hours, and nearby locations.
3. **Google Login Authentication:** Allows users to sign in with their Google account for seamless and secure access to the app.
4. **Optimization Algorithm:** K++ means algorithm together with a custom algorithm that considers various factors such as time spent, opening hours and prioritising eating places during meal times to generate the most efficient itinerary

## Description of Work:

1. Felicia : User Interface, Google Login Authentication and Storing of user data in firebase
2. Dylan : User Interface, Plan Trip page and K means algorithm
3. Varun : Get Recommendations , Get Suggestions and setting up google places api helper class and data storage.
4. Zheng Wei : User Interface, storing of data in shared preferences and the Itinerary Page
5. Aleister : Navigation Bar and completion of the K means algorithm
6. Shjonahan : Setting up project, version control management, implemented Maps Activity and did the optimised schedule algorithm

## Possible Future Work:

1. Visually display the clusters on the map once the optimise schedule is pressed so users can see how each day is planned
2. Able to toggle the places in the itinerary so if user strongly prefers one time before another they can make the changes to the the itinerary and the Itinerary will adjust accordingly
3. Give short description of each places once added to saved places so user can choose whether to add or remove the place
4. Able to plan trip with friends by adding an invite button so that multiple accounts can plan one trip simultaneously
5. Different optimization filters such as choose only outdoor activities or indoor activities, travel by bus/train/walk/car,
6. Make 1 trip include different cities

## Conclusion

TripGenie is an innovative app designed to simplify the process of planning a trip, allowing users to focus on enjoying their vacation. It employs Google Maps API, Google Places API, and Google Login Authentication to provide an intuitive and seamless experience for users. The system architecture incorporates a range of design patterns, such as Model-View-Controller, Adapter Pattern and Observer Pattern, resulting in a well-organised, maintainable, and flexible code structure. With the support of a dedicated team, each member has contributed their expertise to various aspects of the project, such as User Interface, API Integration, Backend Server, and Optimization Algorithm. The app has immense potential for future enhancements, including visual cluster representation, itinerary customization, place descriptions, and collaborative trip planning with friends. TripGenie is a valuable tool that will undoubtedly revolutionise the way people plan their trips, making travel more enjoyable and stress-free.

# Citations:

Google Maps API documentation: <https://developers.google.com/maps/documentation>

Google Places API documentation: <https://developers.google.com/places/web-service/overview>

Google Login Authentication documentation: <https://developers.google.com/identity/sign-in/android/start>