Android Application – 50.001 1D Project Cheng Ruo Xi, Gabriel Wong, Lee Tze How

<u>**TourSG – Application Report**</u>

<u>**Introduction**</u>

The general theme of our application is *convenience*. The Tour Singapore application is an application designed to give users a fuss free experience in exploring Singapore. Using our application, a tourist can easily find information relevant to him through several features designed to provide the optimal user experience.

With convenience being the central theme of our project, the goal of the UI is to give users very quick access to various forms of information they will need as they assess their choices. Every page has hence been designed to enable quick access of information for the user.

<u>**Functionalities**</u>

<u>The home page</u> provides users with a quick introduction and quick access to the functionality of the application. The two main products, *Nearby* and *Itinerary*, encompass the majority of the functionality the user will be using. An *About* button provides a quick Alert Dialog to the user, used as a quick 'help' function of sorts; it also provides a quick foreword from us, the developers, to the user.

<u>A settings option</u> in the top-right corner allows users to alter preferences, linked and fetched from a **Firebase database** linked to a **preference manager** with three currently functional changeable preferences: Email, Night Mode, and daily budget. Two other preferences are currently changeable, but have yet to be implemented – the Notifications and the Language preferences. The database is fetched every time the settings page is accessed, serving as a means of both preserving data across devices independent of the phone's local memory; this database is based on the user's input email, allowing for unique assignment of settings based on each user's identity. A Night Mode option allows for users to change the colour scheme of the settings page from light to dark themes, and the user can set his daily budget for the itinerary.

<u>The Nearby page</u> is responsive, allowing users to quickly find locations relevant to their circumstances. There are several preset checkboxes that allow for users to easily highlight their preferred options, which **pull from a Google Places API** to place markers on a Google Maps Fragment (which is based on an **Implicit Intent**). The locations are highlighted different colours based on their respective tags; for example, the 'Food' checkbox identifies nearby locations under the tags 'restaurant', 'bars', 'café', 'meal_takeaway', and 'meal_delivery'.

The search bar within the Nearby page allows users to quickly find a location as desired; it is also resistant to minor spelling and syntax errors, such as 'sentozza' correctly identifying "Sentosa". Execution of the search option would direct the Map view to the highlighted location, allowing users to find desired locations on the map.

The database retrieved from Google Places is in **JSON format** using an AsyncTask, then parsed via functions in a DataParser file. The locations' respective tags, latitude/longitudes, and vicinities are then processed to show up as markers in the Google Map fragment.

The <u>Itinerary Page</u> has three separate functionalities.

A **Weather** row occupies the top of the LinearLayout, showing the user the current temperature and weather. This information is fetched from a **Weather API** (NEA's 24 hour Forecast API) in a JSON format, and displayed accordingly. The goal of this function is to provide users with an awareness of possible hindrances to their desired journey throughout the day.

The second row is a means for the user to select his/her desired journey for the day. The reason we do this is to heavily reduce the computation time required for route calculation (further elaboration below), while having little impact on their user experience, as tourists generally have *themes* between days in their vacation as well.

Having selected a *theme*, the user can then click the *select locations* button to view a curated list of relevant destinations to tour, built with a **Recycler View**. Highlighted locations are saved within a preference manager; their experience is helped by a live on both a **GettyImages API** (for an image of the most recently selected location) and a **Wikipedia API** for a quick description, allowing tourists with little to no experience in Singapore to make decisions on their preferred destinations.

After selecting his destinations, the user can then select the *Get Itinerary* button to see a list of the suggested order of traversal of his locations. The *Get Itinerary* button runs an algorithm that provides a suggested route that is close to the optimal route desired; further details regarding the algorithms involved will be discussed.

Optimization algorithms

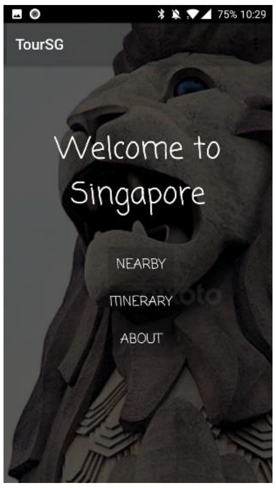
The database contains a list of all relevant locations, with Taxi, Public Transport, and Walking times and costs generated with a Google Distance Metrics API and saved within a csv format and parsed accordingly. All data is stored within code as HashMaps for quick reference (O(1) lookup time) – as the number of data points do not exceed 100 and the destinations are already segregated into several subdivisions, the space taken by a HashMap (n^2) is not computationally unfeasible.

The organization of necessary destinations planned within an itinerary is a computationally complex problem; and a global optimum can only be found with a solution that has a (n-1)!/2 runtime, with n being the number of destinations selected. To tackle this issue for the itinerary planning, we decided to take a *simulated annealing* approach to the itinerary generation.

A **brute force** solution was also designed, and quickly deemed to be unfeasible when the runtime exceeded half an hour without fruition when running on a list with 23 locations (with a runtime on the order of presumably 22!/2, multiplied by the number of different means of transport (public transport, taxi, walking). The estimated runtime of a brute force solution is of the order of n factorial; and the brute force approach was designed using recursion. The brute force algorithm, though not used, is available for reference (as BruteForceSolver.java).

The runtime of simulated annealing, on the other hand, can be configured, allowing for a far faster (but possibly less optimal) solution to be computed. This is ideal for our pathfinding application, as a *generally optimal* solution found in polynomial time is far more useful than a global optimum found in factorial time.

Simulated annealing is a probabilistic algorithm designed to find solutions that are close to the global optimum. Using changing *temperatures*, over time the iterative algorithm increases in probability that it will terminate based on a pre-set *rate of cooling*. Options that are not immediately optimal may also be explored in the iteration through the temperature, meaning that over numerous iterations “bad options” have a lower probability of being adopted; however, as this does not completely eliminate non-immediately-optimal solutions, simulated annealing avoids the downfalls of a naïve greedy algorithm, allowing for better approximations to global optima.

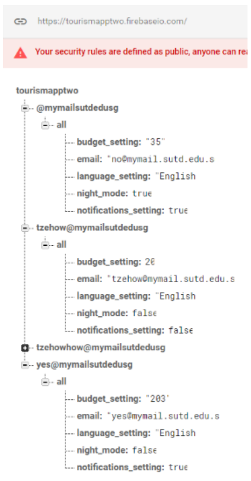

TourSG

Welcome to Singapore

NEARBY

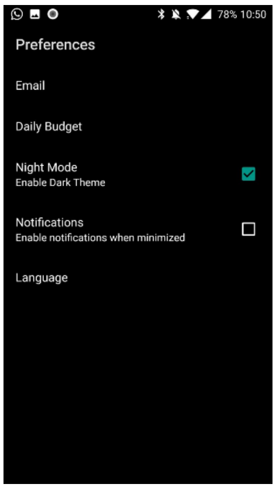
ITINERARY

ABOUT


https://tourismapptwo.firebaseio.com/

Your security rules are defined as public, anyone can re

```
tourismapptwo
⊟-- @mymailsutdedusg
    ⊟-- all
        --- budget_setting: "35"
        --- email: "no@mymail.sutd.edu.s
        --- language_setting: "English
        --- night_mode: true
        --- notifications_setting: true
⊟-- tzehow@mymailsutdedusg
    ⊟-- all
        --- budget_setting: 26
        --- email: "tzehow@mymail.sutd.edu.s
        --- language_setting: "English
        --- night_mode: false
        --- notifications_setting: false
⊞-- tzehowhow@mymailsutdedusg
⊟-- yes@mymailsutdedusg
    ⊟-- all
        --- budget_setting: "203"
        --- email: "yes@mymail.sutd.edu.s
        --- language_setting: "English
        --- night_mode: false
        --- notifications_setting: true
```


Preferences

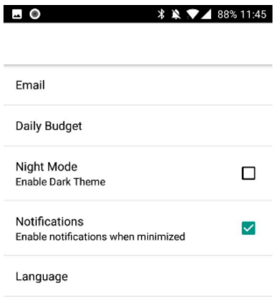
Email

Daily Budget

Night Mode ✓
Enable Dark Theme

Notifications ☐
Enable notifications when minimized

Language

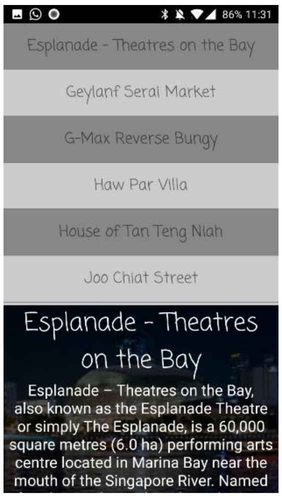

Email

Daily Budget

Night Mode ☐
Enable Dark Theme

Notifications ✓
Enable notifications when minimized

Language


Esplanade - Theatres on the Bay

Geylanf Serai Market

G-Max Reverse Bungy

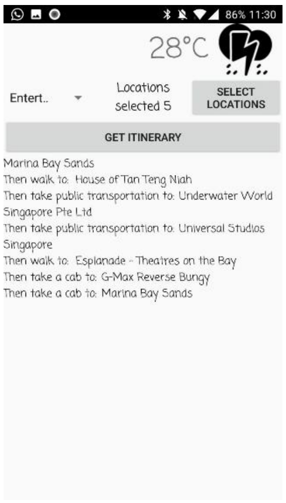
Haw Par Villa

House of Tan Teng Niah

Joo Chiat Street

Esplanade - Theatres on the Bay

Esplanade – Theatres on the Bay, also known as the Esplanade Theatre or simply The Esplanade, is a 60,000 square metres (6.0 ha) performing arts centre located in Marina Bay near the mouth of the Singapore River. Named


28°C

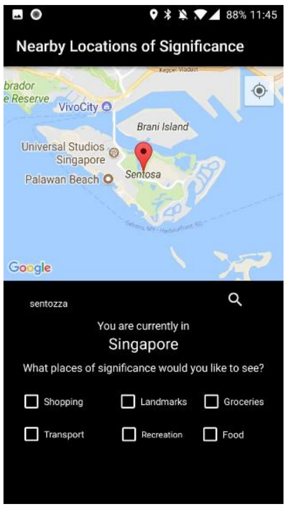
Entert..

Locations selected 5

SELECT LOCATIONS

GET ITINERARY

Marina Bay Sands
Then walk to: House of Tan Teng Niah
Then take public transportation to: Underwater World Singapore Pte Ltd
Then take public transportation to: Universal Studios Singapore
Then walk to: Esplanade - Theatres on the Bay
Then take a cab to: G-Max Reverse Bungy
Then take a cab to: Marina Bay Sands

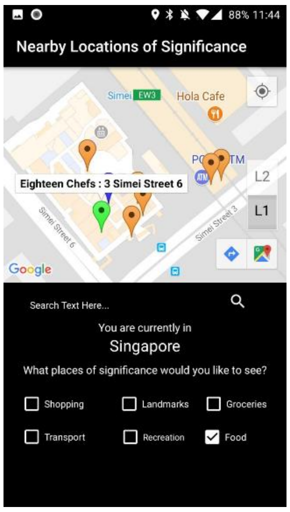

Nearby Locations of Significance

sentozza

You are currently in
Singapore

What places of significance would you like to see?

☐ Shopping ☐ Landmarks ☐ Groceries
☐ Transport ☐ Recreation ☐ Food


Nearby Locations of Significance

Search Text Here...

You are currently in
Singapore

What places of significance would you like to see?

☐ Shopping ☐ Landmarks ☐ Groceries
☐ Transport ☐ Recreation ✓ Food