MASTER OF TECHNOLOGY

(INTELLIGENT SYSTEMS)

PROJECT REPORT
Intelligent Reasoning Systems

TravelAI

Team Members
Mohamed Mikhail Kennerley
Loy Pei Xin Veronica
Oh Chun How

Executive Summary

Travel is a 2.9 trillion U.S. dollar industry. Aggregator sites scrape the web to present flight deals for a single searched date, and hotel deals for a given date range. Traditional travel agencies offer hassle-free multi-destination travel packages at a premium. However, there is no product that meets the needs of a free and easy, multi-destination traveller who would like to find an itinerary with the lowest overall flight and hotel cost. For this traveller to manually look up all possible combinations of flights and hotels to find the lowest price would be infeasible. An exhaustive search for a trip with n different destinations, and d number of days would amount to ($n! \times d$) searches.

This project presents a TravelAI, a hassle-free end-to-end solution to this problem. TravelAI is a conversational agent that helps users book travel journeys optimised for their cost and time priorities. It automates the collection of flight and hotel information, solves for the optimal combination that balances cost and time spent at each destination, and books the optimized package with the relevant travel agencies. In addition, it will analyse the user's profile and destinations and provide a curated list of top places of interest tailored to their interests.

1. Market Research

1.1 Target Market

For many individuals in the developed countries along with a growing number of individuals in the developing countries, the rising household income means that travelling is no longer an uncommon affair. The average number of trips taken by individuals, who had booked at least one trip online in the past one year, ranges from 2.8 trips a year for Canadians to 5.6 trips a year for Mexicans[1]. Travel remains an industry that is likely to see continued growth in the near future.

Of the US 2.9 trillion dollar travel industry[2], most can be attributed to travellers seeking short weekend getaway and single destination trips[3]. However, we believe that there is a small but sizable portion of travelers with very different travelling needs. For these people, travel is not just a short break from work. Instead, it is an opportunity to seek personal growth and broaden horizons, by backpacking across Asia or conquering 15 cities in a month.

Our target users are travellers who have the time for longer journeys with multiple destinations, and are looking for the best value they can get. They might be students on break, fresh graduates or working professionals between jobs, with a limited window of opportunity to go on such trips. Because multi-destination travel is characteristic of a stage in life and opportunistic in nature, we believe the demand for such travels will remain fairly strong.

1.2 Existing Travel Agencies

In booking a trip, a typical user can either use web scrapers such as Skyscanner or Hotels.com, or approach traditional travel agencies such as Chan Brothers or EU Tours. Online

[1] Expedia Group Media Solutions 2018, *Multi-national Travel Trends* [online] Available at: https://info.advertising.expedia.com/hubfs/Content_Docs/Rebrand-2018/MULTI-NATIONAL%20TRAVEL%20TRENDS-Small.pdf [Accessed 9 May 2020]
[2] S.Lock 2020, statista.com, *Tourism worldwide - Statistics & Facts*, Available at: https://www.statista.com/topics/962/global-tourism/ [Accessed 9 May 2020]
[3]T.Borden, Business Insider, Available at: https://www.businessinsider.sg/millennials-are-shortening-travel-and-leading-microcation-mini-vacation-trend-2019-12?r=US&IR=T [Accessed 9 May 2020]

scrappers aggregate flights and hotels from different places, and returns to cheapest flights and hotels for the user-specified date. They answer the question “what are the cheapest flights from Madrid to Paris on June 1st?”, or “where can I book the cheapest one night stay at The Fullerton Hotel Singapore?” For many, the Best Price Guarantee promise is very attractive and these web scrapers are the go-to resource for booking their flights and hotels[4].

Traditional travel agencies, on the other hand, attract a different crowd of travellers. They typically have fixed itineraries, and can be attractive for people who prefer multi-destination travel without the fuss of planning. These people are willing to accept a decreased flexibility in travel destinations for the convenience of a fixed itinerary.

Despite the strengths of the above travel agencies, neither of them fully meet the requirement of our target market, who is looking for free-and-easy multi-destination travel at the best price. This will be elaborated more in the next section.

1.3 Problem Description

Online scrapers return the lowest-priced flights and hotels, and this is sufficient for a user planning a single-destination trip. However, finding the lowest cost of a multi-destination trip is much more complicated. The *sequence* of travel plays an important role because hotel and flight prices vary drastically from day to day as demand varies. As the number of destinations in one trip increases, the number of possibilities increases dramatically. This is similar to the *Travelling Salesman Problem*. A user who would like to visit *n* cities has to search for up to *n!* possible combinations to find the one with the lowest cost. This is the number of flights he has to search for each day of travel, and excludes the search for flights to and from his home city. To put things into perspective, a trip with 5 destinations quickly scales up to 120 different possible flights per travel day. It is simply not practical for the user to spend the time and effort in doing an exhaustive search to get to the lowest-priced flights.

[4] Expedia Group Media Solutions 2018, *Multi-national Travel Trends* [online] Available at: https://info.advertising.expedia.com/hubfs/Content_Docs/Rebrand-2018/MULTI-NATIONAL%20TRAVEL%20TRENDS-Small.pdf [Accessed 9 May 2020]

Therefore, our project introduces TravelAI, a conversational agent that helps users book multi-stop travel journeys optimised for their cost and time priorities, and offers a curated list of places of interest tailored to user profile and his travel destination.

1.4 Project Objectives & Scope

The travel lifecycle can be broadly divided into four stages: Inspiration, Research, Booking, and Taking the Trip. TravelAI focuses on the *Booking* stage. This means that before coming to TravelAI's chatbot, users should have decided on the following:

- Travel destinations: cities
- Travel accommodations: hotel ratings
- Travel dates
- No of adult and children that will be travelling

With this information, the application will generate the combination of flights and hotels that is the cheapest, yet balances the number of travelling days in each destination. Essentially, TravelAI answers the question: "I know where I want to fly, where I want to stay, and the dates I will be travelling on. What is the best package for my flights and hotels?" They will then be directed to complete their booking should they opt to do so.

Additionally, users can also choose to use the trip's Points of Interest (POI) recommender function by providing their information on their age, sex, and whether they are travelling alone, with friends, or family. TravelAI will, based on an in-house knowledge base of travel preference, return a curated list of places of interest tailored to them.

2. System Architecture

The diagram below shows an overview of TravelAI's system architecture. Users interact and provide inputs through the conversational agent that is powered by Google's DialogFlow. This conversational agent is embedded in a web page which is rendered by the Web Server.

The agent's cognitive system will prompt and guide users for the information that the backend needs, and passes this information to the webhook service which is also hosted by the Web Server. As part of the fulfillment, the Web Server will make calls to a Flight API and Hotel API. Based on the user-specified destinations and dates, the Flight API returns the cheapest flights to and from each destination for each date. Similarly with the hotels and dates, the Hotel API returns the cost of a night stay in the hotel for each date.

The information on flight and hotel received from the APIs are then passed to the Optimizer, who will filter for the flight-date and hotel-date pairs that minimizes the total travel cost. This filtered data is then displayed to the user via the chatbot.

Depending on whether provides details on their age, sex and relationship to the companians they will be travelling with, if any, call to the API which provides a list of top Places of Interest on the Trip Advisors's website for the user-specified destinations will be triggered. This data, together with the users' profile, will be fed into our rule-based system. The output will be a curated list of places of interest tailored to the user's profile.

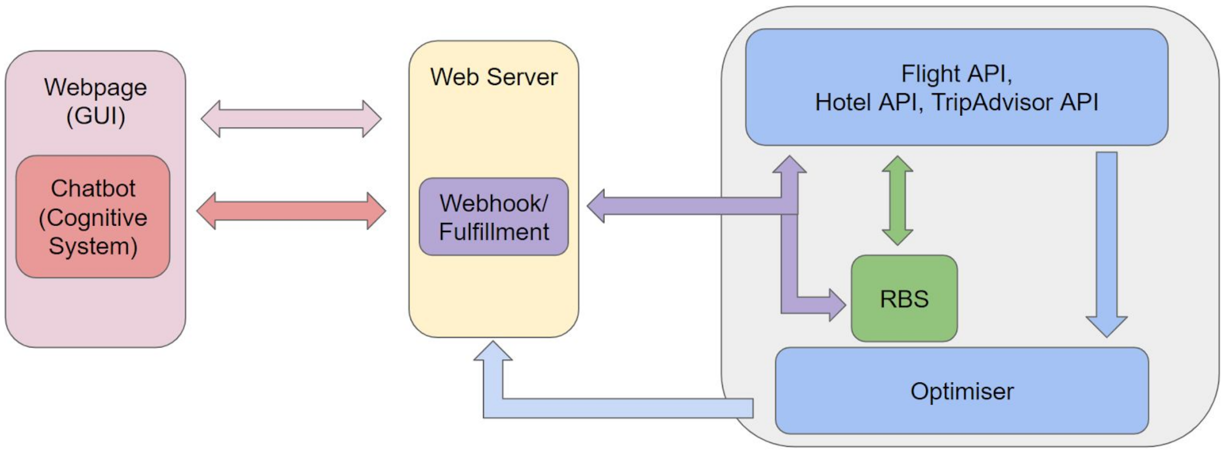

Webpage (GUI)
Chatbot (Cognitive System)
Web Server
Webhook/ Fulfillment
Flight API, Hotel API, TripAdvisor API
RBS
Optimiser

Figure1: System Architecture

3 System Design and Implementation in tools

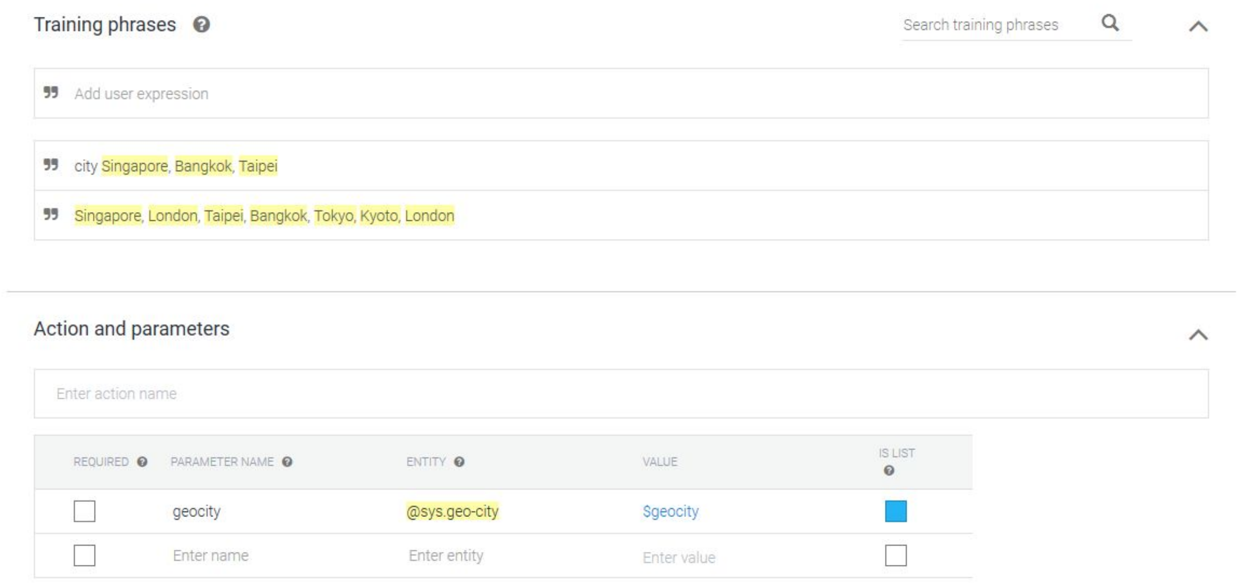
3.1 Cognitive System

Dialogflow

Dialogflow is a serverless natural language processing (NLP) platform that can be used to build conversational applications. This product has a cognitive system that is powered by Google's infrastructure. It enables us to be able to rapidly create and train a text-based conversational user interface (CUI) combined with a graphical user interface (GUI) component by embedding it on a webpage. It serves as the main interaction point with the end-users. CUI is chosen over form based input in conventional GUI interface because CUI can be further developed and scalable to handle more comprehensive and complicated requests from users. This would in turn improve user experience and usability of the system as a whole.

How it Works

The chat agent will illicit input from users by guiding them through a series of questions. Users' input are matched with one of the intents shown in the flow diagram below by using either system entities such as or custom entities created by us. Each intent is created based on the desired user input. For example, city-intent is to get the cities to be travelled. The intent is matched by the system entity, sys.geo-city which will recognize city names. The user input will be saved as the parameter, geocity's value. The intent name and parameter value will be sent in JSON format to the web server via a webhook call as a HTTP POST request for processing and subsequently, a corresponding response will be received from the server. The response, typically a fulfillment text will be rendered as a text reply from the chatbot to the user. It will ask the user to revise the input if it fails the validation process in the fulfillment function on the web server. Otherwise, it will pose the next question to illicit the subsequent required input.


Training phrases

Search training phrases

Add user expression

city Singapore, Bangkok, Taipei

Singapore, London, Taipei, Bangkok, Tokyo, Kyoto, London

Action and parameters

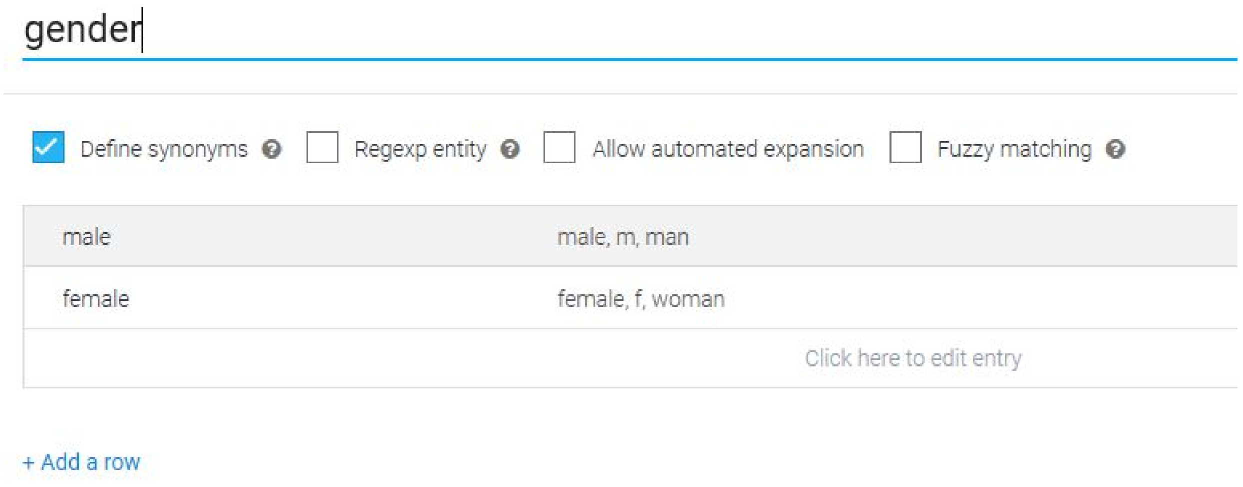
Enter action name

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	geocity	@sys.geo-city	$geocity	■
☐	Enter name	Enter entity	Enter value	☐

Figure 2: Example of System Entity and Intent

System Entities and Custom Entities

Most of the intents created for TravelAI utilizes system entities such as geo-city and time-period. However, certain input requires custom entities to be created in order to recognise those words. For example, one of the custom entities created for this application is gender. This entity is used to match the intent, 'sex-intent' to get the user's gender. The words for gender, male and female were added under this entity. Synonyms of genders that users might input were also added in order to recognise the range of possible answers that may be input by the user such as m, f, woman and man.



gender

[✓] Define synonyms   [ ] Regexp entity   [ ] Allow automated expansion   [ ] Fuzzy matching

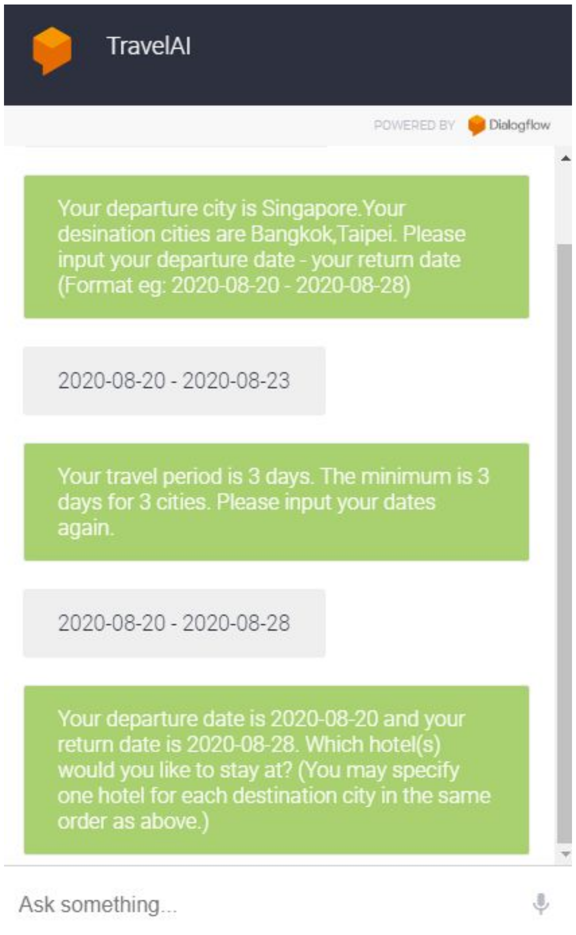
male	male, m, man
female	female, f, woman
	Click here to edit entry

+ Add a row

Figure3: Example of Custom Entity

Webhook and Fulfillment Service

The chat agent will send a HTTP POST request with JSON Webhook from Dialogflow's server to the webhook service hosted by the web server whenever an intent is matched from user's input. Subsequently, the web server will map the request with the corresponding fulfillment function by its matched intent. Typically, the fulfillment function will extract the parameters' values out from the JSON body and process it through business rules, if any. If it fails the condition listed in the business rules, it will provide a response back to the user via the chatbot to ask the user to provide the correct input. Otherwise, it will provide another question to elicit the next required input from the user via a response to the chatbot. For example, if a user provided a travelling period of less than 5 days which is the minimum to travel for 3 cities, it will fail the rules and trigger a response, asking the user to input again for the travelling dates to be more than 5 days.



Your departure city is Singapore.Your desination cities are Bangkok,Taipei. Please input your departure date - your return date (Format eg: 2020-08-20 - 2020-08-28)

2020-08-20 - 2020-08-23

Your travel period is 3 days. The minimum is 3 days for 3 cities. Please input your dates again.

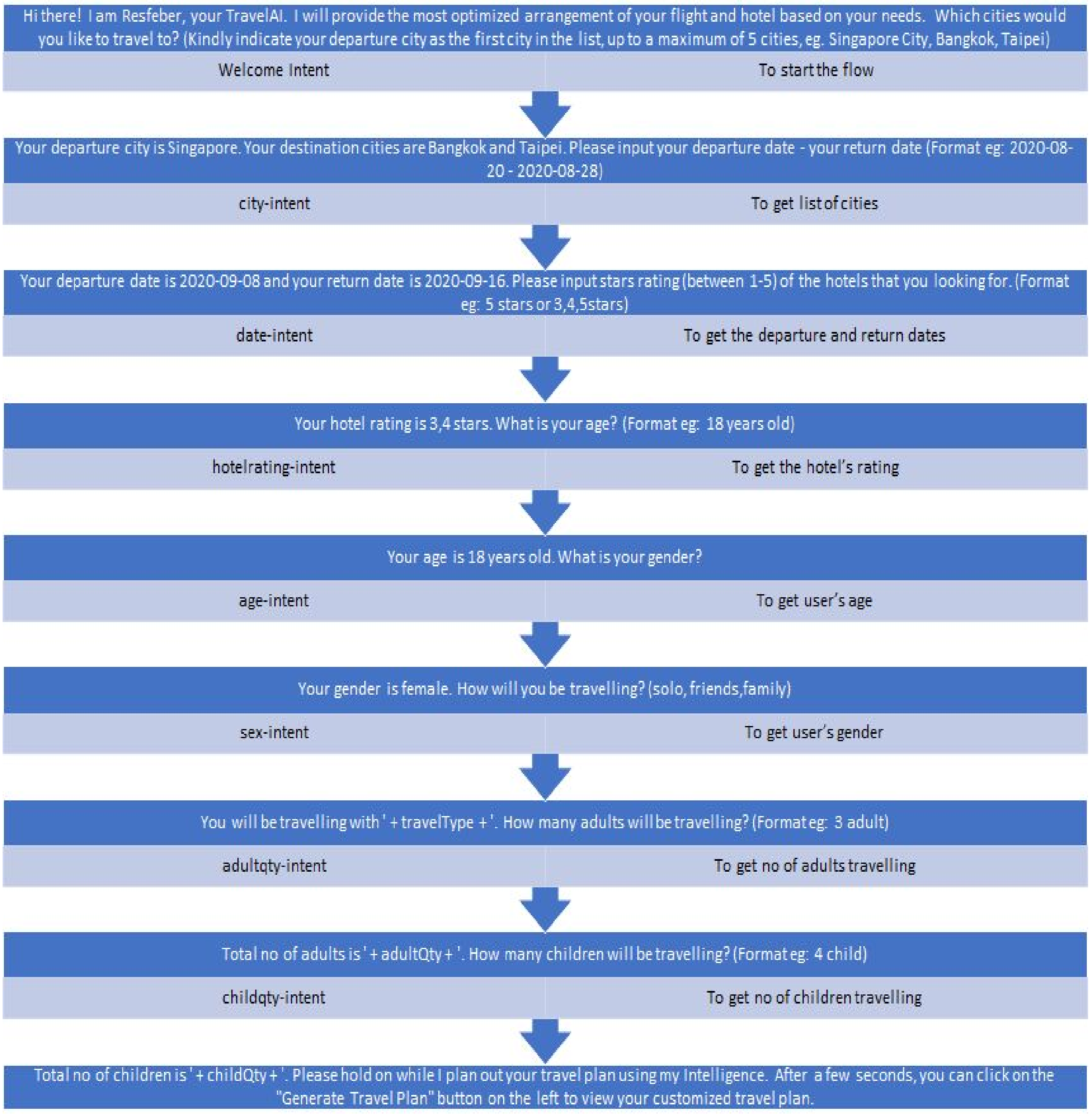
2020-08-20 - 2020-08-28

Your departure date is 2020-08-20 and your return date is 2020-08-28. Which hotel(s) would you like to stay at? (You may specify one hotel for each destination city in the same order as above.)

Figure4: Example of Conversation with Invalid Input

Conversation Flow

An example of the typical conversation would flow as illustrated below. Any invalid input would be processed by the business rules set in the web server and the chatbot will prompt the user to input again based on the expected input format. Each intent will be triggered along the way until all the necessary inputs have been collected. Exception to this would be for solo traveller as there is no need to indicate quantity of adults and children traveller. The flow would exit upon getting the input, ‘solo’.



Hi there! I am Resfeber, your TravelAI. I will provide the most optimized arrangement of your flight and hotel based on your needs. Which cities would you like to travel to? (Kindly indicate your departure city as the first city in the list, up to a maximum of 5 cities, eg. Singapore City, Bangkok, Taipei)

Welcome Intent	To start the flow

Your departure city is Singapore. Your destination cities are Bangkok and Taipei. Please input your departure date - your return date (Format eg: 2020-08-20 - 2020-08-28)

city-intent	To get list of cities

Your departure date is 2020-09-08 and your return date is 2020-09-16. Please input stars rating (between 1-5) of the hotels that you looking for. (Format eg: 5 stars or 3,4,5stars)

date-intent	To get the departure and return dates

Your hotel rating is 3,4 stars. What is your age? (Format eg: 18 years old)

hotelrating-intent	To get the hotel's rating

Your age is 18 years old. What is your gender?

age-intent	To get user's age

Your gender is female. How will you be travelling? (solo, friends,family)

sex-intent	To get user's gender

You will be travelling with ' + travelType + '. How many adults will be travelling? (Format eg: 3 adult)

adultqty-intent	To get no of adults travelling

Total no of adults is ' + adultQty + '. How many children will be travelling? (Format eg: 4 child)

childqty-intent	To get no of children travelling

Total no of children is ' + childQty + '. Please hold on while I plan out your travel plan using my Intelligence. After a few seconds, you can click on the "Generate Travel Plan" button on the left to view your customized travel plan.

Figure5: Conversation Flow and Intents

3.2 Flight API, Hotel API, Places of Interest API

Flights API

Flight Offers Search API from Amadeus was chosen as the API to obtain real-time flight details. It is a traditional flight search API that will give you a list of flight offers from a wide choice of airlines and that meet the specific travel search criteria (origin, destination and dates). The results contain the flight details and will allow the user to select their preferred option based on either price or convenience. In TravelAI, the search is narrowed to only direct flights to allow for more time in the travel destination. Prices returned are denominated in SGD.

Given that the number of flights to search for can quickly scale to an alarming number ((n+1)! x number of travelling days, where there are n number of cities to visit), some measures are taken to reduce the number of searches made. During the first day of travel, only flights from home city to each travel destination are searched. Similarly, during the last day of travel, only flights from each travel destination to the home city are searched. For the remaining days in between, flights between all the travel destinations are searched. The diagram below shows the types of flights searched for a travel date. Dotted lines represent searches for one way flight while solid lines represent searches for two-way flight.

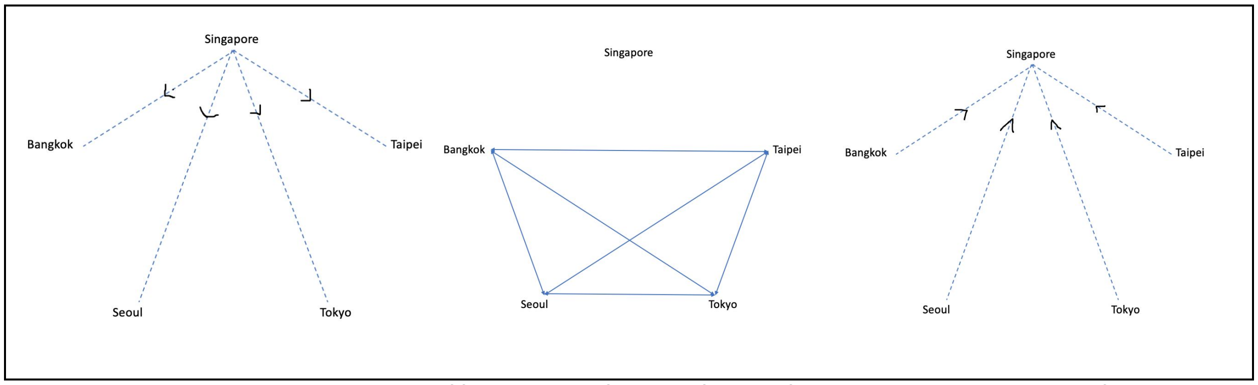


Figure 6: Diagram illustrating types of flight searched for on the first day, following days, and the last day of travel.

The Flight Search returns the departure details, arrival details, travel duration, airlines, and price of flight, which is consolidated into a dataframe and passed to the Optimizer. A sample of the output Flight Search can be found below

.

Departure Date	Arrival Date	Departure Time	Arrival Time	Duration	Type of Flight	Price	Airline	Departure Airport	Arrival Airport	Departure City	Arrival City
1/8/20	1/8/20	12:20:00	13:50:00	2H30M	direct	90.9	THAI LION AIR	SIN	DMK	SIN	BKK
1/8/20	1/8/20	20:30:00	22:00:00	2H30M	direct	90.9	THAI LION AIR	SIN	DMK	SIN	BKK
1/8/20	1/8/20	22:20:00	23:45:00	2H25M	direct	131.2	SCOOT	SIN	DMK	SIN	BKK
1/8/20	1/8/20	06:30:00	08:00:00	2H30M	direct	141.2	SCOOT	SIN	BKK	SIN	BKK
1/8/20	1/8/20	08:20:00	09:50:00	2H30M	direct	141.2	SCOOT	SIN	BKK	SIN	BKK

Figure 7: Sample dataframe output from calling Flight API

Hotels API

Hotel Search API from Amadeus was chosen as the API to obtain real-time hotel details. The Hotel Search API returns a list of available hotels and prices for a given location and date range. The API returns hotel rates, name, address, room type, amenities list and other hotel information. The cheapest available option for each hotel is shown by default. The results can be further refined using filters like hotel category, hotel chain name, facilities or budget. Since TravelAI is optimizing for the specific hotel that user has chosen, TravelAI searches for the details of the specific hotel. Only when the user-specified hotel is not found or not available for booking does TravelAI search for the cheapest hotel in the city with a 4 star rating instead.

The number of searches that TravelAI makes can be represented as n x number of travelling days, where n is the number of cities to visit. The Hotel Search returns the check-in and check-out dates, along with the room type and prices. A sample of the output Hotel Search can be found below.

Hotel	Check In Date	Check Out Date	Category	Price	City
Novotel Bangkok Silom Road	1/8/20	2/8/20	SUPERIOR_ROOM	63.22015165	BKK
Novotel Bangkok Silom Road	2/8/20	3/8/20	SUPERIOR_ROOM	63.22015165	BKK
Novotel Bangkok Silom Road	3/8/20	4/8/20	SUPERIOR_ROOM	63.22015165	BKK
Novotel Bangkok Silom Road	4/8/20	5/8/20	SUPERIOR_ROOM	63.22015165	BKK
Novotel Bangkok Silom Road	5/8/20	6/8/20	SUPERIOR_ROOM	63.22015165	BKK

Figure 8: Sample dataframe output from calling Hotel API

Places of Interest API

The TripAdvisor API on rapidapi.com was chosen as the API to obtain details of popular places of interest in a given city. The returned data comes a “Category” column which categorizes the place of interest into at least one of the following 10 categories: Sights & Landmarks, Nature & Parks, Water & Amusement Parks, Zoos & Aquariums, Fun & Games, Outdoor Activities, Museums, Tours, Shopping, Concerts & Shows. The rule-based system uses these categories as a gauge for whether a user would be interested in visiting the place of interest based on his user’s profile.

Name	Website	Image	Category	City
Gardens by the Bay			['Sights & Landmarks', 'Nature & Parks']	singapore
Universal Studios Singapore			['Water & Amusement Parks']	singapore
S.E.A. Aquarium			['Zoos & Aquariums', 'Nature & Parks']	singapore
Singapore Botanic Gardens			['Nature & Parks']	singapore
Adventure Cove Waterpark	Omitted in report for visual clairty		['Water & Amusement Parks', 'Fun & Games']	singapore

Figure 9: Sample dataframe output from calling Hotel API

3.3 Optimiser

The optimiser uses Google's OR-Tools CP-SAT Solver. The CP-SAT solver uses a combination of a Boolean Satisfiability (SAT) solver with Lazy Clause Generation. The reason this was chosen over other tools is that it performs better than Google's CP Solver and it is able to be programmed on Python over using Java with OptaPlanner.

The optimiser takes the outputs from the Flight and Hotel API to find the best order to visit each city and how many days to stay in each city to minimise the flight and hotel costs. This ensures the user receives the cheapest option when travelling with TravelAI. The optimiser is broken down into 4 steps:

Step 1: Transformation

The first section is the transformation of the data. As the CP-SAT Solver is not able to work with pandas DataFrames, the data to be used, in this case the flight and hotel costs, had to be translated into arrays. The DataFrame is transformed into lists and sublists that represent the day, current location, cost of flight to next location for the flight data; and current location and hotel costs for each day for the hotel data. The first and last flight to the home city is stored separately from the main flightData. Below is a sample of how the flightData would look like:

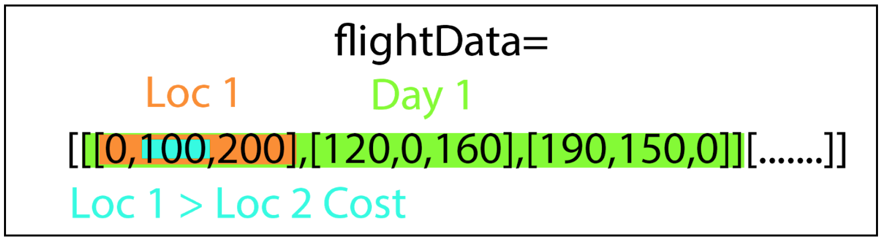


Figure 10: Sample of flightData

Step 2: Creating the Model

After transforming the data, we would need to create the model. As this is set up as an CP-SAT Solver, we would need to add constraints for the model to solve.

The (hard) constraints are:

- No overlapping of days per City
- Everyday must be used for travel
- The first and last flight is from/to the Home city.
- No flights the day after/before flying.

As the solver is not able to directly get the costs of the flights and hotels from the arrays that we created, we have to create a placeholder IntVar that represents which flight/hotel was selected. We then used boolean logic to link the array to a new IntVar value that will be used to minimise the model. One of the drawbacks of the CP-SAT is that it is unable to do boolean comparisons for if statements between an IntVar and integer (ie. IntVar1==1), a roundabout method to do this was required.

```
If IntVar1 & IntVar2:

    IntVar3=flightdata.Price(1)

----------------------------------Becomes--------------------------------------------------

model.AddBoolOr([IntVar1.NOT(),IntVar2.NOT()]).OnlyEnforceIf(AndRule.NOT())

model.AddBoolAnd([IntVar1,IntVar2]).OnlyEnforceIf(AndRule)

model.Add(IntVar3==flightdata.Price(1)).OnlyEnforceIf(AndRule)
```

Figure 11: Sample of flightData

Step 3: Solving the Problem

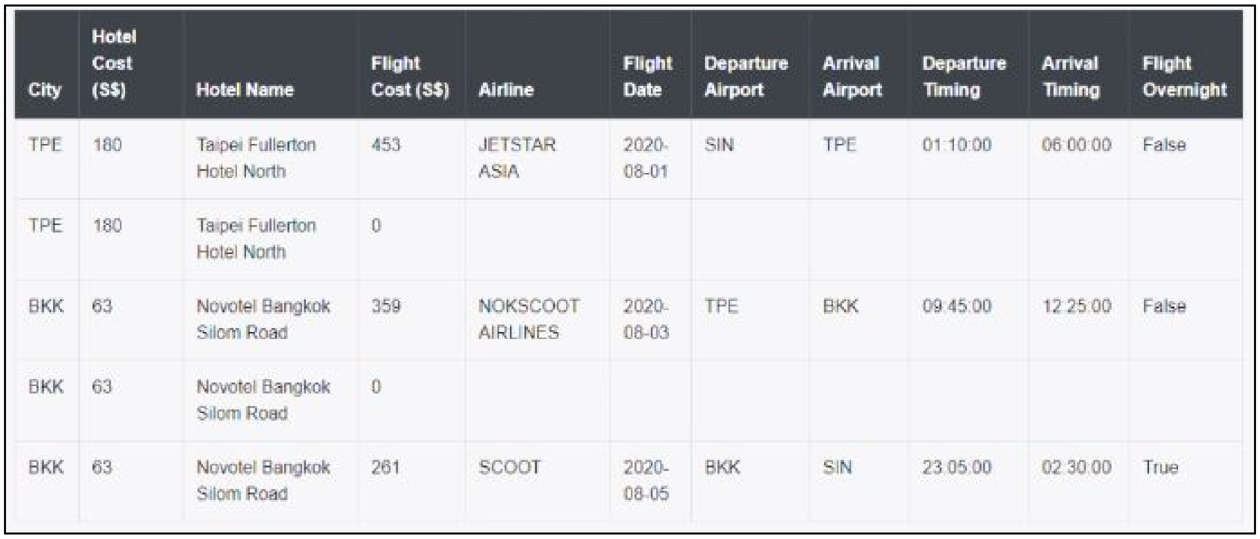
Once we have created the model we will have to solve it. Once the data has been entered, will get the minimal value of the output based on:

- Hotel Costs
- Flight Costs
- Difference between longest and shortest stay in a city * 100

The last point is a soft constraint that allows us to not stay in a single city the majority of the time with a weight of 100.

Step 4: Creating the Output

After the solver has solved the problem, we will need to create a JSON output to send it back to the user. This is done by comparing the location, day as well as costs of the flights and hotels to the pandas DataFrame that was originally received to get the additional data required to show to the user. Below is a sample output of the data.



City	Hotel Cost (S$)	Hotel Name	Flight Cost (S$)	Airline	Flight Date	Departure Airport	Arrival Airport	Departure Timing	Arrival Timing	Flight Overnight
TPE	180	Taipei Fullerton Hotel North	453	JETSTAR ASIA	2020-08-01	SIN	TPE	01:10:00	06:00:00	False
TPE	180	Taipei Fullerton Hotel North	0							
BKK	63	Novotel Bangkok Silom Road	359	NOKSCOOT AIRLINES	2020-08-03	TPE	BKK	09:45:00	12:25:00	False
BKK	63	Novotel Bangkok Silom Road	0							
BKK	63	Novotel Bangkok Silom Road	261	SCOOT	2020-08-05	BKK	SIN	23:05:00	02:30:00	True

Figure 12: Sample of Optimizer in table form

3.4 Rule-Based System

The rule-based system is used to recommend Places of Interests to the user based on his travel destination and his profile. The system gathers data from the TripAdvisor API mentioned in Section 2.2, and consults a knowledge-base derived from a survey[5] on travel preferences. The knowledge-base is stored using python library durable-rules.

Survey on Travel Preferences

The purpose of the survey was to gather travelling preference for each type of traveller, based on his age group, sex, and travel companions, if any. It was a quantitative survey, hosted on google forms for the period of 20th April 2020 to 27th April 2020. Most of the participants are connections of the developers of TravelAI. Participants were asked basic questions on their sex, gender, and people they travel with. They were also asked to rate, on a scale of 1 to 10, how likely they were to engage in the following activities while travelling. Giving a rating for each category is compulsory; Visit Sights & Landmarks, Head to Nature & Parks, Have fun at Water & Amusement Parks, See the animals at Zoos and Aquariums, Find all the Fun & Game locations, Head out for Outdoor Activities, Check out Museums, Go on Tours and Do some Shopping.

[5] Survey questions be found here: https://docs.google.com/forms/d/e/1FAIpQLSdC2Y7xog7j_TOfePEiTvBtVzQNIbYZGJ3FVbrEiBRr2jBweA/viewform

Knowledge Extraction and Knowledge Base

We determine the preference of a user profile by taking the average of the ratings within each type of user profile. The top three types of activities with the highest ratings are taken as the preferred activities for the user type. This includes all activities that are tied in the third position.

In gathering responses for the survey, not all user types were successfully reached out to. For these user types, we find the user type that they are most similar to, and use *forward-chaining* to direct the rules to this user type. For example, for a female traveller under 18 who is travelling with friends, forward chaining is used to direct the rules to a male traveller under 18 who is travelling with friends. The assumption here is the female traveller would travel with friends of similar age, of which would include males, and the places of interest they would like to visit as a group would be similar.

3.5 Web Server

The Web Server is developed using Node.js with Express.js framework and also various tools and packages such as EJS and bodyParser. It acts as the middleware serving multiple functions as listed below.

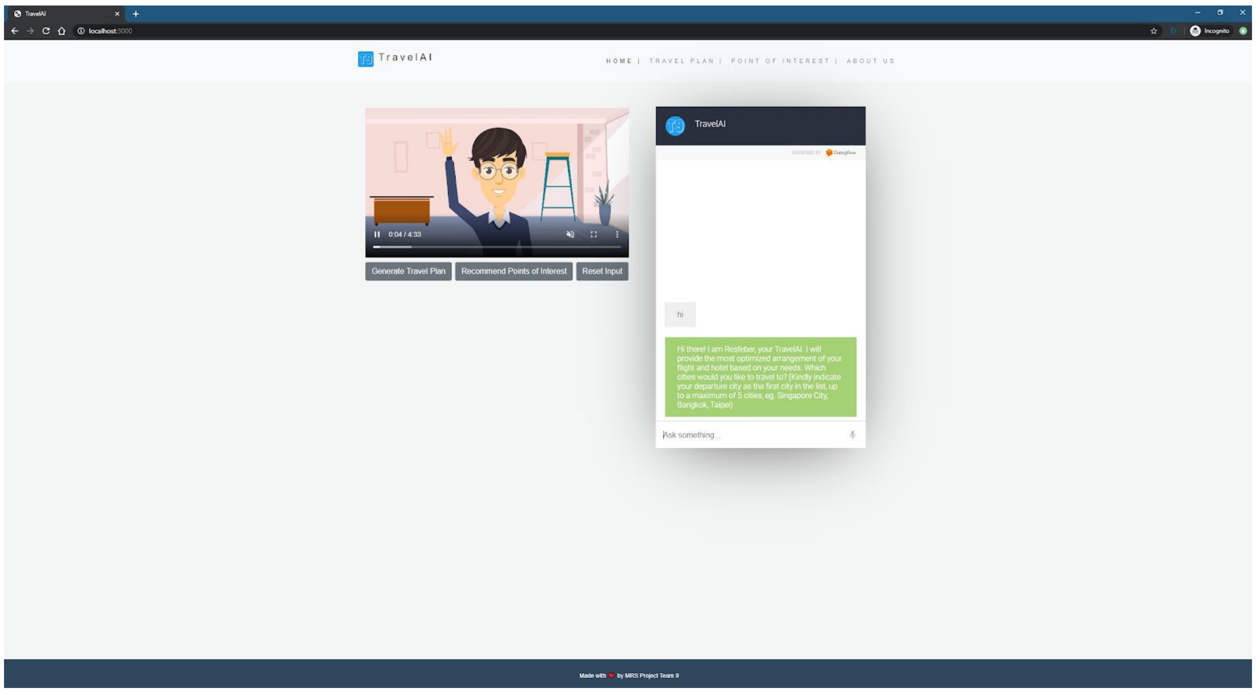
- serving front-end GUI webpage;
- webhook and fulfillment service;
- processing business rules;
- web-server to handle HTTP requests;
- calling python scripts via child process to call various API and run the optimizer.
- Handles degraded mode for offline user

Web-server to handle HTTP requests

It primarily handles the HTTP requests on this system to fulfill other functions such as webhook service or to fulfill clients requests to serve web pages.

Server for Front-end Web Page

This server is hosting the files such as HTML and CSS which is used to render the web pages to be displayed via a browser client to end users. Web pages in TravelAI are built with responsive web design in mind using various tools and frameworks such as HTML5, CSS3, ES6, EJS, jQuery and Bootstrap. EJS allows the server to dynamically generate different pages with a uniform template. When a user requests the homepage via a browser client, the server will render the page based on the HTML template files with the embedded video and chatbot shown below.

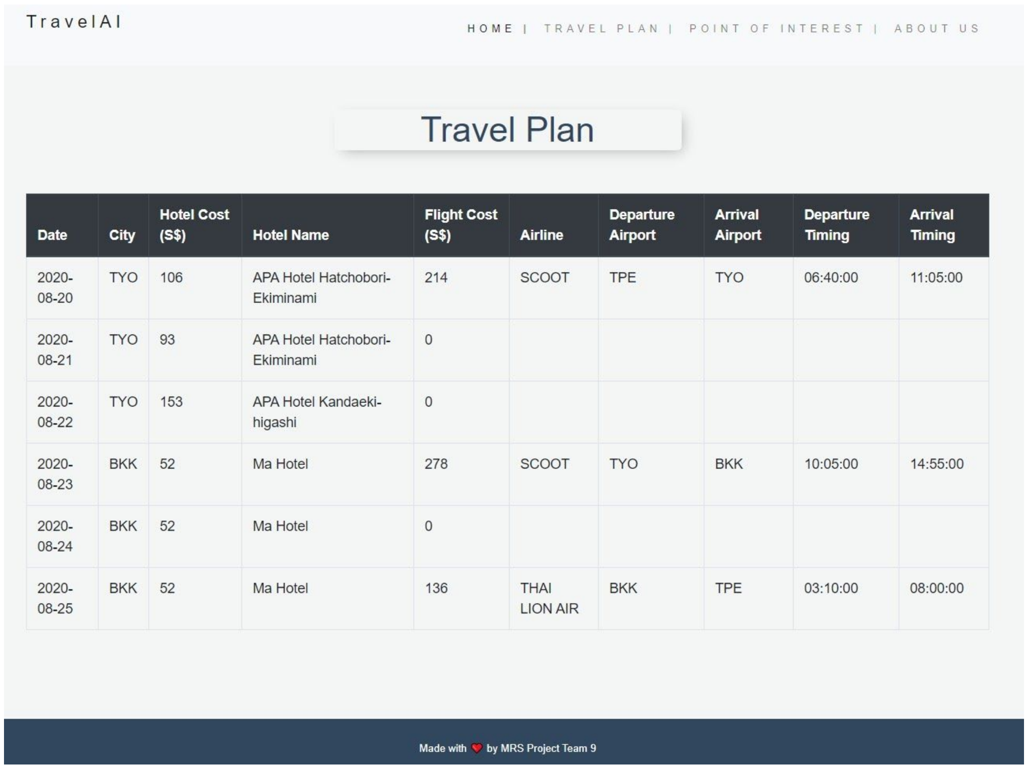

TravelAI
localhost:3000
TravelAI
HOME | TRAVEL PLAN | POINT OF INTEREST | ABOUT US
0:04 / 4:33
Generate Travel Plan
Recommend Points of Interest
Reset Input
TravelAI
Ask something...
Made with ❤ by MRS Project Team 9

hi

Hi there! I am Resfeber, your TravelAI. I will provide the most optimized arrangement of your flight and hotel based on your needs. Which cities would you like to travel to? (Kindly indicate your departure city as the first city in the list, up to a maximum of 5 cities, eg. Singapore City, Bangkok, Taipei)

Figure13: Homepage of TravelAI

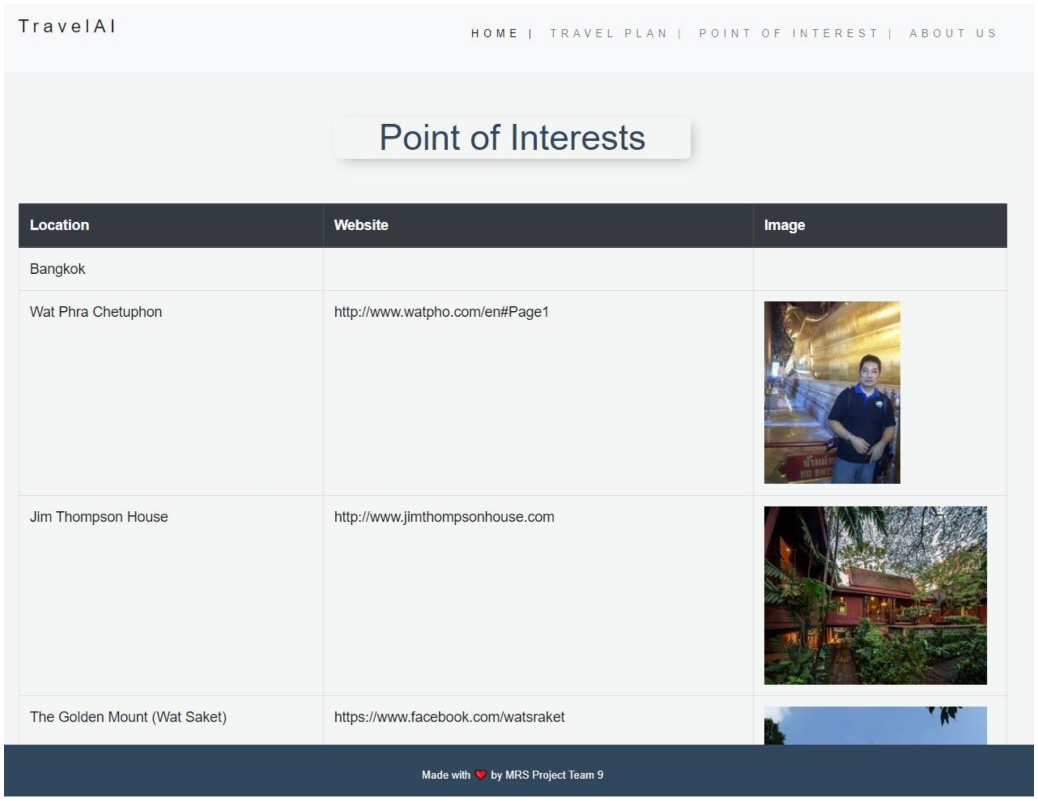
Subsequently, when the result data is returned by the optimizer and API servers, the web server will be able to render the Travel Plan and POI web page using templates. The tables are dynamically generated using data passed to the template utilizing EJS as shown in the screenshots below.



Travel Plan

Date	City	Hotel Cost (S$)	Hotel Name	Flight Cost (S$)	Airline	Departure Airport	Arrival Airport	Departure Timing	Arrival Timing
2020-08-20	TYO	106	APA Hotel Hatchobori-Ekiminami	214	SCOOT	TPE	TYO	06:40:00	11:05:00
2020-08-21	TYO	93	APA Hotel Hatchobori-Ekiminami	0					
2020-08-22	TYO	153	APA Hotel Kandaeki-higashi	0					
2020-08-23	BKK	52	Ma Hotel	278	SCOOT	TYO	BKK	10:05:00	14:55:00
2020-08-24	BKK	52	Ma Hotel	0					
2020-08-25	BKK	52	Ma Hotel	136	THAI LION AIR	BKK	TPE	03:10:00	08:00:00

Figure14: Travel Plan Page



TravelAI
HOME | TRAVEL PLAN | POINT OF INTEREST | ABOUT US

Point of Interests

Location	Website	Image
Bangkok		
Wat Phra Chetuphon	http://www.watpho.com/en#Page1	
Jim Thompson House	http://www.jimthompsonhouse.com	
The Golden Mount (Wat Saket)	https://www.facebook.com/watsraket	

Made with ❤ by MRS Project Team 9

Figure15: Points of Interest Page

Webhook, Fulfillment Service and Business Rules

The server listens and accepts HTTP POST requests with JSON Webhook sent from Dialogflow's server. The parameters' values will be extracted out from the JSON body and process it through business rules, if any. Example of business rules that are enforced in this application are listed below:

- Users may choose a minimum of 3 cities and maximum of 5 cities.
- Travelling dates must be in future.
- Travelling period must be more than 5 days at minimum. Each additional cities will require 2 additional days to the minimum travelling period requirement.
- Departure date must before arrival date.
- Solo travellers do not need to go through the last two questions to input quantity of adults and children in the travelling group.

Running Python Scripts and API Call

The web server will save all the required input from users which were extracted with the parameters' value sent via chatbot's webhook into a list of variables. The server will spawn two child processes which will run TravelAI.py and TravelAI_POI.py respectively. These values will be passed into the child processes to be used as input arguments by the two python scripts and API calls. When the scripts complete their process and exits, they will pass the output data back to the server via stream. Finally, the server will be able to generate the result table and render the Travel Plan and POI page with those data as described in the subsection above.

Degraded Mode for Offline user

As this application's core components such as Dialogflow and web API are web-based, it necessitates the user to have internet connection in order for it to work. Due to that reason, the system does not have any meaningful level of degraded mode to serve the user under offline scenarios. As such, the server employs the method of service worker where it will cache an offline HTML file in the user's local device during the user's first visit to the webpage. During subsequent user visits under an offline situation, the service worker will intercept the request and display the webpage rendered from the offline HTML file. This webpage is to inform users that they are offline and to come back and try again when their internet connection is resumed.

5. Conclusion

In this project, we propose and implement an effective solution through a user-friendly conversational-agent, TravelAI, to address the complexity of getting an optimal multi-destination travel package. Optimal here is defined as the cheapest set of flight and hotel that provides roughly equal number of days at each destination. As an added bonus, TravelAI also provides a list of recommended Place of Interest, specially curated for the user profile by drawing intelligence from an in-house knowledge-base. This system can be expanded in future with ease to increase business value and monetization potential of this solution. The web server can make use of other API from Amadeus to integrate booking functions directly into this system. This would allow the users to immediately make bookings of the flights and hotels that were shortlisted by TravelAI.

APPENDIX

Appendix A: Project Proposal

GRADUATE CERTIFICATE: Intelligent Reasoning Systems (IRS)
PRACTICE MODULE: Project Proposal

Date of proposal: 9 May 2020
Project Title: ISS Project – TravelAI: Intelligent Booking Platform and Recommender System for Multi-Destination Travel delivered through a conversational agent
Sponsor/Client: *(Name, Address, Telephone No. and Contact Name)* Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace, Singapore NATIONAL UNIVERSITY OF SINGAPORE (NUS) Contact: Mr. GU ZHAN / Lecturer & Consultant Telephone No.: 65-6516 8021 Email: zhan.gu@nus.edu.sg
Background/Aims/Objectives: The proposed intelligent eco-system will make use of knowledge, techniques and skills of modular courses: MR, RS, CGS.
Requirements Overview: · Research ability · Programming ability · System integration ability

Resource Requirements (please list Hardware, Software and any other resources) Hardware proposed for consideration: · CPU Software proposed for consideration: · Ngrok.exe · Nodejs · Nodejs packages · Python · Python Libraries: · pandas==0.25.0 · durable-rules==2.0.27 · ortools==7.6.7691
Number of Learner Interns required: (Please specify their tasks if possible) A team of 3 members

Methods and Standards:

Procedures	Objective	Key Activities
Requirement Gathering and Analysis	The team should meet with ISS to scope the details of project and ensure the achievement of business objectives.	1. Gather & Analyze Requirements 2. Define internal and External Design 3. Prioritize & Consolidate Requirements 4. Establish Functional Baseline
Technical Construction	· To develop the source code in accordance to the design. · To perform unit testing to ensure the quality before the components are integrated as a whole project	1. Setup Development Environment 2. Understand the System Context, Design 3. Perform Coding 4. Conduct Unit Testing
Integration Testing and acceptance testing	To ensure interface compatibility and confirm that the integrated system hardware and system software meets requirements and is ready for acceptance testing.	1. Prepare System Test Specifications 2. Prepare for Test Execution 3. Conduct System Integration Testing 4. Evaluate Testing 5. Establish Product Baseline
Acceptance Testing	To obtain ISS user acceptance that the system meets the requirements.	1. Plan for Acceptance Testing 2. Conduct Training for Acceptance Testing

		3. Prepare for Acceptance Test Execution 4. ISS Evaluate Testing 5. Obtain Customer Acceptance Sign-off
Delivery	To deploy the system into production (ISS standalone server) environment.	1. Software must be packed by following ISS's standard 2. Deployment guideline must be provided in ISS production (ISS standalone server) format 3. Production (ISS standalone server) support and troubleshooting process must be defined.

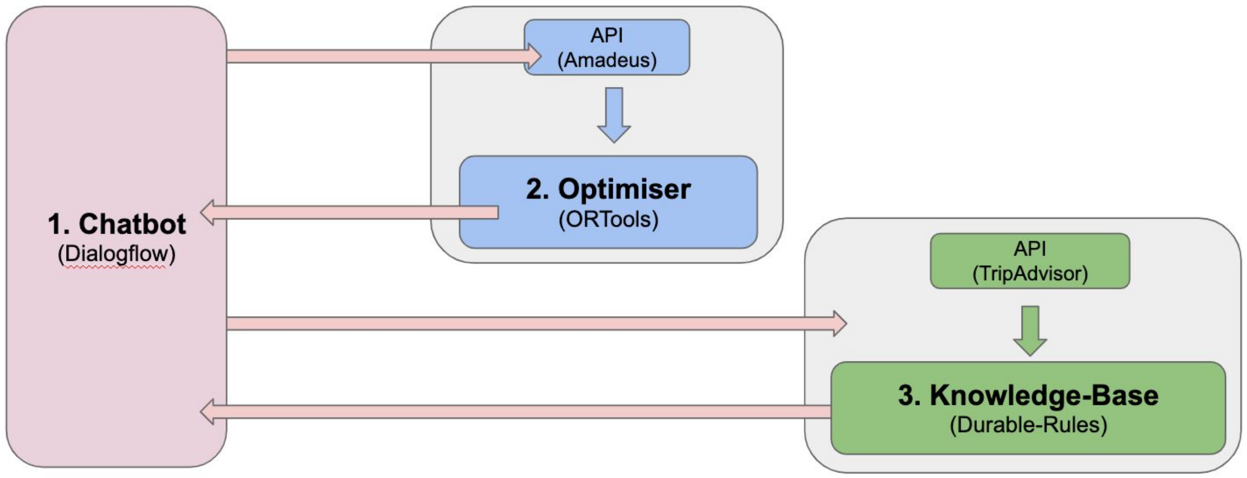
Team Formation & Registration

Team Name: TravelAI
Project Title (repeated): TravelAI
System Name (if decided): TravelAI

Team Member 1 Name: Mohamed Mikhail Kennerley
Team Member 1 Matriculation Number: A0213548E
Team Member 1 Contact (Mobile/Email): e0508649@u.nus.edu
Team Member 2 Name: Loy Pei Xin Veronica
Team Member 2 Matriculation Number: A0213546J
Team Member 2 Contact (Mobile/Email): E0508647@u.nus.edu
Team Member 3 Name: Oh Chun How
Team Member 3 Matriculation Number: A0213524R
Team Member 3 Contact (Mobile/Email): Chunhow.oh@u.nus.edu

For ISS Use Only		
Programme Name:	Project No:	Learner Batch:
Accepted/Rejected/KIV:		
Learners Assigned:		
Advisor Assigned: Contact: Mr. GU ZHAN / Lecturer & Consultant Telephone No.: 65-6516 8021 Email: zhan.gu@nus.edu.sg		

Appendix B: Mapped System Functionalities against knowledge, techniques and skills of modular courses: MR, RS, CGS



	Function	**Knowledge, Technique, Skills**
1. Chatbot	Converse with user to obtain input	**1. Cognitive Framework** **2. Business Rules** - Min 5 days, min 3 cities - Additional 2 days for each city - Travel dates must be in future
2. Optimizer	Finds lowest cost package of flight and hotel, that provides roughly equal number of days in each city	**1. Search**, with **2. Constraint Satisfaction:** - Hard Constraints: ● No overlapping of days per city, ● Everyday must be used for travel ● The first and last flight is from/to the home city ● No flights the day after/before flying - Soft constraints: ● Difference between longest and shortest stay in a city * 100
3. Knowledge -Base	Recommends Places of Interest based on user profile	**1. Knowledge base** gathered from results of survey data **2. Forward-Chaining** used for user profiles that were not reached out to.

Appendix C: Installation and User Guide

Please refer to this link:
https://github.com/mecarill/IRS-PM-2020-01-18-IS02PT-GRP09-TravelAI/blob/master/ProjectReport/User%20Guide.pdf

Appendix D: Individual project reports

Mikhail Kennerley Individual Report

During this project I was responsible for the optimisation of the collected flight and hotel data to produce the cheapest flight/hotel plan from the selected data. I also created the survey that was used to recommend points-of-interest to the user based on their age, gender and travelling group (solo, family or friends). The last responsibility was the creation of the video that promotes our project and goes through the process of it generating the holiday plan for the user and how it works.

Using Google OR-Tools for the optimiser has let me learn the ways in which constraint satisfaction optimisers work and the benefits of it over other types of optimisers in certain use-cases. With OR-Tools allowing me to code in Python, I was able to improve my coding ability while learning to implement search and constraint satisfaction in a real world use-case. Due to how OR-Tools works, not every problem can be easily solved such as boolean comparison, this has taught me to think outside the box on how to solve this issue while teaching me to find solutions when there are solutions available rather than reinventing the wheel.

The creation of the survey to gather data from participants was another unique experience as I had to create a survey that was simple enough for participants to want to complete, as there was no incentive for them to do so, while ensuring the gathered data would be useful for our project.

What I felt that I learnt a lot from was while creating the video as I had to understand what my other team members were doing and how they were completing their tasks for me to showcase it during the 5-minute video. This gave me an inside peek and how that section of the project worked, rules and APIs from Veronica and web servers and front-end from Chun How, which I had not been exposed to before.

From what I have learnt during this project, I can imagine implementing it in my workplace by creating something similar to the job shop problem, where there are many tasks to be completed on an item and a certain order that it needs to be completed in. Working in SMRT, the trains are our main asset and there are many sub-assets that require innovative solutions such as this. Additionally, learning what I have during this project, I would be able to incorporate business rules unique to my organisation and even a front-end that would encourage users to actually use the solution by making it both attractive and easy for them.

Loy Pei Xin Veronica Individual Report

For this project, I was responsible for data collection, creating the recommender system, the project report, and the user guide. Data collection meant getting the hotel data, flight data and Places of Interest data into a form that is ready for use by the optimizer and rule-based system. The recommender system recommends Places of Interest to users based on their user profile, using a rule-based approach.

Working on the data collection pipeline taught me to structure my code such that it is easily understandable and easily editable. This is because data collection was the first part of the project and will need to be re-visited again and again, depending on the needs of the optimizer and the needs of the chatbot. Writing the project report enabled me to see clearly how different parts of the project fit together into a single product, and also taught me about coming up with a value proposition for our product by doing market scan and highlighting the problem that our product solves. Coming up with a rule-based system was new to me, and in doing a literature review on it, I was able to see how the concepts taught in class were implemented by the different libraries. To me, concepts like propositional logic seemed simple when taught in class, but by delving into the libraries such as PyKE, I could see that implementing the concepts was not as straight-forward. Lastly, my teammates each had different strengths, different skills, and different points of view. Some were more concerned with the user-experience and skilled in front-end work, and others were better with coding in the back-end system. The exposure to the different tools and ways of approaching a problem allowed me to become a better developer and a better team player.

Applying this to my workplace, I can see using APIs are another avenue to generate more data, or generate additional features, especially when the amount of training data that clients provided is insufficient. Another idea that can be applied at the workplace is, when matching trainees to a project, we can use a recommender system that takes into account various metrics, and finally recommends a project for the trainee. Some metrics to be placed in the rule-based system could be the aptitude of each trainee (say, how fast trainee completes the project, lint score of their code, accuracy of their model), how well mentors of the project work together with the trainees, personal interest of each trainee, other skills and time constraints. This would serve as an additional guide for trainees to pick a project, in addition to relying on hunch or other heuristics.

Oh Chun How Individual Report

In this project, I am responsible for developing the front-end web pages, web-server, business logic/rules, the chatbot/ cognitive portion, and finally integrating the entire system together. I have designed the conversation flow to illicit based on inputs required by the API, optimizer and POI API. I have coded some business rules to accept/reject the inputs to ensure the inputs are within the working constraint of the back end system. System integration was done by ensuring the correct input and output interface specification with the two python scipts that would be called by the web server. The result pages were dynamically generated based on the output.

I have learnt a lot and have a deeper understanding of the concepts taught in class such as the conversational agent and optimization that was put into practice in this project. For the first time, I was doing end to end system integration from the ground up, especially to integrate with new tools such as the optimiser. Through this, I have learnt through actual implementation of these tools and how to integrate them into existing conventional applications. This new combination of technologies and tools have given me the insight in solving many of the real world problems. Also, I have managed to put design thinking into practice during the design of the system, front end and the chatbot's conversational flow. The biggest take away that I have gained from this project is the foundation in the understanding and technical skill sets to develop an intelligence system design. An intelligent system is basically composed of three parts; the system input from human or environment; the machine processing and logic and the output.

From what I have learned through this project, there would be a number of areas in my workplace that I can implement these new tools to improve the systems in the land transport industry especially the taxi business. Taxi dispatch engines can be improved with an optimizer and driver profiling system based on RBS to dispatch the job to the most suitable and nearest driver with the highest probability of acceptance. This would improve the efficiency as well as satisfaction level of both drivers and customers. The various customer facing media front such as the social media page, the website and app can be integrated with an intelligent chatbot to serve customer needs with higher response time. Currently, the customer service center is manned by at least 50 staff around the clock and it is not scalable. Scalability is crucial especially during crises where customers require immediate attention and have less patience. It would bring damages to the company's reputation. Through implementation of a chatbot, cost can be reduced while customer satisfaction can be increased. Data on the customers' main complaints and query can also be collected easily via the same channel to identify the high priority root cause to be resolved.