

COMP7015: Artificial Intelligence Systems Engineering

Group Report on Tourist Itinerary Planning Application

Team Members

19177307- Billy Allsworth
19182799 - Sam Trowbridge
19179660- Jayalakshmi Vijayan

Word Count:5009

Table of Contents

Requirement Analysis	3
Functional requirements	3
Use Cases	4
Non Functional Requirements	6
Software Design	8
Modules and dependencies	8
Communication between nodes	8
Overview of Implementation	10
Setup and session start	10
Itinerary Creation	11
Itinerary planning tasks	11
Proactive help : Chatbot	13
AI Recommendation	14
Other special cases	16
NLP	16
Testing Report	16
References	22
Appendix: Group Meeting Notes	23

Requirement Analysis

Functional requirements

Many functional requirements have also been deduced by the team from the user story, these functional requirements are what is required to have a fundamental basic working application that provides the necessary services to complete the task at hand. This does not include such aspects as usability, reliability, effectiveness and so on.

(SysT_FR-01) – system should have an overview page or tutorial

A help page or lesson on how to use the system should be available on the website. It should be made available to the user when needed.

(SysT_FR-02) – system should collect user's trip location, date and activity information

The user should be able to enter the destination of their journey, the activity they desire to do there, as well as the date and time of their visit.

(SysT_FR-03) - system should display the list of places and activities at the selected location

The system should provide a list of places to visit or activities to do at that location when the user enters the location, activity, and date of visit.

(SysT_FR-04) - system should allow user to add places to visit in a wish list

The user should be able to choose which places they want to visit and save them to their personal wish list.

(SysT_FR-05) - System provide an itinerary plan to the user

The system should check the locations on the user's wish list and produce an itinerary with the shortest travel time, the most time available at each location, and the ability to access the locations during working hours.

(SysT_FR-06) - System should have a proactive component

The system should feature a proactive component that provides support and notifications when a user is inactive or looks to be seeking assistance.

(SysT_FR-07) - System should have a reactive component

The system should include a reactive component that interacts with the user, taking user input and responding appropriately.

(SysT_FR-08) - System should have a learning component

The system should have a learning component that determines the user's interests based on previous activity patterns such as visited sites and accepted itineraries. It should make recommendations based on what has been learned.

(SysT_FR-09) - System should have NLP text to speech component

The system should have a natural language text to speech interface to interact with the user

(SysT_FR-10) - System should have NLP speech to text component

The system should have a natural language speech to text interface to interact with the user

(SysT_FR-11) - All user information and activities should be saved in the database.

To train the learning component, the system should save all user-related data in the database.

Use Cases

ID (FR)	Use Case	Preconditions	Trigger	Action	Postconditions
01	Give system overview	System on	1.Timer_systemon>30s 2.User_state = inactive	1.chatbot pops up 2.provides help page link	User state= active
02	Collect trip location, date and activity	System on	1.User active	1. user enter location 2. user enter activity 3.user enter date 4. search button to continue	1.Retrieve search results using API 2.display list of places and related information
03	Display list of places and activities	1.System on 2.User entered location, activity and date	1.User clicked search button	1. Get information using google API 2.Display the collected information to user	1.User checks the places and activities

04	Add items to wish list	1.System on 2.Suggested places are listed to user	1.User picks favourite places	1.Personal wish list of the user for that trip is updated 2. wish list can be updated until user clicks generate itinerary	1. User data is saved in the database 2.Data is passed to the itinerary generating logic
05	Provide itinerary	1.System on 2.User's wish list is ready	User clicked the provide itinerary button	1.get the details of the places from google (opening, closing time and travel time) 2.Route planning logic will run and generate a best route	1. Itinerary is generated and displayed to the user.
06	Proactive user help	1.System on 2.User inactive for a while	User inactive more than a threshold time of 2 minutes	1.Chatbot pop ups	Chatbot offers help
07	Reactive user interaction	1.System on 2.User accepts chatbot assistance	User confirms chatbot help	Chatbot gives response to the user queries	chatbot interacts with user
08	Learn the user behaviour and give recommendation	1.System on	1.User history is available or system has good suggestions for the user 2. Self-activation of the recommendation	Recommendation window pops up	User gets recommendation from the system
09	NLP text to speech interaction with user	1.System on	1. system has a message to the user (eg: chatbot message)	System gives speech output using google speech recognition	System talks to the user

10	NLP speech to text interaction with user	1.system on	1. User speech input is processed by the system and converted to text data	System converts speech input to text using google speech recognition	User talks to the system
11	Update database	1. system on	inputs from the user	All the user inputs are stored in the MySQL database	Data is saved

Table 1

Non Functional Requirements

NFR-01: The application should have an account creation and login system to enable the saving of itineraries.

Itineraries are already stored however a user is assigned by the device of which they access the application therefore to provide security and multiple device usage for users a login system would be ideal.

NFR-02: The application should enable the user to choose a location on an interactive map.

This would then provide the user with another method in which they can interact with the application other than speech and text. This could also make the application a little more intuitive, and provide extra ways in which the AI agent could be utilised.

NFR-03: The application should have mobile integration.

Due to time constraints the current design does not take device size into consideration, this is however a vital aspect of almost all application designs.

NFR-04: Likes system to feed into the recommendation system. The category / type of liked items will be more likely to be recommended.

A ‘like’ system would provide a lot of opportunity for not only an improved AI agent and search system but also for other functionality within the application. The database is already prepared for this functionality however time restrictions and delays caused development to be put on hold.

NFR-05: Recommendations based on the contents of individual days i.e the user has a restaurant on days 1 to 3 of the itinerary, but none on day 4, recommend a restaurant for day 4.

Currently the application does not utilise the dates of activities and itineraries to the extent at which it should, taking the date into consideration on many aspects of this application would dramatically improve the basic functionality.

NFR-06: The chatbot's output will be read aloud in text-to-speech.

This had always been the original idea however due to delays in the development of the chatbot, not enough time was left to implement this functionality.

NFR-07: Allow manual customization of itinerary, including dates and times of activities on the itinerary page.

Currently the user is unable to manually customise the itinerary to create a route plan, this would be a nice improvement for usability of the application.

NFR-08: Improved GUI frontend design.

Due to time constraints and delays in development the frontend design was forced to be put on a complete halt, the design of an application is vital to its usability and the users satisfaction.

NFR-09: Enable a 'lock-in' system on activities for route planning e.g. restaurant booking is at 1pm, therefore plan route and itinerary around this.

Often activities require a booking beforehand, therefore it would be a good improvement to include an activity locking system during the best route phase in which the route can be planned around certain time locked activities. This would also then allow lodging and food activities to be at the set time they are required.

Software Design

Modules and dependencies

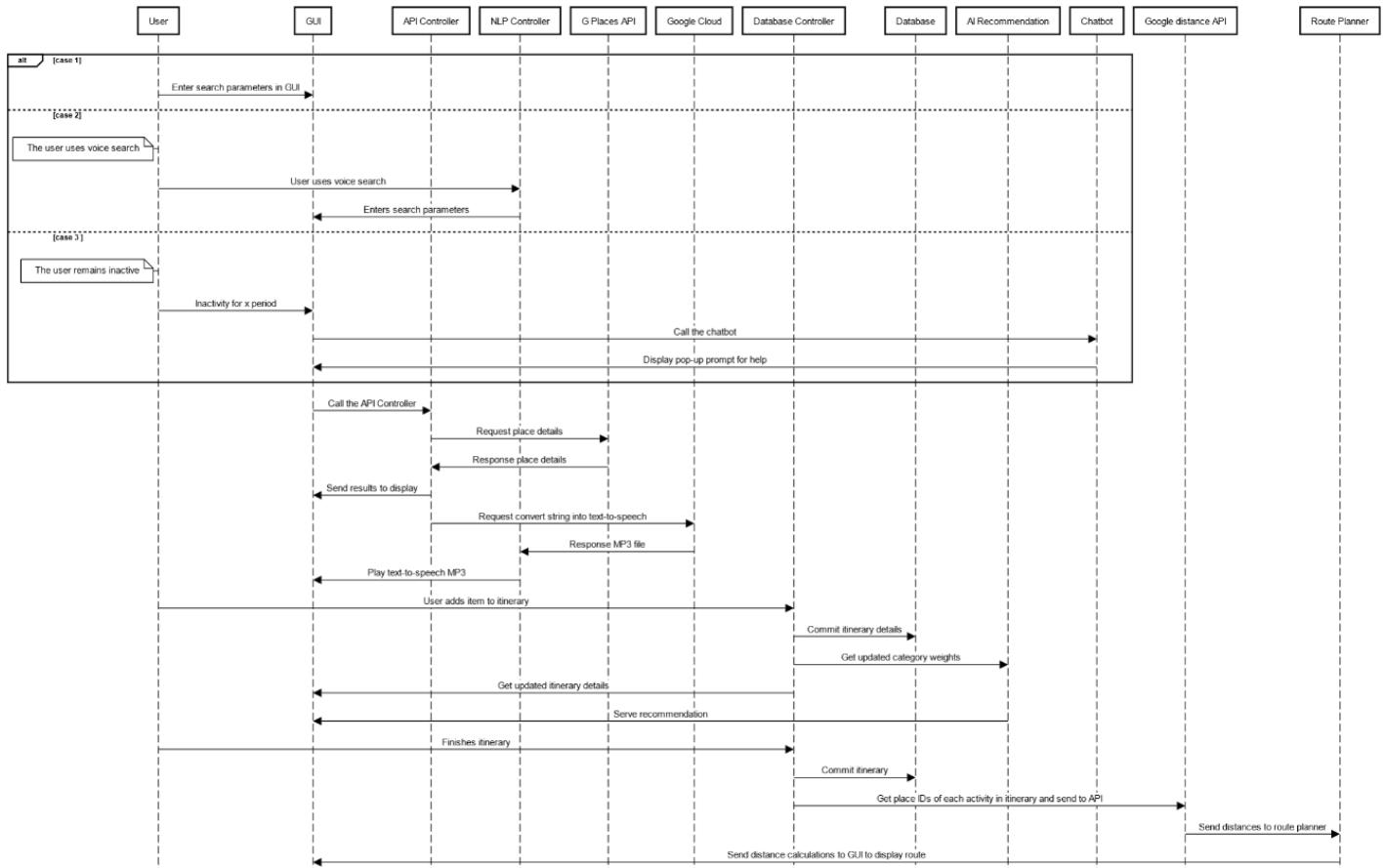


Figure 1

To view in a better way: [Link](#)

The process of searching

The first box with alt in the top left is an if-else representing the three main ways the user can interact with the application. The first case is the 'standard' way of interacting with the application, entering search details via keyboard. The second method is utilising voice search and so the NLP controller is required to communicate with the Google Cloud Speech-to-Text API. The third possibility is a period of inactivity, which will prompt a pop-up in the GUI by the chatbot asking if the user needs help.

Once the user searches for itinerary activities the API controller will be called and will request place information from the Google Places API based on the search parameters the user entered, when the API's response is received the data will be sent to the GUI to display. Shortly after the response is received the API controller will send a string to the Google Text-to-Speech API and will receive an MP3 response to play, the response will read "There are [results_returned] in [location] for [activity] at [date] at [time]"

Adding an item to Itinerary

When the user clicks the add to itinerary button on the GUI the database controller will commit the activity details to the database, and add 1 to the weight value of the user activities category in the AI database. These updated category weights will be sent to the AI recommendation system to tailor recommendations.

Finishing the itinerary and route planning

When the user finishes an itinerary the details of the itinerary will be committed to the database by the controller. The controller will then get the place ID of each activity in the finished itinerary, and send it to the Google distance API to receive every possible distance between each location, which is then sent to the route planner to perform the necessary calculations. Once calculations are complete the end result is sent to the GUI to display the route plan.

Communication between nodes

The communication between nodes can be summarised as below

Node1	Node2	Communication type and channel
Web page	views.py	When the Django project receives a http request, it first checks the urls.py and call the view function. The request will then be processed by the functions defined in the views.py
views.py	ai_learning_controller.py	The <code>home_page()</code> function in the view.py calls the <code>make_recommendation</code> function in the <code>ai_learning_controller.py</code> to make the recommendations for the user.
ai_learning_controller.py	api_controller.py	The <code>make_recommendation</code> function in the <code>ai_learning_controller.py</code> calls the <code>find_recommendation</code> function in the <code>api_controller.py</code> to get the recommended places using google api
views.py	api_controller.py	The <code>home_page()</code> function in the view.py calls the <code>search_handler</code> function in the <code>api_controller.py</code> to get the search results for the user using google api
view.py	api_controller.py	The <code>home_page()</code> function in the view.py calls the <code>location_details</code> function in the <code>api_controller.py</code> to get the details of a particular activity for the user using google api.
views.py	static.NLP.speechtotext.py	The <code>function</code> in the <code>views.py</code> calls the <code>run_speech_recognition</code>

		Function in the speechtotext.py to process the speech input from the user. The user can give speech input to all user input features available in the home page.
Itinerary_object.py	activity.py	All the variables containing the itinerary details are defined in the activity.py. Itinerary_object.py calls the functions in activity.py to pass the variable values and to access them
Itinerary_object.py	user.py	All the variables containing the user details are defined in the user.py
route_planning.py	Itinerary_object.py	The route planning logic receives input to the <code>prepare_data()</code> function from the <code>get_itinerary_all()</code> function in the Itinerary_object.py. The <code>prepare_data()</code> will then calculate the opening time and closing time of the places and activity ID.
route_planning.py	api_controller.py	The route planning logic receives travel time from <code>-get_travel_time</code> function In the api_controller.py file
Chatbot	Web page	Web channel socket (ASGI application) implemented to communicate with http request. Through the websocket chatbot will receive user input and it gives response to that. The routing configuration is defined in the consumers.py and routing.py
All data handling functions	Database	The Database_handler.py file handles all the database access, read ,write and modify functionalities.

Table 2

Overview of Implementation

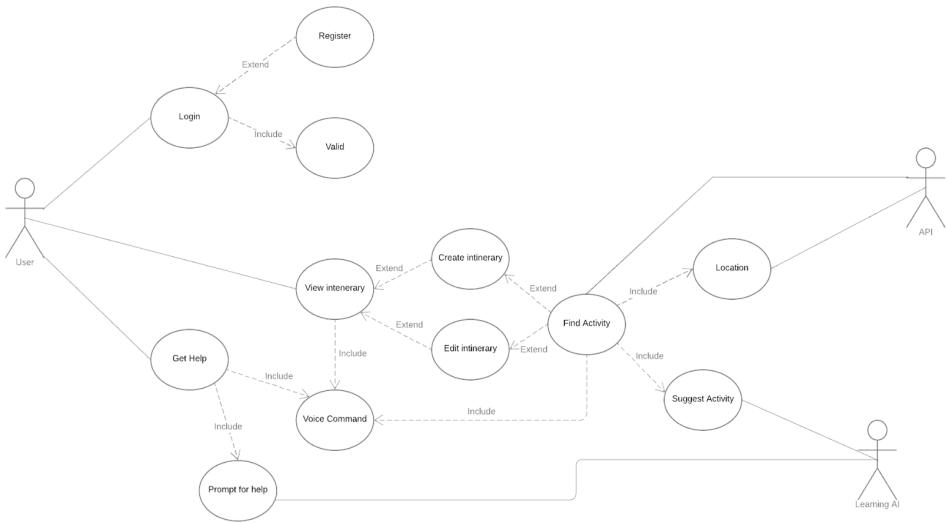


Figure 2

Setup and session start

As per the coursework specification the system can be deployed as a mobile app, a desktop app, or a website, and we opted for the website development. We looked into various cloud-based server and database setup options and decided on an Amazon web services server and a MySQL cloud database at AWS. It was chosen since it provided a one-year free trial. We all are more comfortable with python programming, thus we wanted to use it as the primary language for the website construction. So, we went on the search for a python-based web framework and settled on Django.

Django is a python web development framework that is free and open source. Despite the fact that none of us had any prior expertise with Django, we knew from our research that it would make things easier for us. We wanted to concentrate on the system's main functions rather than website construction, therefore we went with the more straightforward approach. To have a graphical interface with the cloud server and database, we tried tools like Putty, tigervnc, matedesktop, and Mysql workbench. The Mysql database was first verified with SQL commands, and then the implementation of database configuration, read, write and modify were done in Python.

We were also working on the requirement analysis, system architecture, and use cases in the meantime. As soon as the setup was complete, we went on to build the essential functionalities.

Itinerary Creation

The main building blocks of the application consisted of the ability to search for activities in a certain area using an API and then having the capability to create or add an activity to an itinerary. In order to achieve the former task the Google Place API was chosen, with this api it is possible to search for a place by location and activity using the “textsearch” method. A user is then able to add the activity to a previously created itinerary or an existing itinerary, these itineraries could then be customised or deleted on the itinerary page, it's also on this page that the user can activate the best route search.

Itinerary planning tasks

For route planning, we designed an iterative deepening search algorithm that provides the optimum route for the user.

The algorithm considers the minimum amount of time spent at each location, as well as the location's opening and closing times. It will first get a list all the destinations the user wishes to visit, and then create a combination of all possible routes. The route planning logic will then consider a minimum time spent at each location as one hour and create an itinerary based on the condition that the user should visit all of the locations within working hours, i.e., after the opening time and before the closing time. If there are more than one possible route that meet those criteria, the minimum spend time will be extended by one hour, and the itinerary logic will run again, returning new routes. The iteration will continue until we find a single optimal path, and if there are still multiple options, the first route will be chosen as the final route. Also, if there is no possible route covering all the places, the logic will request the user to remove any of the items from the wish list.

With an example, a detailed description of route planning is provided.

Assume the user has chosen to visit three locations: A, B, and C, with S as the starting location.

As stated in the table below, we have the travel time from one location to another as well as the working hours of each location.

Travel time	S	A	B	C
S	0	1	3	1
A	1	0	2	2
B	3	2	0	1
C	1	2	1	0
Working hours		10:00 AM to 18:00 PM	12:00PM to 20:00 PM	14:00 PM to 17:00 PM

Figure 3

The logic will first generate a list of possible routes as below

Itineraries	Travel time in hours	Reaching time at each place (Minimum time spend 2 hour and travel time considered)	Route planning logic
SABC	1+2+1=4	A at 10:00 AM, B at 14:00 PM, C at 17:00 PM	Reject the route, will not reach C on time
SACB	1+2+1=4	A at 10:00 AM, C at 14:00 PM, B at 17:00 PM	Accept
SBAC	3+2+2=7	B at 12:00 PM, A at 16:00 PM, C at 20:00PM	Reject, will not reach C on time
SBCA	3+1+2=6	B at 12:00 PM, C at 15:00 PM, A at 19:PM	Reject, will not reach A on time
SCAB	1+2+2=5	C at 14:00 PM, A at 18:00 PM, B at 22:00 PM	Reject, will not reach B on time
SCBA	1+1+2=4	C at 14:00 PM, B at 17:00 PM, A at 21:00 PM	Reject, will not reach A on time

Figure 4

So, if the minimal spend time is one hour, we have two options.

```
combination of all possible routes
[['S', 'A', 'B', 'C', 4], ['S', 'A', 'C', 'B', 4], ['S', 'B', 'A', 'C', 7], ['S', 'B', 'C', 'A', 6], ['S', 'C', 'A', 'B', 5], ['S', 'C', 'B', 'A',
'A', 4]]
best route when minimum spend time 1
[['S', 'A', 'B', 'C', 4], ['S', 'A', 'C', 'B', 4]]
```

Figure 5

Because we have two options, the logic will run again for a minimum of two hours, and we will have a single optimal route, which will be used as the route planning's final result.

```
combination of all possible routes
[['S', 'A', 'B', 'C', 4], ['S', 'A', 'C', 'B', 4], ['S', 'B', 'A', 'C', 7], ['S', 'B', 'C', 'A', 6], ['S', 'C', 'A', 'B', 5], ['S', 'C', 'B',
'A', 4]]
best route when minimum spend time 2
[['S', 'A', 'C', 'B', 4]]
```

Figure 6

The limitation of this implementation is that the time spent at each location cannot be customised. The implementation couldn't be improved like this due to a time restriction.

Proactive help : Chatbot

The website includes a chatbot that provides proactive assistance to the user. If the user interface is idle for an extended period of time, the system will presume the user requires assistance and the chatbot page will attempt to communicate with the user. The chat bot will provide a website documentation for the user to familiarise themselves with the website.

The chatbot was created using the python chatterbot module. It's simple to integrate with the Django framework. Chatterbot generates responses to user input using a machine learning algorithm. The chatbot library includes built-in functions for training the bot and generating responses to user input. There are choices to specify the storage adapter and logic adapter in the chatterbot library, and we've gone with the default "chatterbot.logic.bestmatch." From the data we used to train the chatbot, it would choose the best match output for the user input.

Django and Chatterbot have a direct integration. To do so, we followed the steps given in the chatterbot instruction. A channel's routing configuration is implemented, which is essentially an ASGI application that informs the channels what code to execute when they get any http request, as well as configured channel's development server. Thus, a chat server is being constructed.

When Django receives a http request, it first checks the urlconf and call the view function. The request will then be processed by the view function. Similarly, the websocket channels we set up take a connection, check the routing configuration, and call the consumer function. After that, the consumer function will handle all of the events. We created a basic consumer that accepts websocket connections and replies to any websocket messages. We created an asynchronous websocket that receives requests from one client and responds to that same client.

The chatbot's front end isn't very appealing due to timing restrictions. We've created a basic user interface for the chatbot, which you can see below.

```

user: hello
Bot : Hi there
user: overview
Bot : please find it here: http://tourist_iternaryplanner/quickstart.html
user: help
Bot : Choose an option for assistance 1.Location 2.Activity 3.Date 4.Search 5.Wishlist
user: Location
Bot : Enter the location you want to visit
user: search
Bot : search for places at that location
user: activity
Bot : Enter what you would like to do at the selected location
user: date
Bot : Set your date of travel
user: wishlist
Bot : add your favourite places to wishlist
user: thanks
Bot : you're welcome

```

Send

Figure 7

In order to provide proactive help the application tracks the amount of time a user is inactive, after a set amount of time the application will initiate a pop up asking if the user requires help. If the user chooses no the pop will not show again until a new page is loaded, if the user selects yes the chatbot will be displayed along with a simple instruction pointing them towards the help page.

AI Recommendation

An AI recommendation system consists of an agent that learns from a user's activity, it is a common mistake to initially make a complicated design plan with various learning aspects. This can easily lead to confusion, mistakes and a lot of bugs causing severe delays, unnecessary stress and in the worst case a complete halt during the development stages. Therefore it was decided to start from the ground up by building something extremely basic that could easily be further developed at a later stage if required.

The first stage involved the creation of an AI database including the tables: “user_categories”, “ref_Categories” and “likes”, “ref_Categories” consists of all the google place api (96 in total) categories for locations with an unique numeric ID assigned. “user _categories” is a table in which the user’s category preferences are stored, each of which have a weighting that increases every time the user adds an activity of that category to an itinerary as shown in figure ___. The “likes” table was intended to be implemented for a favoriting or liking system in future development.

userID	categoryID	score
tempc3ae73a9-365b-4a4f-b289-ed030c58510e	55	0
tempc3ae73a9-365b-4a4f-b289-ed030c58510e	75	0
tempc3ae73a9-365b-4a4f-b289-ed030c58510e	15	0
tempc3ae73a9-365b-4a4f-b289-ed030c58510e	60	0
tempc3ae73a9-365b-4a4f-b289-ed030c58510e	85	0
tempc3ae73a9-365b-4a4f-b289-ed030c58510e	9	0
tempc3ae73a9-365b-4a4f-b289-ed030c58510e	65	0
tempc3ae73a9-365b-4a4f-b289-ed030c58510e	90	0
tempd1cb7d9f-b9cd-41e2-b200-05c60df52241	15	0
tempd1cb7d9f-b9cd-41e2-b200-05c60df52241	60	0
tempd1cb7d9f-b9cd-41e2-b200-05c60df52241	85	0
temp62a0f4f0-68fc-43c2-826b-6c19e8714212	15	0
temp62a0f4f0-68fc-43c2-826b-6c19e8714212	60	0
temp62a0f4f0-68fc-43c2-826b-6c19e8714212	85	0
tempff4a94dd-f654-4f44-8255-80362d90f03	60	0
tempe0c9a39a-8ee0-4e8a-9582-6fb84cd4ala0	60	0
temp8aadb7c0-d63c-4826-8ba1-97a39687d3c2	60	1
temp8aadb7c0-d63c-4826-8ba1-97a39687d3c2	13	2
temp8aadb7c0-d63c-4826-8ba1-97a39687d3c2	65	1
temp7afb2bc5-c9d7-4e19-ac95-f426617alb96	15	0
temp7afb2bc5-c9d7-4e19-ac95-f426617alb96	60	0
temp7afb2bc5-c9d7-4e19-ac95-f426617alb96	85	0
temp7afb2bc5-c9d7-4e19-ac95-f426617alb96	65	0
temp8aadb7c0-d63c-4826-8ba1-97a39687d3c2	90	2
temp8aadb7c0-d63c-4826-8ba1-97a39687d3c2	62	1
temp8aadb7c0-d63c-4826-8ba1-97a39687d3c2	23	0
*	(NULL)	(NULL)

Figure 8

As the user creates more itineraries and adds more activities the weighting of favoured categories increases, the recommendation system will initially search for any currently active itineraries. The agent will then loop through these itineraries to discover that categories that have already been added, these categories will then be removed from the list of options pulled from the database. Finally the agent will take the 5 highest rated categories and select one at random, it will then find the first activity of that category within a set distance from an api search and provide that as a recommendation.

The Agent will also check the itinerary for food related categories, if the user has no activity planned including food then the agent will recommend one. If the itinerary is set to be longer than a day the agent will check for a type of lodging or hotel as well and recommend one if it is not found, this means up to 3 recommendations can be made on an active itinerary.

Future plans of the AI development consisted involving a liking functionality, if the user dislikes a location the will not be shown within a search in future and 1 is removed from the score for that category. If the user liked a location that location would have taken priority over others and been displayed first when searching the same activity in the same location. Ideally this functionality would allow the user to change their personal settings whether to activate it or not. Liked activities and places would also have been displayed on a separate page for the user to browse and customise.

Other special cases

NLP

For Speech-to-Text we used the Google Cloud Speech-to-Text API. This was implemented in the search functionality, after activating the functionality with a button the user could then input the search terms by voice command. The required format for a successful search included naming the search term beforehand and in the set order, an example would look much like “Location... Oxford, Activity...Cinema, Date.... 5th of May 2022, Time... 7pm”. It works by recording audio input from the user’s PC primary audio input device, converting it to a transcript and then using that transcript as the search parameters.

For Text-to-Speech we used the Google Cloud Text-to-Speech API. After a user has entered search parameters for itinerary activities in the GUI and the API response is received the text-to-speech class will create an MP3 file reading the following string to be played: “There are [results_returned] in [location] at [date] at [time]”. It was planned that the output of the help chat bot would be read aloud via text-to-speech but this didn’t come together due to time constraints.

Testing Report

The testing plan was to cover the following key areas:

- Speed testing, to ensure the wait times were reasonable.
- The process of the user searching for itinerary items, to ensure the user isn’t able to break the application with their search inputs.
- The adding of items to the itinerary to ensure the item is properly committed to the database.

The type of testing chosen was black box testing to attempt to view the application from the user’s perspective, to this end we have also conducted a short user experience test with 5 individuals that have never seen the application before.

Application speed testing

The aim of this test is to gauge the amount of time it takes the application to return a response after a user searches for an activity. Two activities were chosen for testing; ‘pub’ because of a large amount of expected responses, and ‘cinema’ for the opposite reason.

Location: Oxford, Activity: Cinema, Date: 31/03/2022, Time: 14:24, Results returned: 4.

Iteration	Time
1	49.22
2	37.30
3	36.55
4	35.44
5	36.21
6	36.32
7	38.60
8	36.76
9	36.57
10	37.08

Table 3

Average response time: 38.005 Seconds

Location: Oxford, Activity: Pub, Date: 31/03/2022, Time: 19:24, Results returned: 13.

Iteration	Time
1	1:23.71
2	1:21.87
3	1:23.11
4	1:23.80
5	1:21.18
6	1:24.53
7	1:21.70
8	1:22.02
9	1:23.65
10	1:27.37

Table 4

Average response time:

These results highlight both the worst case and best case in the response time of the application due to the amount of responses each activity returns.

Validation

The process of searching for itinerary items requires user input, it is therefore important that the validation of this input is tested to ensure that said input does not break the application. The class in charge of this is the validator class, which contains the following functions:

checkSpecialChar(string)

The purpose of this function is to check a string for special characters. The function defines special characters as any characters that aren't numeric or alphabetical. If a special character is detected the function returns True, otherwise it returns False.

Test ID	Test	Input	Expected Output	Actual Output	Changes
01	Testing letters	abc	Return False	Returns False	N/A
02	Testing ints	123	Return False	Returns False	N/A
03	Testing a mix	Sf1l	Return False	Returns False	N/A
04	Testing special chars	!	Return True	Returns True	N/A
05	Testing special chars	@	Return True	Returns True	N/A
06	Testing ints, letters and a special char	w1!	Return True	Returns True	N/A

Table 5

valUsername(string)

The purpose of this function is to validate usernames. To pass validation, a username must not be Null, receive False from the checkSpecialChar function, and be less than or equal to 20 characters in length.

Test ID	Test	Input	Expected Output	Actual Output	Changes

01	Testing empty username.	Null	Return False	Returns None	Fixed incorrect indentation. Test performs as expected post fix.
02	Testing a single letter.	a	Return False	Returns True	Highlighted an oversight, in which the minimum username length wasn't defined, and so a single char would be accepted. Implemented a minimum length of 6. Test performs as expected post fix.
03	Testing a single letter after implementing new length requirement.	a	Return False	Returns False	N/A

Table 6

valPassword(string)

The purpose is to validate passwords. To pass validation, the password must at least 8 characters long, contain at least 1 special character and 1 capital letter.

Test ID	Test	Input	Expected Output	Actual Output	Changes
01	Testing empty passwords.	Null	Return False	Returns False	N/A
02	Testing too short password.	123	Return False	Returns False	N/A
03	Testing password of minimum length.	12345678	Return False	Returns False	N/A
04	Testing password of minimum length with special character.	1234567!	Return False	Returns False	N/A
05	Testing what should be a valid password.	A123456!	Returns True	Returns False	Fixed incorrect indentation. Test performs as expected post fix.

06	Testing a password missing the capital requirement.	a123456!	Returns False	Returns False	N/A
07	Testing a password without the special character requirement.	a1234567	Returns False	Returns False	N/A

Table 7

valEmail(string)

The purpose of this function is to validate email addresses. This is achieved with an API that checks if an email address exists.

Test ID	Test	Input	Expected Output	Actual Output	Changes
01	Null email	Null	Return False	Returns False	N/A
02	Invalid email	rgsdf	Return False	Returns False	N/A
03	Existing email	19177307@brookes.ac.uk	Returns True	Returns True	N/A

Table 8

valAL(string)

This function validates activities and locations that the user enters. It disallows special characters and enforces a length requirement.

Test ID	Test	Input	Expected Output	Actual Output	Changes
01	Null	Null	Return False	Returns False	N/A
02	String that meets length requirement	Yoga	Return True	Returns False	Fixed broken indentation.
03	Special characters	!?	Returns False	Returns False	N/A
04	String that's too long	abcdefghijklmnopqrstuvwxyz p12345	Returns False	Returns False	Works as intended, but decided to increase the length limit.

Table 9

Adding items to the itinerary

Creating an itinerary is the core functionality of the application and so it is important that it is tested. The database handler class handles database operations. The function we tested was:

writelternary(user_id, iter_id, iter_details)

This function is used to create a new itinerary and is called when the user adds the first item to an itinerary.

Test ID	Test	Input	Expected Output	Actual Output	Changes
01	Add an item to an itinerary to cause function call.	Add ODEON cinema to itinerary.	ODEON cinema is added to itinerary.	As expected	N/A.

Table 10

removeActivity(ID)

Remove an activity from an itinerary.

Test ID	Test	Input	Expected Output	Actual Output	Changes
01	Remove activity from itinerary	Remove ODEON cinema from itinerary.	ODEON is no longer on itinerary.	As expected.	N/A

Table 11

User experience testing

To test how the application would feel to new users we gave it to 5 users and asked them to record their experience based on three criteria: ease of use, responsiveness and the help options provided.

User	Ease of use /10	Responsiveness /10	Help provided / 10	Score	Comments
Alice Bennion	5	3	4	13	Slow. Chatbot isn't very useful.
Wendy Coling	6	4	5	15	
Mandy Trowbridge	8	3	4	15	Straightforward but slow.
Cherie Allsworth	4	1	2	7	Slow to search for things.
Craig Coling	8	3	4	15	Slow searching. Help page isn't clear. Chatbot isn't useful.

Table 12

References

[About ChatterBot — ChatterBot 1.0.8 documentation](#)

[Tutorial Part 1: Basic Setup — Channels 3.0.4 documentation](#)

Appendix: Group Meeting Notes

Week1: Meeting 1 Notes

Date & Time:	28/1/2022 (1:00 PM to 1:45 PM)
Goal:	Establish each member's Strengths and weaknesses, decide team leader and initiate the first steps towards development of Itinerary App.
Attendees	Sam Trowbridge, Jayalakshmi Vijayan and Billy Allsworth
Location	University Library

Sam Trowbridge:

Strengths – Experience with API's, Leadership, Reliable, Communication.

Weaknesses – Attention to detail, Python knowledge, Organisation.

Opportunities – Seeking experience in app development.

Threats – Availability.

Jayalakshmi Vijayan:

Strengths – python coding , time management

Weaknesses – lack of front end development skills

Opportunities – coding skills

Threats – communication and interaction

Billy Allsworth:

Strengths – Python

Weaknesses – Organisation

Opportunities – Development experience

Threats – Organisation

Meeting minutes:

- Scheduled weekly meetings.
- Discussed an overview of the topic.
- Google drive and bitbucket prepared.
- Team leader decided (Sam Trowbridge).

Plans for next week:

- Get details on design patterns.

- Ensure understanding of assignment brief.
- Create a lucid chart UML class diagram.
- Set up communication.

Sign Off

Date	28/1/2022	28/1/2022	28/1/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution	33%	33%	33%

Week2: Meeting 1 Notes

Date & Time:	02/02/2022 - 6:00pm - 7:00pm
Agenda:	<ol style="list-style-type: none"> 1. Set up online meeting methods 2. Discuss specifications 3. Work on UML class diagram
Attendees	Sam Trowbridge, Billy Allsworth, Jayalakshmi Vijayan
Location	Online

Achievements since last Meeting:

- Get details on design patterns. : **All**
- Ensure understanding of assignment brief. : **All**
- Create a lucid chart UML class diagram. : **Sam**
- Set up communication. : **Sam**

Points still open since last Meeting:

- UML class diagram

Meeting minutes:

- Set up a group on discord
- Discussed the specifications and decided the primary and secondary functions
- Further developed a UML class diagram

Plans for next week:

- Finish UML diagram
- Research different design patterns

Sign Off

Date	02/02/2022	02/02/2022	02/02/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution	45%	35%	20%

Week2: Meeting 2 Notes

Date & Time:	04/02/2022 - 3:00pm - 3:30pm
Agenda:	<ol style="list-style-type: none"> 1. Finish UML class Diagram 2. Decide design pattern 3. Decide API usage 4. Assign development tasks
Attendees	Sam Trowbridge, Billy Allsworth (Jayalakshmi Vijayan is unavailable due to an emergency)
Location	Online

Achievements since last Meeting:

- Finish UML diagram : All
- Research different design patterns : All

Points still open since last Meeting:

- none

Meeting minutes:

- Using Observer design pattern
- API using google with possible backups
- Finished UML basic design
- Confirmed python is the confirmed language
- Considered local versus cloud database but have decided more research is needed and all members

Plans for next week:

- API start Sam
- Create mock up of interface using photoshop **Sam**
- Look into database options **J**
- Build basic framework from UML **Billy**

Sign Off

Date	04/02/2022	04/02/2022	
Name:	Sam Trowbridge	Billy Allsworth	

Week3: Meeting 1 Notes

Date & Time:	12/02/2022 - 3:00pm - 3:30pm
Agenda:	<ol style="list-style-type: none"> 1. Review current progress made 2. Catch up on recent events 3. Agree mobile app, desktop app or web app 4. Discuss database options and come to a decision (server too possibly) 5. Should we design a use case diagram to ensure the class diagram has everything we need. 6. Assign tasks
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	Online

Side note: due to one member's emergency situation and another catching Covid19, less meetings have taken place and progress has been slow this week. The main aim of this meeting is to discuss new information gained this week and get the group back on track.

Achievements since last Meeting:

- API started **Sam**
- Created mock up of interface using photoshop **Sam**
- Look into database options **J**
- Build basic framework from UML **Billy**

Points still open since last Meeting:

- none

Meeting minutes:

- Caught up on the progress each person has made since last meeting
- Discussed thursdays lesson
- Decided on web based application
- Database will be amazon web services
- Server needs to be further researched, probabili amazon web services
- Discussed use case diagram began developing
- Longer meeting Friday in person
- No progress on API due to illness

Plans for next week:

- Research more on amazon web services (Server): **Jayalakshmi**
- Create design to browser size: **Sam**
- Practice with API: **Sam**
- Create login functionality: **Billy**
- Finish use case diagram: **Everyone**

Sign Off

Date	12/02/2022	12/02/2022	12/02/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution	40%	35%	25%

Week4: Meeting 1 Notes

Date & Time:	18/02/2022 - 12:00pm - 2pm
Agenda:	<ol style="list-style-type: none">1. Review current progress made2. Review data returned from the api, what will we display and how.3. Discuss database storage, what do we store?4. What do we want/need done for next weeks demo.5. Discuss any worries or opinions6. Assign tasks.

	7. Choose some group programming sessions for next week.
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	Online

Side note: due to one member's emergency situation and another catching Covid19, less meetings have taken place and progress has been slow this week also. The main aim of this meeting is to discuss new information gained this week and get the development under way.

Achievements since last Meeting:

- Researched more on amazon web services (Server): **Jayalakshmi**
- Created design to browser size: **Sam**
- Practised with API: **Sam**
- Finish use case diagram: **Sam**

Points still open since last Meeting:

- Create login functionionality:

Meeting minutes:

- Discussed data we will be using from the API
- Agreed favourites is a gd shout
- Database, Server is set up users need to be included
- Class Framework is complete,
- Discussed preposition we would like to be in for demo, homepage to do some basic functions: full login, api demo of displaying result data
- Discussed, django might be too extensive.
- Organised two programming 1 tues 10 thurs
- Create example spreadsheet of stored data

Plans for next week:

- Find more ideal methods of storing, accessing and updating server files and database tables : **Jayalakshmi**
- Create example spreadsheet of stored data: **Jayalakshmi**
- Create login functionionality: : **Billy**
- Further implement API and display data returned : **Sam**

Sign Off

Date	18/02/2022	18/02/2022	18/02/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution	55%	20%	25%

Week5: Meeting 1 Notes

Date & Time:	24/02/2022 - 1:00pm - 1:30pm
Agenda:	<ol style="list-style-type: none">1. Review current progress made2. Discuss demo examples3. Discuss Django4. Discuss next steps5. Discuss any worries or opinions6. Assign tasks.
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	University Campus

Side note: due to another deadline only small tasks are being assigned

Achievements since last Meeting:

- Researched Django : **Jayalakshmi**
- Created demo of a working API : **Sam**

Points still open since last Meeting:

- Find more ideal methods of storing, accessing and updating server files and database tables : **Jayalakshmi**
- Create example spreadsheet of stored data: **Jayalakshmi**
- Create login functionality : **Billy**

Meeting minutes:

- Shown a demo of a working API
- Shown the database and discussed possible required tables, 'clients', 'itinerary' and 'history'
- Django seems to be a good option for python web framework.
- Discussed visual commands and text to speech assistance, decided these are non functional requirements
- Next week we will discuss a testing plan and route planning.

Plans for next week:

- Create a django example: **Jayalakshmi**
- Create html front end: **Sam**
- Continue validation and login: **Billy**

Sign Off

Date	24/02/2022	24/02/2022	24/02/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution:	80%	10%	10%

Week6: Meeting 1 Notes

Date & Time:	1/03/2022 - 1:00pm - 2:00pm
Agenda:	<ol style="list-style-type: none"> 1. Review current progress made 2. Demo of combined Django project 3. Discuss the database tables & columns required 4. Discuss search parameters 5. Discuss route planning options 6. Discuss Help requirements 7. Assign tasks
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan (30m late)
Location	University Campus

Achievements since last Meeting:

- Created a django project and combined all aspects: **Completed by Sam (assigned to Jayalakshmi)**
- Created html front end: **Sam**
- Find more ideal methods of storing, accessing and updating server files and database tables : **Completed by Sam (assigned to Jayalakshmi)**
- Created basic database read and write functions: **Jayalakshmi**
- Further developed validation : **Billy**

Points still open since last Meeting:

- Create example spreadsheet of stored data: **Jayalakshmi**
- Continue validation and login: **Billy**

Meeting minutes:

- Shown a demo of Django project
- Discussed the various tables and data that will be required
- Discussed the search parameters that will be implemented
- Discussed the different help functions that we may need to implement

- Team Decided login functionality was unnecessary

Plans for next week:

- Help commands and Notifications, research and ideas, simple implementation, How can NLPs be used: **Jayalakshmi**
- GUI creation, attempt of search and itinerary creation: **Sam**
- Finish validation, create database tables and read and write functions: **Billy**

Sign Off

Date	01/03/2022	01/03/2022	01/03/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution:	70%	20%	10%

Week7: Meeting 1 Notes

Date & Time:	11/03/2022 - 1:00pm - 1:30pm
Agenda:	<ol style="list-style-type: none"> 1. Review current progress made 2. Quick catch up 3. Discuss Help Chatbot 4. Discuss route planning parameters 5. Assign tasks
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	University Campus

Achievements since last Meeting:

- Researched Help options: **Jayalakshmi**
- GUI creation, successful attempt of search: **Sam**
- Finished validation, database development, created database tables **Billy**
- Database controller: **Sam**
- Create example spreadsheet of stored data: **Completed by Sam (assigned to Jayalakshmi)**

Points still open since last Meeting:

- How can NLP be used: **Jayalakshmi**
- itinerary creation: **Sam**

Meeting minutes:

- Had a quick catch up on any progress made, which was little due to deadlines.
- Discussed the possibility of a chatbot and whether it would be a good solution for our Help requirements
- Route planning will focus primarily on distance

Plans for next week:

- Researched searching agents : **All**
- Create chatbot demo : **Jayalakshmi**
- Add items to itinerary object of user : **Sam**

Sign Off

Date	11/03/2022	11/03/2022	11/03/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution:	55%	30%	15%

Week8: Meeting 1 Notes

Date & Time:	18/03/2022 - 11:30am - 12:45pm
Agenda:	<ol style="list-style-type: none"> 1. Review current progress made 2. Update Team on thursdays practical and the meeting with Pete tuesday 3. Discussed the current progress made 4. What still needs to be achieved 5. Assign tasks
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	University Campus

Achievements since last Meeting:

- Add items to itinerary object of user : **Sam**
- Started report with testing and non-functional: **Billy**

Points still open since last Meeting:

- How can NLP be used: **Jayalakshmi**
- Researched searching agents : **All**
- Create chatbot demo : **Jayalakshmi**

Meeting minutes:

- Decided weekly group work sessions are required to ensure progress is made
- Planned an online meeting for sunday night in order to create a step by step plan
- Had a catchup on the chatbot, discussed whether or not it was taking too long to develop a demo
- Decided that NLP would work perfectly with Chatbot

Plans for next week:

- Review current progress made : **All**
- Researched searching agents : **All**
- Create chatbot demo : **Jayalakshmi**

Sign Off

Date	18/03/2022	18/03/2022	18/03/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution:	40%	40%	10%

Week8: Meeting 2 Notes

Date & Time:	20/03/2022 - 7:00pm - 8:30pm
Agenda:	<ol style="list-style-type: none">1. Review current progress made2. Create a task timetable plan3. Decide on the route planning search agent4. Demo of chatbot5. Check and ensure all tables and columns are created for the database.6. Assign tasks
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	Online

Achievements since last Meeting:

- Review current progress made : **All**
- Researched searching agents : **All**
- Finish chatbot demo : **Jayalakshmi**
- Task list first draft : **Sam**

Points still open since last Meeting:

- none

Meeting minutes:

- Discussed and formed a basic step by step project task list
- Double checked and created all tables and columns for database
- Discuss search agent options and decided iterative deepening to be the best options
- Received demo of chatbot
- Agreed on a group programming session for tuesday.

Plans for next meeting:

- Search agent demo : **Jayalakshmi**
- Convert validation to javascript to allow it to integrate into project : **Billy**
- Fix itinerary bug : **Sam**
- Ensure information required from api is gathered, tidy project : **Sam**

Sign Off

Date	20/03/2022	20/03/2022	20/03/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan

Week9: Meeting 1 Notes

Date & Time:	25/03/2022 - 11:30am - 12:30pm : 1:30pm - 3pm
Agenda:	<ol style="list-style-type: none"> 1. Review current progress made 2. Discuss the AI learning functionality 3. What needs to be done to achieve these functions 4. What will be stored in the database 5. Integrate validation with the django project 6. Assign tasks
Attendees	Sam Trowbridge and Billy Allsworth

	(Jayalakshmi Vijayan is unwell)
Location	University

Achievements since last Meeting:

- Converted validation to Javascript: **Billy**
- Fix itinerary bug : **Sam**
- Ensure information required from api is gathered, tidy project : **Sam**

Points still open since last Meeting:

- Search agent demo : **Jayalakshmi**

Meeting minutes:

- Discussed and made notes on fixes that need to be made to the current implementation:
 - when the itinerary is deleted all activities need to also be deleted
 - in the search function currently the date and time does not affect the search results
 - photos bug, fix it
 - allow users to add a name to their itinerary
 - store History of itin, if the itin is deleted or complete store in history
- Discussed and made following notes on AI learning functionality
 - weighting dictionary stored into the database for activity types
 - recommendations will be a pop up (including basics e.g. hotels & restaurants based on time and how many days)
 - non-functional: keep track of the restaurants that they choose
 - Weighting dictionary for locations? : maybe base this on the currently active itineraries and previously visited locations
 - weighting can either be done using the searches, added activities or completed itin
 - regulate when and how often pop ups appear
 - favoriting system could regulate the api listings (favourites appear at the top, not liked are removed)
 - other users liked this recomm
 - do you wanna see more action, or turn off
- Discussed the possible AI data that will be stored in the database
 - How to organise the categories tables
 - How to set up the like functionality
 - How the search terms can be stored for recommendations

Plans for next meeting:

- Search agent demo : **Jayalakshmi**
- Create new tables, integrate validation in javascript: **Billy**
- NLP demo: **Billy**
- Begin developing the implementation for the AI learning: **Sam**

Sign Off

Date	25/03/2022	25/03/2022	
Name:	Sam Trowbridge	Billy Allsworth	
Contribution:?	40%	40%	20%

Week10: Meeting 1 Notes

Date & Time:	31/03/2022 - 12:00am - 2pm
Agenda:	<ol style="list-style-type: none">1. Search agent demo2. AI learning demo3. Discuss the current progress made4. Go through what we currently have and still require5. Prepare for demo6. Assign tasks
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	University

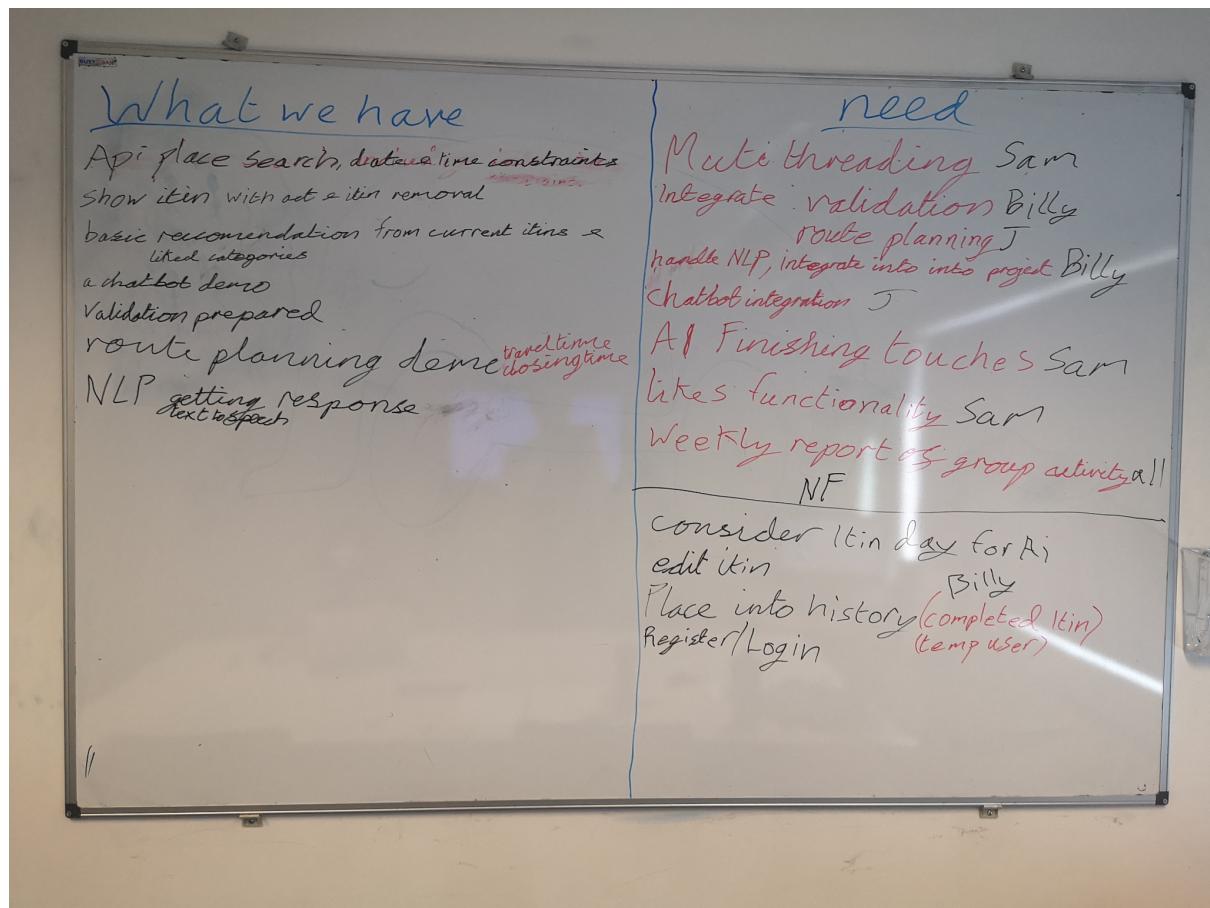
Achievements since last Meeting:

- Search agent demo : **Jayalakshmi**
- Created new tables, populated ai_categories table, added to testing section of the report: **Billy**
- Report test section write up : **Billy**
- Developed basic AI learning: **Sam**

Points still open since last Meeting:

- NLP demo: **Billy**

Meeting minutes:



- Jayalakshmi explained the current demo of the iterative deepening search
- Sam explained the current implementation of the ai learning recommendation system

Plans for next meeting:

- Integrate search agent: **Jayalakshmi**
- integrate validation: **Billy**
- NLP demo: **Billy**
- Further Develop AI functionality: **Sam**

Sign Off

Date	31/03/2022	31/03/2022	31/03/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution:	30%	40%	30%

Week11: Meeting 1 Notes

Date & Time:	08/04/2022 - 12:00am - 1pm
Agenda:	1. Discuss easter holiday and progress to be made
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	University

Achievements since last Meeting:

- Report writing : **Billy**
- NLP demo: **Billy**

Points still open since last Meeting:

- Integrate search agent: **Jayalakshmi**
- integrate validation: **Billy**
- Further Develop AI functionality: **Sam**

Meeting minutes:

- We discussed the important aspects that need to be integrate and completed by the end of easter holidays

Plans for next meeting:

- Integrate search agent: **Jayalakshmi**
- Integrate chatbot: **Jayalakshmi**
- Integrate NLP text to speech once chatbot is integrate: **Billy**
- Develop AI functionality: **Sam**

Sign Off

Date	31/03/2022	31/03/2022	31/03/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution:	10%	80%	10%

Week12: Meeting 2 Notes

Date & Time:	24/03/2022 - 12:00am - 2pm
Agenda:	1. Catchup
Attendees	Sam Trowbridge, Billy Allsworth and

	Jayalakshmi Vijayan
Location	University

Achievements since last Meeting:

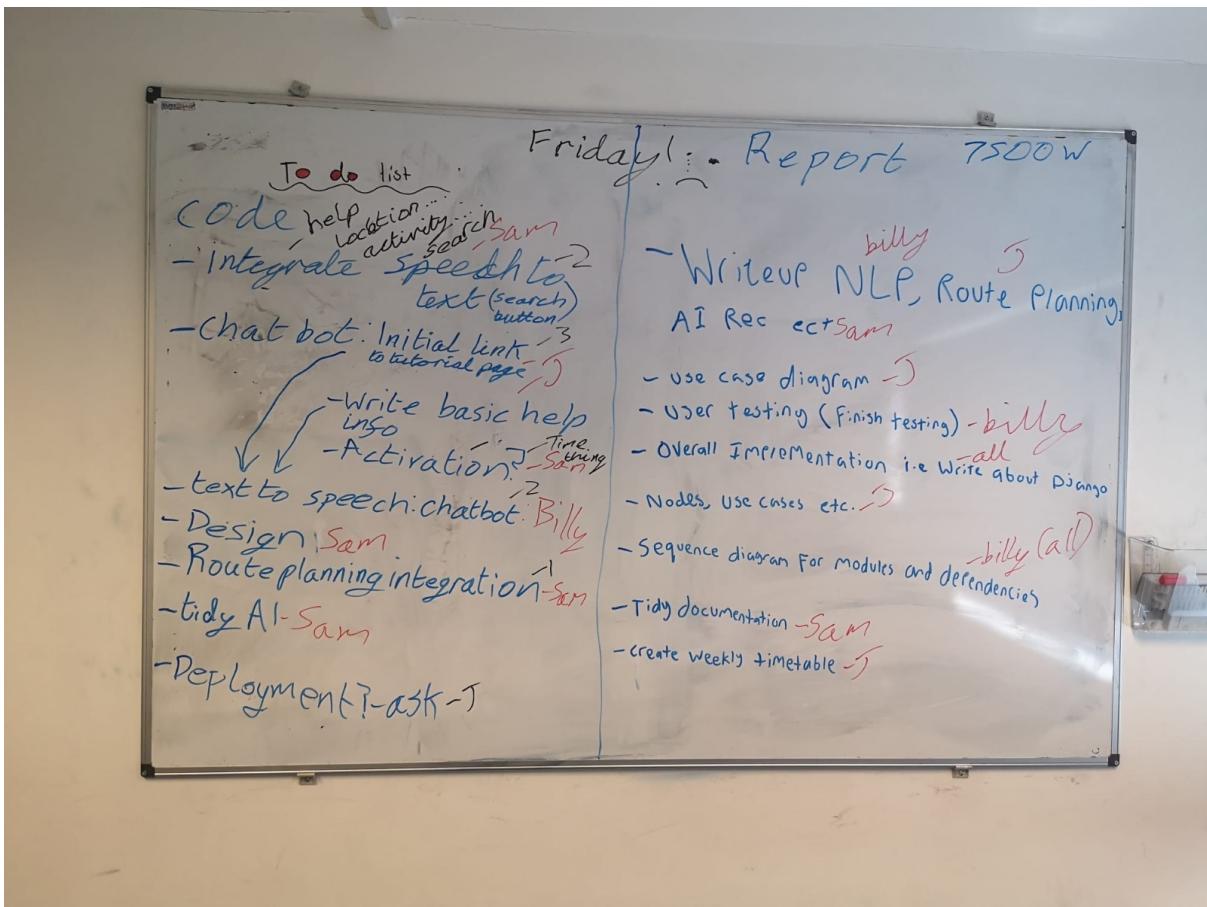
- Some report writing - **Billy**
- Bug fixes: **Sam**

Points still open since last Meeting:

- Integrate search agent: **Jayalakshmi**
- Integrate chatbot: **Jayalakshmi**
- Integrate NLP text to speech once chatbot is integrate: **Billy**
- Develop AI functionality: **Sam**

Meeting minutes:

- Catch up on progress over easter holidays
- Created final task list V



Plans for next meeting:

- Integrate search agent: **Sam**
- Integrate chatbot: **Sam**
- Write help information for chatbot: **Jayalakshmi**
- Write help page: **Jayalakshmi**
- Integrate NLP speech to text: **Sam**

- Integrate NLP text to speech once chatbot is integrate: **Billy**
- tidy AI functionality: **Sam**
- Write report: **All**

Sign Off

Date	31/03/2022	31/03/2022	31/03/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution:	45%	45%	10%

Week12: Meeting 2 Notes

Date & Time:	29/03/2022 - 12:00am - 2pm
Agenda:	1. demo
Attendees	Sam Trowbridge, Billy Allsworth and Jayalakshmi Vijayan
Location	University

Achievements since last Meeting:

- Integrate search agent: **Sam**
- Integrate chatbot: **Sam**
- Integrate NLP speech to text: **Sam**
- tidy AI functionality: **Sam**
- Write report: **All**

Points still open since last Meeting:

- Integrate search agent: **Jayalakshmi**
- Integrate chatbot: **Jayalakshmi**
- Integrate NLP text to speech once chatbot is integrate: **Billy**

Meeting minutes:

- Demo
- Regular online meetings during this week

Plans for next meeting:

- Integrate search agent: **Jayalakshmi**
- Integrate chatbot: **Jayalakshmi**

- Integrate NLP text to speech once chatbot is integrate: **Billy**
- Write report: All

Sign Off

Date	31/03/2022	31/03/2022	31/03/2022
Name:	Sam Trowbridge	Billy Allsworth	Jayalakshmi Vijayan
Contribution:	50%	35%	15%