CSCI 6511

Artificial Intelligence Project Part 2 – Project Report

Team 5

Sagar Sheth

Yuling Li

Akhil Dara

Table of Contents

1. Introduction

The project's goal is to create an expert system for the London Underground Metro System using CLIPS. The system will work on Zones 1 and 2 and will work on stations that exist on the following Lines;

- Bakerloo
- Central
- Circle
- District
- Hammersmith & City
- Jubilee
- Metropolitan
- Northern
- Piccadilly
- Victoria
- Waterloo & City

In this part of the project we have designed and developed initial set of templated and facts which would be the building blocks required to build the expert system. The facts were developed based on the data gathered from the following websites.

- https://tfl.gov.uk/maps/track/tube
- https://en.wikipedia.org/wiki/List_of_London_Underground_stations

The following are the initial list of templates designed to develop the expert system.

- Station
- NextStation
- Line
- Attraction
- Fare
- Route
- glbvisited
- menuInfo

2. Templates & Facts

2.1 Station

This template will capture the following information about each station.

- Name, name of the station.
- Line, the lines the station belongs to. Can have multiple values.
- Zone, the zone the station belongs to.
- Interchange, whether the station is an interchange station. Can have values either Yes/No.
- InternalInterchange, whether the station is an Internal Interchange station. Can have values either Yes/No.

Clips deftemplate

```
(deftemplate Station
		(slot name (type SYMBOL))
		(multislot line (type SYMBOL))
		(slot zone (type INTEGER))
		(slot Interchange (type SYMBOL) (allowed-symbols yes no) (default no))
		(slot InternalInterchange (type SYMBOL) (allowed-symbols yes no) (default no))
)
```

Clips deffacts Sample

```
(deffacts Station
		(Station (name Aldgate)(line Metropolitan Circle)(zone 1)(Interchange Yes)(InternalInterchange No))
)
```

2.2 NextStation

This template will capture the following information about each station.

- CurrStation, name of the current station.
- CurrStationLine, the lines the current station belongs to. Can have multiple values.
- NextStation, name of the next station.
- NextStationLine, the lines the next station belongs to. Can have multiple values.

Clips deftemplate

```
(deftemplate NextStation
		(slot CurrStation (type SYMBOL))
		(multislot CurrStationLine (type SYMBOL))
		(slot NextStation (type SYMBOL))
		(multislot NextStationLine (type SYMBOL))
)
```

Clips deffacts Sample for Algate Station

```
(deffacts NextStation
		(NextStation (CurrStation Aldgate)(CurrStationLine Metropolitan Circle)(NextStation TowerHill)(NextStationLine District Circle))
		(NextStation (CurrStation Aldgate)(CurrStationLine Metropolitan Circle)(NextStation LiverpoolStreet)(NextStationLine Metropolitan Central
Circle HammersmithandCity))
)
```

2.3 Line

This template will capture the following information about each Line.

- name, name of the current Line.
- startingStation, the starting station of the Line from a specific direction. Note that this might not be the true starting Location, as we are only considering Zone 1 & 2. Additionally may change based on direction. May have multiple Values.
- endingStation, the ending station of the Line from a specific direction. Note that this might not be the true starting Location, as we are only considering Zone 1 & 2. Additionally may change based on direction. May have multiple Values.
- stationList, this documents the list of stations in this line. Can have multiple values.
- transferBetween, this an attribute which may be used in future parts to help identify the transfer stations on the line. Currently values are null.

Clips deftemplate

```
(deftemplate Line "Description of lines"
		(slot name (type SYMBOL))
		(multislot startingStation (type SYMBOL))
		(multislot endingStation (type SYMBOL))
		(multislot stationList (type SYMBOL))
		(multislot transferBetween (type SYMBOL) (default nil))
)
```

Clips deffacts Sample for WaterlooandCity Line

```
(deffacts Line
		(Line (name WaterlooandCity)(startingStation Waterloo)(endingStation Bank)(stationList Waterloo Bank)(transferBetween ))
)
```

2.4 Attraction

This template will capture the following information about each Attraction.

- name, name of the Attraction.
- nearbyStations, will state the nearby station for the Attraction. If An attraction can be reached by multiple stations, there will be a separate facts for the same.
- Description, gives a short description for the attracton.

Clips deftemplate

```
(deftemplate Attraction "the attractions in London area"
   (slot name (type SYMBOL))
   (multislot nearbyStations (type SYMBOL))
   (slot description (type STRING))
)
```

Clips deffacts Sample for RoyalAlbertHall

```
(deffacts Attraction
                                  (Attraction (name BritishMuseum)(nearbyStations RussellSquare)(description "A vast treasure trove of global art and artifacts,
including the Rosetta Stone."))
)
```

2.5 Fare

This template will capture the following information about travel fare.

- StartStationZone, The Zone of the station the user starts the travel in.
- DestinationStationZone, The Zone of the station the user will end their trip.
- amount, the fare for travel between Zones. Has 4 enteries.

Clips deftemplate

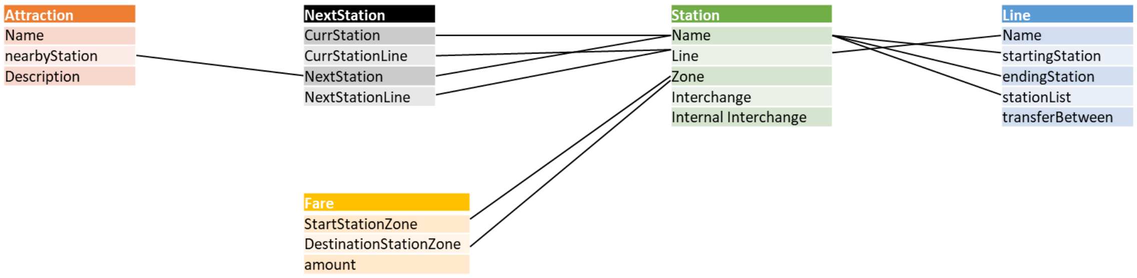
```
(deftemplate fare
        (slot StartStationZone (type INTEGER))
        (slot DestinationStationZone (type INTEGER))
        (slot amount (type FLOAT))
)
```

Clips deffacts

```
(deffacts fare
        (fare (StartStationZone 1)(DestinationStationZone 1)(amount 2.4))
        (fare (StartStationZone 2)(DestinationStationZone 2)(amount 2.4))
        (fare (StartStationZone 1)(DestinationStationZone 2)(amount 2.9))
        (fare (StartStationZone 2)(DestinationStationZone 1)(amount 2.9))
)
```

3. Relationship Between the templates

Below Image shows the relationship between the templates.



Kindly note If an attribute a is linked with an attribute b in another template, and that other attribute b is linked with another attribute c in a third template; Attribute a and c are directly linked but are not shown as part of the image to make the image more readable.

4. Additional Templates

4.1 Route

This template will capture the following information about the route.

- stations, will store the list of stations in sequential order of the path taken.
- currStn, temp variable which will store the current station our route is on.
- visited, will store the list of stations visited.
- count, will store the count of stations.

Clips deftemplate

```
(deftemplate Route
        (multislot stations (type SYMBOL))
        (slot currStn (type SYMBOL))
        (multislot visited (type SYMBOL))
        (slot count (type INTEGER))
)
```

Clips deffacts

As this is a fact created at run time there will be no preloaded facts into the system.

4.2 glbvisited

This template will capture the following information about the all the routes generated by the system.

- vst, will store the list of stations visited by all the routes generated by the system.

Clips deftemplate

```
(deftemplate glbvisited
		(multislot vst (type SYMBOL))
)
```

Clips deffacts

This is a fact created at run time, by default we will store start as the first Value.

4.3 menuInfo

This template will capture the following information.

- mInfo, will store the option the end user wants.

Clips deftemplate

```
(deftemplate menuInfo
   (slot mInfo (type STRING))
)
```

Clips deffacts

This is a fact that is modified at run time, by default we will store Main menu as the first Value.

```
(deffacts menuInfo
   (menuInfo( mInfo "Main Menu"))
)
```

5. Rules

5.1 menu

This rule will generate the Menu required for the Underground Metro Chatbot. It will have the following options.

- Station Info: Information Regarding Station.
- Fare Info: Fare information.
- Attraction Info: Information Regarding Attractions.
- Route Info: Based on start and end station what route should be taken
- Attraction Route: Based on a start station what route should be taken to reach the Attraction

Users can input any of the following values to access the different parts of the menu.

- Station Info
- Fare info
- Attraction Info
- Route info
- Attraction Route

Clips defrule

```
(defrule menu
  ?v <- (menuInfo (mInfo ?mInfo))
  (test (eq ?mInfo "Main Menu"))
  =>
  (printout t crlf)
  (printout t "Hello I am chatbot created to help you travel the London Ungerground Metro System" crlf)
        (printout t "I am currently limited to stations within Zone 1 and 2" crlf)

  (printout t "Kindly select how I can assist you today by selected one of the below options" crlf)
  (printout t "Information" crlf)
  (printout t "1. Station Info :Information Regarding Station." crlf)
  (printout t "2. Fare Info :Fare information." crlf)
  (printout t "3. Attraction Info :Information Regarding Attractions." crlf)

  (printout t crlf)

  (printout t "Point to Point Travel Information" crlf)
  (printout t "4. Route Info :Based on start and end station what route should be taken" crlf)
  (printout t "5. Attraction Route :Based on a start station what route should be taken to reach the Attraction" crlf)

  (printout t crlf)

  (printout t "Exit: Type anything execept the above options to exit this chatbot" crlf)
  (modify ?v (mInfo (readline)))
  (printout t crlf)
)
```

5.2 Station info

To retrieve information about the Station we have 4 rules defined

- Info-Station: Will retrieve station name from end user.
- get-station-info-response: Will print out the following information about the station,
  - Station Name
  - Zone
  - Lines the station Exists on
  - Interchange
  - Internal Interchange
- get-station-info-response2: Will print all the next stations of the current station
- get-menu-back: Will return the program back to the main menu and retract the required temporary facts.

Clips defrule

```
(defrule Info-Station
   (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Station Info"))
   =>
   (printout t "Enter Station" crlf)
   (assert (Stn (read)))
)

(defrule get-station-info-response
   ?mi <- (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Station Info"))
            ?st <- (Stn ?Stn)
            (Station (name ?Stn)(line $?lines)(zone ?zone)(Interchange ?Interchange)(InternalInterchange ?InternalInterchange))

   =>
            (printout t "Station Name:" ?Stn crlf)
            (printout t "Zone:" ?zone crlf)
            (foreach ?line $?lines
            (printout t "Station Line:" ?line crlf))
            (printout t "Interchange:" ?Interchange crlf)
            (printout t "Internal Interchange:" ?InternalInterchange crlf)
            (modify ?mi (mInfo "Station Info2"))
)
(defrule get-station-info-response2
   ?mi <- (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Station Info2"))
            ?st <- (Stn ?Stn)
            (NextStation (CurrStation ?Stn)(NextStation ?NextStation))
            =>
            (printout t "Next Station:" ?NextStation crlf)

)

(defrule get-menu-back
   ?mi <- (menuInfo (mInfo ?mInfo))
            ?st <- (Stn ?Stn)
   (test (eq ?mInfo "Station Info2"))
            =>
            (retract ?st)
            (modify ?mi (mInfo "Main Menu"))
)
```

5.3 Fare info

To retrieve information about the Station we have 2 rules defined

- Info-Fare: Will retrieve the Start and End Station Name from the end User
- get-fare-info-response: Will print out the following information about the station, retract the temporary facts and return the program to the main menu.
    - Start Station Zone
    - End Station Zone
    - Fare

Clips defrule

```
(defrule Info-Fare
   (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Fare Info"))
   =>
   (printout t "Enter Start Station" crlf)
   (assert (stStn (read)))
             (printout t "Enter End Station" crlf)
   (assert (endStn (read)))
)

(defrule get-fare-info-response
   ?mi <- (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Fare Info"))
   (Attraction (name ?name) (nearbyStations ?nearbyStations) (description ?description))
   ?st <- (stStn ?stStn)
             ?end<- (endStn ?endStn)
             (Station (name ?stStn) (zone ?startZone))
             (Station (name ?endStn) (zone ?endZone))
             (fare (StartStationZone ?startZone) (DestinationStationZone ?endZone) (amount ?amt))
   =>
             (printout t "Zone of " ?stStn ": " ?startZone crlf)
             (printout t "Zone of " ?endStn ": " ?endZone crlf)
             (printout t "Fare is:" ?amt crlf)

   (retract ?st)
             (retract ?end)

             (modify ?mi (mInfo "Main Menu"))
)
```

5.4 Attraction info

To retrieve information about the Station we have 2 rules defined

- Info-Attraction: Will retrieve name of the Attraction from the end user.
- get-attraction-info-response: Will print out the following information about the station, retract the temporary facts and return the program to the main menu.
    - Attraction Name
    - Description
    - Nearby Station/s

Clips defrule

```
(defrule Info-Attraction
   (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Attraction Info"))
   =>
   (printout t "Enter Attraction anme, you want information about." crlf)
   (assert (attrName (read)))
)

(defrule get-attraction-info-response
   ?mi <- (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Attraction Info"))
   (Attraction (name ?name) (nearbyStations ?nearbyStations) (description ?description))
   ?an <- (attrName ?attrName)
   (test (eq $?name $?attrName))
   =>
   (printout t "Attraction " ?name crlf)
              (printout t ?description crlf)
              (printout t "Neaby Station:" $?nearbyStations crlf)


   (retract ?an)
              (modify ?mi (mInfo "Main Menu"))
)
```

5.5 Route info

To retrieve information about the Station we have 6 rules defined

- Info-Route: Will retrieve the Start and End Stations from the end user.
- start-dfs: Will initialize the route fact with the start station.
- dfs: Will build the best possible route between the start and end station.
- delete-route: Will delete unwanted routes from routes facts.
- print-route: Will print out the following information about the route, Additionally will retract the temporary facts and reset the glbvisited fact.
    - Route taken sequentially.
    - No of stations.
    - Fare.
- get-menu-back2: Will return the user to the main menu and clean all the route facts.

Clips defrule

```
(defrule Info-Route
   (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Route Info"))
   =>
   (printout t "Enter Start Station" crlf)
   (assert (stStn (read)))
                (printout t "Enter End Station" crlf)
   (assert (endStn (read)))

)
(defrule start-dfs
   (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Route Info"))
   ?st <- (stStn ?stStn)
                ?end <- (endStn ?endStn)
                (NextStation (CurrStation ?stStn) (NextStation ?next))
                =>
                (printout t "Route between:" ?stStn  " and " ?endStn crlf)
                (assert (Route (stations ?stStn) (currStn ?stStn)(count 1)))
)

(defrule dfs
                ?route <- (Route (stations $?stations) (currStn ?currStn) (visited $?visited) (count ?count))
                ?end<- (endStn ?endStn)
                ?vt<-(glbvisited (vst $?vst))
   ?mi <- (menuInfo (mInfo ?mInfo))
                (test (eq ?mInfo "Route Info"))
                (NextStation (CurrStation ?CurrStation) (NextStation ?endStn &:(not (member$ ?CurrStation $?visited))))
                (NextStation (CurrStation ?currStn) (NextStation ?next &:(not (member$ ?next $?visited))))
                (test (not (member$ ?next $?vst)))

                (not (eq ?currStn ?endStn))
                =>
                (assert (Route (stations $?stations ?next) (currStn ?next) (visited $?visited ?currStn) (count (+ ?count 1))))
   (modify ?vt (vst $?vst ?next))
)

(defrule delete-route
                ?end <- (endStn ?endStn)
                ?route <- (Route (currStn ?v &:(neq ?v ?endStn)))
                =>
                (retract ?route)
)

(defrule print-route
   ?mi <- (menuInfo (mInfo ?mInfo))
                ?allroute <- (Route (stations $?stations) (currStn ?currStn) (count ?count))
                ?st <- (stStn ?stStn)
                ?end<- (endStn ?endStn)
                ?vt<-(glbvisited (vst $?vst))

                (Station (name ?stStn) (zone ?startZone))
                (Station (name ?endStn) (zone ?endZone))
                (fare (StartStationZone ?startZone) (DestinationStationZone ?endZone) (amount ?amt))
                (test (eq ?currStn ?endStn))
;               (test (eq ?mInfo "Route Info"))
                =>
                (foreach ?station $?stations
                (printout t "->" ?station ))
                (printout t crlf)
                (printout t "No of Stations: " ?count crlf)
                (printout t "Fare: " ?amt crlf)
                (retract ?st)
                (retract ?end)
                (retract ?allroute)
                (modify ?vt (vst Start))
)


(defrule get-menu-back2
   ?mi <- (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Route Info"))
                =>
                (do-for-all-facts ((?f Route)) (retract ?f))
                (modify ?mi (mInfo "Main Menu"))
)
```

5.6 Attraction Route

To retrieve information about the Station we have 6 rules defined

- Info-Attraction-Route: Will retrieve the Start Station and the destination Attraction from the end user.
- get-attraction-route-info-response: Will initialize the route fact with the start station.
- dfs-attr-route: Will build the best possible route between the start and end station.
- delete-route-attr-route: Will delete unwanted routes from routes facts.
- print-route: Will print out the following information about the route, Additionally will retract the temporary facts and reset the glbvisited fact.
  - Route taken sequentially.
  - No of stations.
  - Fare.
- get-menu-back3: Will return the user to the main menu and clean all the route facts.

Clips defrule

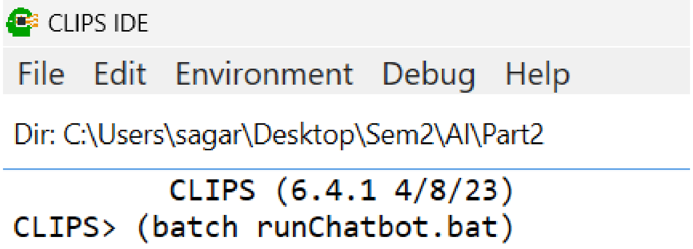
```
(defrule Info-Attraction-Route
   (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Attraction Route"))
   =>
   (printout t "Enter Attraction name, you want travel to" crlf)
   (assert (attrName (read)))
                (printout t "Enter Station you will start the journey from" crlf)
                (assert (stStn (read)))
)

(defrule get-attraction-route-info-response
   ?mi <- (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Attraction Route"))
   ?an <- (attrName ?attrName)
                ?st <- (stStn ?stStn)
   (Attraction (name ?attrName) (nearbyStations ?nearbyStations) (description ?description))
   =>
                (printout t "Route between:" ?stStn  " and " ?attrName ", through " ?nearbyStations crlf)
                (assert (endStn ?nearbyStations))
                (retract ?an)
                (assert (Route (stations ?stStn) (currStn ?stStn)(count 1)))
)

(defrule dfs-attr-route
                ?route <- (Route (stations $?stations) (currStn ?currStn) (visited $?visited) (count ?count))
                ?end<- (endStn ?endStn)
                ?vt<-(glbvisited (vst $?vst))

                (NextStation (CurrStation ?CurrStation) (NextStation ?endStn &:(not (member$ ?CurrStation $?visited))))
                (NextStation (CurrStation ?currStn) (NextStation ?next &:(not (member$ ?next $?visited))))
                (test (not (member$ ?next $?vst)))

                (not (eq ?currStn ?endStn))
                =>
                (assert (Route (stations $?stations ?next) (currStn ?next) (visited $?visited ?currStn) (count (+ ?count 1))))
   (modify ?vt (vst $?vst ?next))
)

(defrule delete-route-attr-route
                ?end <- (endStn ?endStn)
                ?route <- (Route (currStn ?v &:(neq ?v ?endStn)))
                =>
                (retract ?route)
)
(defrule print-route
   ?mi <- (menuInfo (mInfo ?mInfo))
                ?allroute <- (Route (stations $?stations) (currStn ?currStn) (count ?count))
                ?st <- (stStn ?stStn)
                ?end<- (endStn ?endStn)
                ?vt<-(glbvisited (vst $?vst))

                (Station (name ?stStn) (zone ?startZone))
                (Station (name ?endStn) (zone ?endZone))
                (fare (StartStationZone ?startZone) (DestinationStationZone ?endZone) (amount ?amt))
                (test (eq ?currStn ?endStn))
;               (test (eq ?mInfo "Route Info"))
                =>
                (foreach ?station $?stations
                (printout t "->" ?station ))
                (printout t crlf)
                (printout t "No of Stations: " ?count crlf)
                (printout t "Fare: " ?amt crlf)
                (retract ?st)
                (retract ?end)
                (retract ?allroute)
                (modify ?vt (vst Start))
)
(defrule get-menu-back3
   ?mi <- (menuInfo (mInfo ?mInfo))
   (test (eq ?mInfo "Attraction Route"))
                =>

                (do-for-all-facts ((?f Route)) (retract ?f))
                (modify ?mi (mInfo "Main Menu"))
)
```

6. Running the Chatbot

6.1 Menu

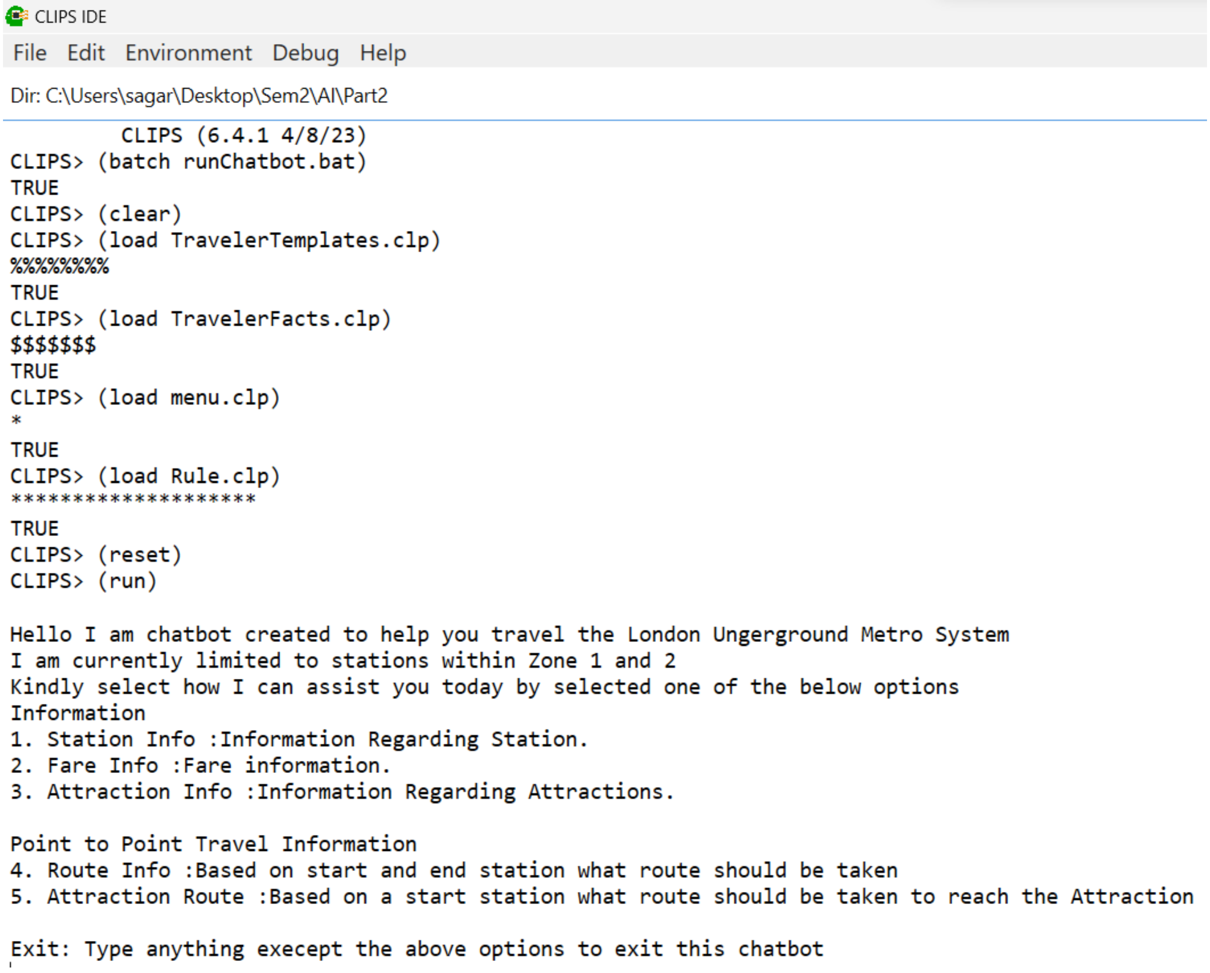
Using the runChatbot.bat file we will load all the rules, templates and facts into clips.

```
(clear)
(load TravelerTemplates.clp)
(load TravelerFacts.clp)
(load menu.clp)
(load Rule.clp)
(reset)
(run)
```



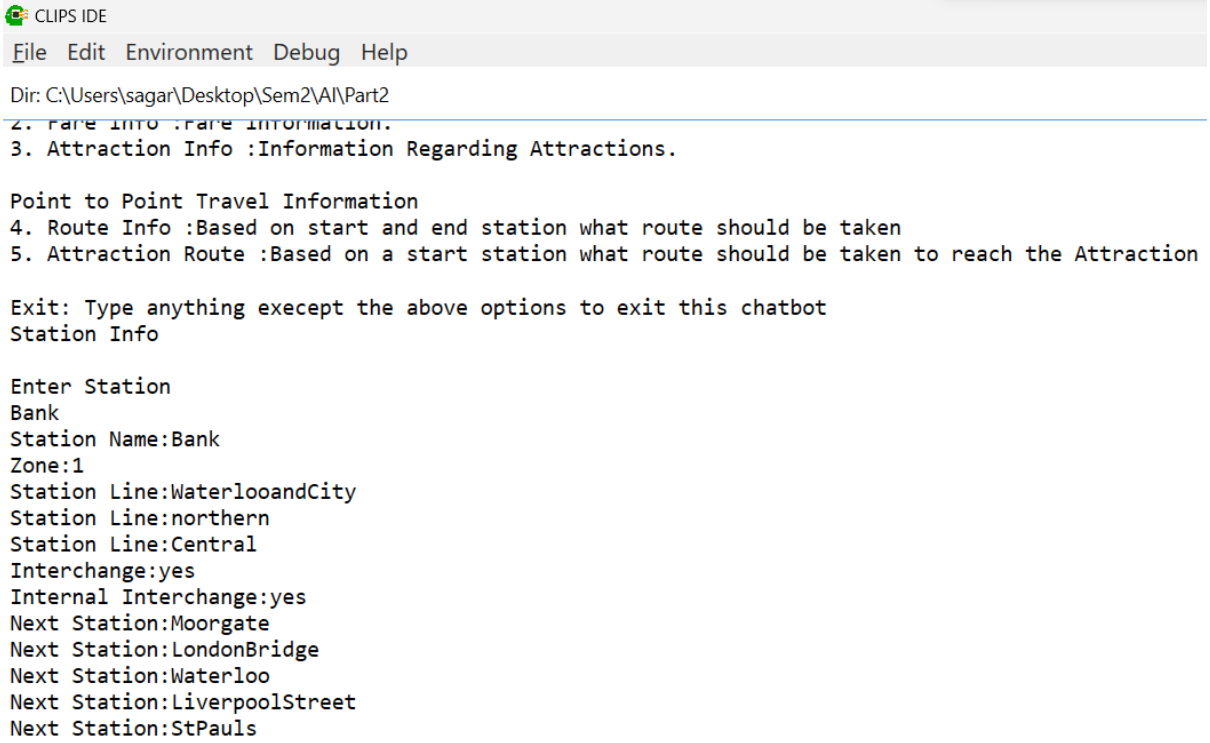
```
Dir: C:\Users\sagar\Desktop\Sem2\AI\Part2
            CLIPS (6.4.1 4/8/23)
CLIPS> (batch runChatbot.bat)
```

Clips Output



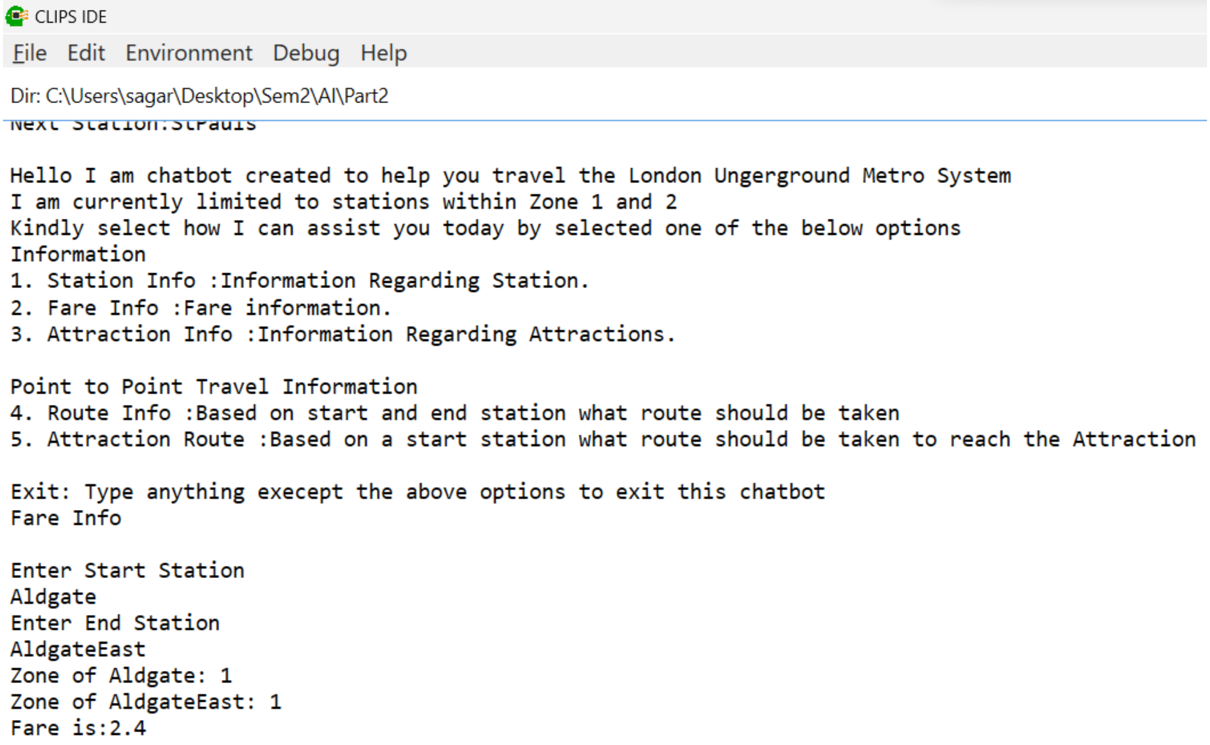
```
Dir: C:\Users\sagar\Desktop\Sem2\AI\Part2
         CLIPS (6.4.1 4/8/23)
CLIPS> (batch runChatbot.bat)
TRUE
CLIPS> (clear)
CLIPS> (load TravelerTemplates.clp)
%%%%%%%
TRUE
CLIPS> (load TravelerFacts.clp)
$$$$$$$
TRUE
CLIPS> (load menu.clp)
*
TRUE
CLIPS> (load Rule.clp)
********************
TRUE
CLIPS> (reset)
CLIPS> (run)

Hello I am chatbot created to help you travel the London Ungerground Metro System
I am currently limited to stations within Zone 1 and 2
Kindly select how I can assist you today by selected one of the below options
Information
1. Station Info :Information Regarding Station.
2. Fare Info :Fare information.
3. Attraction Info :Information Regarding Attractions.

Point to Point Travel Information
4. Route Info :Based on start and end station what route should be taken
5. Attraction Route :Based on a start station what route should be taken to reach the Attraction

Exit: Type anything execept the above options to exit this chatbot
```

6.2 Station Info


CLIPS IDE
File Edit Environment Debug Help
Dir: C:\Users\sagar\Desktop\Sem2\AI\Part2

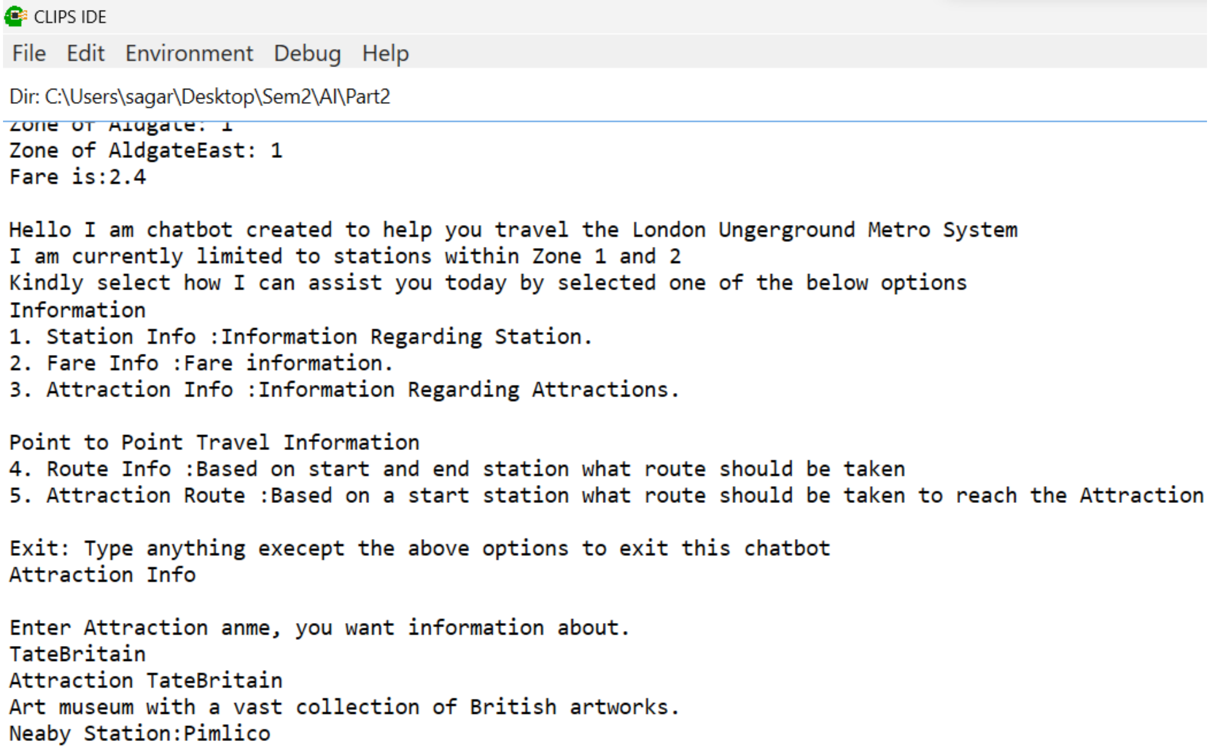
```
2. Fare Info :Fare Information.
3. Attraction Info :Information Regarding Attractions.

Point to Point Travel Information
4. Route Info :Based on start and end station what route should be taken
5. Attraction Route :Based on a start station what route should be taken to reach the Attraction

Exit: Type anything execept the above options to exit this chatbot
Station Info

Enter Station
Bank
Station Name:Bank
Zone:1
Station Line:WaterlooandCity
Station Line:northern
Station Line:Central
Interchange:yes
Internal Interchange:yes
Next Station:Moorgate
Next Station:LondonBridge
Next Station:Waterloo
Next Station:LiverpoolStreet
Next Station:StPauls
```

6.3 Fare Info


CLIPS IDE
File Edit Environment Debug Help
Dir: C:\Users\sagar\Desktop\Sem2\AI\Part2

```
Next Station:StPauls

Hello I am chatbot created to help you travel the London Ungerground Metro System
I am currently limited to stations within Zone 1 and 2
Kindly select how I can assist you today by selected one of the below options
Information
1. Station Info :Information Regarding Station.
2. Fare Info :Fare information.
3. Attraction Info :Information Regarding Attractions.

Point to Point Travel Information
4. Route Info :Based on start and end station what route should be taken
5. Attraction Route :Based on a start station what route should be taken to reach the Attraction

Exit: Type anything execept the above options to exit this chatbot
Fare Info

Enter Start Station
Aldgate
Enter End Station
AldgateEast
Zone of Aldgate: 1
Zone of AldgateEast: 1
Fare is:2.4
```

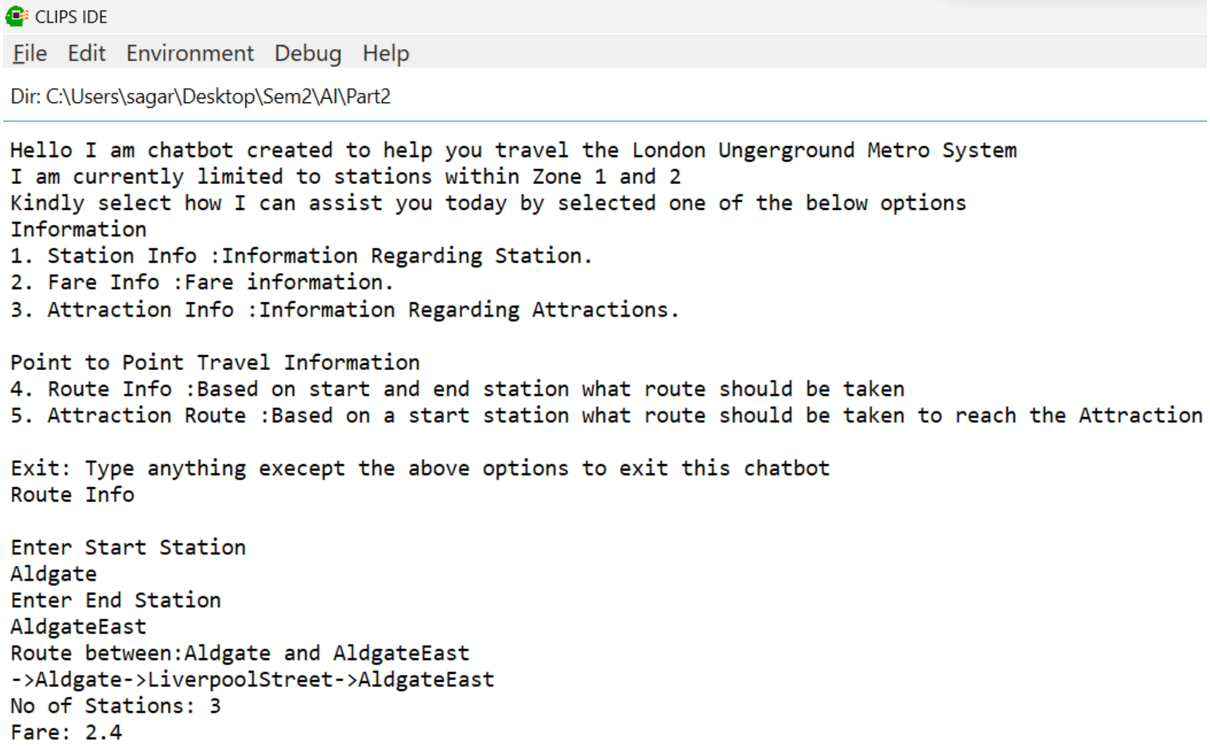
6.4 Attraction Info



Dir: C:\Users\sagar\Desktop\Sem2\AI\Part2

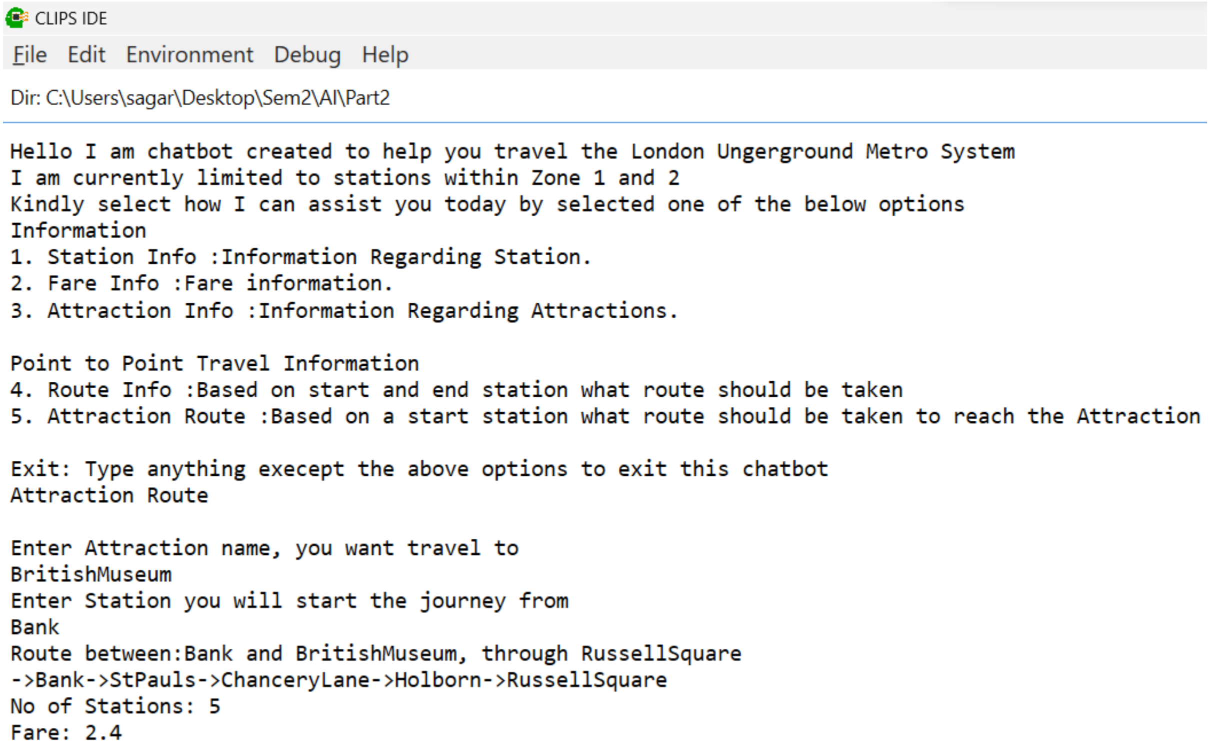
```
Zone of Aldgate: 1
Zone of AldgateEast: 1
Fare is:2.4

Hello I am chatbot created to help you travel the London Ungerground Metro System
I am currently limited to stations within Zone 1 and 2
Kindly select how I can assist you today by selected one of the below options
Information
1. Station Info :Information Regarding Station.
2. Fare Info :Fare information.
3. Attraction Info :Information Regarding Attractions.

Point to Point Travel Information
4. Route Info :Based on start and end station what route should be taken
5. Attraction Route :Based on a start station what route should be taken to reach the Attraction

Exit: Type anything execept the above options to exit this chatbot
Attraction Info

Enter Attraction anme, you want information about.
TateBritain
Attraction TateBritain
Art museum with a vast collection of British artworks.
Neaby Station:Pimlico
```

6.5 Route Info


CLIPS IDE
File Edit Environment Debug Help
Dir: C:\Users\sagar\Desktop\Sem2\AI\Part2

```
Hello I am chatbot created to help you travel the London Ungerground Metro System
I am currently limited to stations within Zone 1 and 2
Kindly select how I can assist you today by selected one of the below options
Information
1. Station Info :Information Regarding Station.
2. Fare Info :Fare information.
3. Attraction Info :Information Regarding Attractions.

Point to Point Travel Information
4. Route Info :Based on start and end station what route should be taken
5. Attraction Route :Based on a start station what route should be taken to reach the Attraction

Exit: Type anything execept the above options to exit this chatbot
Route Info

Enter Start Station
Aldgate
Enter End Station
AldgateEast
Route between:Aldgate and AldgateEast
->Aldgate->LiverpoolStreet->AldgateEast
No of Stations: 3
Fare: 2.4
```

6.6 Attraction Info


CLIPS IDE
File Edit Environment Debug Help
Dir: C:\Users\sagar\Desktop\Sem2\AI\Part2

```
Hello I am chatbot created to help you travel the London Ungerground Metro System
I am currently limited to stations within Zone 1 and 2
Kindly select how I can assist you today by selected one of the below options
Information
1. Station Info :Information Regarding Station.
2. Fare Info :Fare information.
3. Attraction Info :Information Regarding Attractions.

Point to Point Travel Information
4. Route Info :Based on start and end station what route should be taken
5. Attraction Route :Based on a start station what route should be taken to reach the Attraction

Exit: Type anything execept the above options to exit this chatbot
Attraction Route

Enter Attraction name, you want travel to
BritishMuseum
Enter Station you will start the journey from
Bank
Route between:Bank and BritishMuseum, through RussellSquare
->Bank->StPauls->ChanceryLane->Holborn->RussellSquare
No of Stations: 5
Fare: 2.4
```