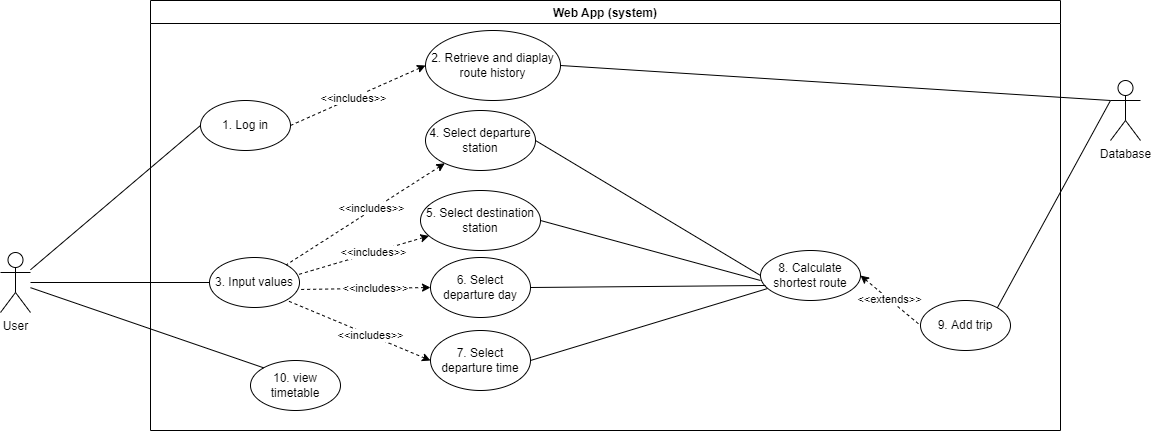
2.2

Use cases:



**Typical course of events**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. User logs in | 1. Retrieve unique route history and suggest trips |
| 1. User enters departure station, destination station, day, and departure time. | 1. Validate stations, day, and time. 2. Calculate and display shortest route. 3. Add trip to database. |

**1. Login –** A user who has signed up has unique details to login. Once they log in the system will find previous trips from the database to make suggestions on stations to travel between.

**Alternate course of events**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| User does not log in. | System behaves the same, except no previous routes (suggestions) are displayed and no shortest route calculations are stored in the database. |
| User enters invalid login details. | System responds with error message and prompts valid login details. |

**3. Input values –** This is when a user chooses specific stations (one for departure and one as a destination) as well as the day they would like to travel and a departure time (optional).

**Alternate course of events**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| User clicks on links to train schedules. | Open pdf files of schedules in new tab. |
| User enters invalid departure station and/or destination station. | System gives error message and prompts for valid stations. |
| User enters invalid day. | System gives error message prompting for a valid day. |
| User enters the same station in both departure and destination fields. | System responds with message saying they are already at the required destination. |
| User enters no time. | System will display all routes between the given stations on the day given. These will be ordered by time, so the quickest trip will be first. |

**7. Calculate shortest route –** The inputted values are then validated and, if all input is valid, they are passed into a function which consists of an algorithm to calculate the shortest route which coincides with the given time. This route is displayed on the web app as a list of trains (including times) the user must take to achieve the fastest trip.

**Alternate course of events**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| Input values have no trains that can achieve this requirement. | System will inform the user of this unfortunate situation. |

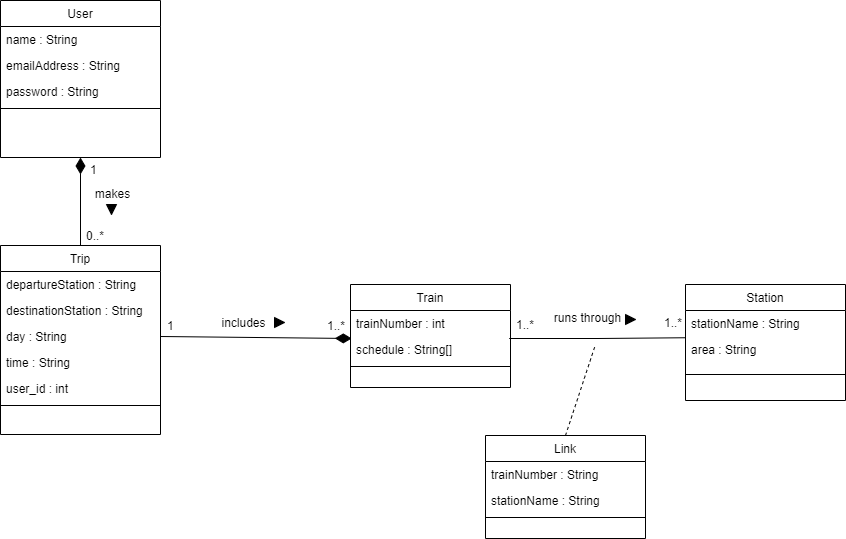
**8. Add trip –** The trip is then added to the database with the users ID, given they have logged in.

**Alternate course of events**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| User is not logged in. | Trip is not added to database. |

**9. View timetable –** The user can also just simply view the times that trains arrive at specific stations via links on the home page.

2.3



In Python there is no existence of private variables so object variables could be accessed from outside the class regardless of whether there were getters or not. Henceforth we felt it not necessary to implement getters and rather just access class variables as “class name”.”variable” wherever it was needed.

The class setup is as follows:

There are trains which run through the same handful of stations everyday of the week. However, some stations have multiple trains which pass by and so we needed a connecting class to distinguish between the trains. Each train also has a schedule which differs on days – Monday to Friday is always the same but Saturday and Sunday differ slightly.

A user is then able to make a trip which consists of a handful of trains they must take to achieve the shortest route which matches this trip.

2.4

Our web UI consisted of a base.html file which included a navigation bar for easy movement between pages for users. The following html files were then extensions of this base file:

* sign\_up.html - This page was used for users who wished to add their details to the database so that their searches would be stored.
* login.html - This was for already registered users to log in and view previous trips they had made.
* home.html - This was the main page where users were able to input values and calculate a shortest route. They could also view train schedules via this page. Logged-in users were also shown previous trips.
* result.html - This was the page that displayed the results of the calculation which was a list of trains to take.

We used a python library called Flask to create our web app as it made linking the html files to the backend programs much simpler. We used another library called Flask-SQLAlchemy which is a toolkit for python and databases – it allowed us to link tables in the database to objects (classes) so that we could run queries on the database for both the algorithm and validation of any input data. Another library called Flask-login allowed us to easily manage whether a user was logged in or not and it stored the users session so they would be remembered when accessing the app again.