1. **Use Cases:**

Diagram

Description automatically generated

* **Login –** A user has unique details to login. Once they log in the system will find previous route history to make suggestions on stations to travel between.
* **Input values –** This is when a user chooses specific stations and times. It includes selecting the departure station and selecting the destination station. The user must also select a departure time and has the option of selecting an arrival time.
* **Calculate routes –** The inputted stations and times are then used to find all routes between the stations.
* **Calculate shortest route –** An algorithm then calculates the shortest of these routes which coincides with the given time(s).
* **Confirm route –** The user must then confirm the available route so that it can be added to their route history for future travelling.
* **View timetable –** The user can also just simply view the times that trains arrive at specific stations.
* **Update train routes –** Admin can update train routes or schedules whenever these change. These updates must reflect in the database afterwards.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. User logs in | 1. Retrieve unique route history and suggest stations |
| 1. User chooses departure station | 1. Verifies station |
| 1. User chooses destination station | 1. Verifies station |
| 1. User chooses departure time (and possibly arrival time) | 1. Finds all routes that fit the given inputs 2. Calculate and return shortest route |
| 1. Confirm shortest path | 1. Add route to history |

**Alternative flows**

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. User chooses invalid departure station. | Returns error of invalid station and prompts valid station (back to step 1) |
| 1. User chooses times which are unrealistic/unachievable | Returns error of unachievable times and prompts user for new times (back to step 6) |
| 1. User chooses invalid destination station | Returns error of invalid station and prompts valid station (back to step 4) |
|  |  |

**2. Analysis Class Diagram**

Graphical user interface, application

Description automatically generated

There are only a few railway lines which all include several stations. Some of these stations may be apart of more than one line – this is where a transfer could occur. Each line also has a few trains which run daily on specific schedules. Users interact with the web interface (system) by inputting the required information, as well as optional additional information. The interface will then calculate the shortest route between two stations and return a list of trains (with times) which achieve this shortest route. Admins can edit the train schedules through the system.

**3. Interaction Diagram**

Diagram

Description automatically generated

1. **Project Plan**
2. **Preliminary Test Plan**

|  |  |
| --- | --- |
| 1. Test case number 1 | A test where the departure is Cape Town station, and the destination is the other end of the same line. Cape Town should be able to go to every station without a stop, therefore testing is to check if the system shows any stops in between, which is not normal. There will be no need to change the train regardless of where you are going. I will be using formulas to determine if the date is on which day. I am expecting train routes that have no stops in between.  Input: str (arrival station), str (departure station), time (wanted arrival time) or time (wanted departure time), date (date)  Output: int (train number), int (platform number), time (actual departure time), time (actual arrival time) |
| Test case number 2 | A test where the departure and arrival train stations are on the same line. I am testing if the system knows that the two stations are on the same line, they don’t necessarily have to be on both ends of the route, they can be in between or right next to each other. Not all clients will start from Cape Town therefore it is important to test if the system works for another part of the area. For the output, I am expecting no stops in between because both arrival and destination train stations are on the same line.  Input: str (arrival station), str (departure station), time (wanted arrival time) or time (wanted departure time) or/and date (date)  Output: int (train number), int (platform number), time (actual departure time), time (actual arrival time), int (number of stops) |
| Test case number 3 | A test where the departure is on one line and the destination is on another line. I am trying to test if the system can pick up that the destination station is not on the same line as the departure station and able to find the best location for the client to get off the train to get onto another train that will take them to the destination. This also tests whether the system can take the time for the client to hop into another train into consideration. I am expecting at least 1 stop as my output, with the platform numbers, train numbers, time of arrival, and time of departure for both trains. The second train’s time of departure must be later than the time of arrival of the first train.  Input: str (arrival station), str (departure station), time (wanted arrival time) or time (wanted departure time) or/and date (date)  Output: int (train number), int (platform number), time (actual departure time), time (actual arrival time), int (number of stops) |
| Test case number 4 | A test where one of the inputs, such as departure time, arrival time, departure station or destination station, and login details is wrong. This test aims to see that if the system can pick up if the client has inputted the wrong information. This is crucial because the client might think they have inputted the right information, but they are wrong. So, if it is the wrong password or wrong username, the system will have to ask the client to redo their username and password or create a new account. If the information regarding the trains is wrong, then the system will tell the user that the format of the information they inputted is wrong. Regarding train information, we are also testing for the autofill part of the train station location names.  Input: str (arrival station), str (departure station), time (wanted arrival time) or time (wanted departure time) or/and date (date)  Output: str (error message) |
| Test case number 5 | The time to depart or arrive is not on the same day.  A test where the departure time is so late that there are no trains that can satisfy the time that the client wants to leave. This is to test whether the system can show the client the other possible trains that will leave either earlier during the day or the trains that will leave the next day.  Input: str (arrival station), str (departure station), time (wanted arrival time) or time (wanted departure time) or/and date (date)  Output: str (message),  Recommend these to the client: int (train number), int (platform number), time (actual departure time), time (actual arrival time), int(number of stops) |