|  |  |
| --- | --- |
| **Name** | **Sign up** |
| **Actors** | Guest |
| **Entry conditions** | The guest is on the log in page of the application and clicks on “Register” button. |
| **Flow of events** | 1. A pop-up shows up asking to the guest if he wants to connect an existing account, such as Google or Facebook, or if he want to create a new account.    1. If the guest connects his account, the system accepts the request and creates a new Travlendar+ account based on provided account.    2. If the guest chooses to create a new account, he is redirected to the registration page which contains all the fields to be filled. 2. The guest fills out all the mandatory fields. 3. (Optional) The guest fills out the preferences. 4. The guest clicks on button “Confirm”. 5. The system checks data provided and eventually creates and registers the user account. |
| **Exit conditions** | The guest has successfully created a new account and he can log into the system with his credentials. |
| **Exceptions** | * Email provided is already in use. The system does not proceed in the registration process and the account is not created. It is possible to repeat the procedure. * Data provided are incorrect. The system highlights the incorrect fields and asks the user to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Log in** |
| **Actors** | User |
| **Entry conditions** | The user launches the application and clicks on the “Log In” button |
| **Flow of events** | 1. The user inserts his email. 2. The user inserts his password. 3. The user clicks on the “Log In” button |
| **Exit conditions** | The login procedure is successfully completed. The user is logged into the system and is able to access to all the functionalities. |
| **Exceptions** | The credentials provided are not associated to any existing account. The login procedure is rejected and the guest is brought back to the login page. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Manage preferences** |
| **Actors** | User, guest |
| **Entry conditions** | The user clicks on the “Preferences” button during the registration process |
| **Flow of events** | 1. User/Guest actives the options that he need. 2. User/Guest clicks on back arrow and return to the previous screen (lateral menu for the user or registration form for the guest). 3. The system receives the user/guest choice and replaces/creates existing/new preferences. |
| **Exit conditions** | Preferences are set. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **View daily schedule** |
| **Actors** | User, External APIs |
| **Entry conditions** | The user clicks on the “View daily schedule” button while checking an appointment on his calendar. |
| **Flow of events** | 1. The system |
| **Exit conditions** | User clicks on back arrow and return to homepage (daily view). |
| **Exceptions** | * Appointments inserted are overlapped. * The travel to reach one appointment is too long to reach it without overlaps with another appointment.   When user views daily schedule, this is computed by exploiting weather condition or the live situation about the street. When the system run into two appointments that are overlapped (first exception) or travel too long (second exception) a message is displayed to the user and commit him to select which is/are the appointment/s to keep. |

|  |  |
| --- | --- |
| **Name** | **View travel details** |
| **Actors** | User |
| **Entry conditions** | 1. The user is on daily schedule page and this is created without errors. 2. User clicks on one travel that are schedule for the day. |
| **Flow of events** | 1. User visualizes all information about the selected travel, such as the vehicle to reach the appointments, the trace route and the travel estimated time. |
| **Exit conditions** | User clicks on back arrow and return to daily schedule. |
| **Exceptions** | No exceptions expected. |

|  |  |
| --- | --- |
| **Name** | **View travel alternatives** |
| **Actors** | User |
| **Entry conditions** | 1. The user is on daily schedule page and this is created without errors. 2. User clicks on one travel that are schedule for the day. |
| **Flow of events** | 1. User clicks on one of available alternatives and changes the travel way. 2. The system receives the user request and re-route the travel on the user choice. 3. The system replaces the old travel screen with new compute travel and shows to the user all details regarding it. |
| **Exit conditions** | The system shows the new travel way on the user choice. |
| **Exceptions** | Only exception provided is the absence of travel alternatives. This is handled with a message that report to the user the lack of alternatives. |

|  |  |
| --- | --- |
| **Name** | **Choose travel alternative** |
| **Actors** | User |
| **Entry conditions** | User have been selected one among the travel alternative and it exists. |
| **Flow of events** | 1. The system shows travel details to the user. 2. User, clicking on save button, overwrites the old travel with the new. 3. Application stays on the same page, thus to allow to the user to visualize travel details and wait a user action. |
| **Exit conditions** | User clicks on back arrow and return to daily schedule. Now the old travel has been overwritten with the new choice. |
| **Exceptions** | The only exception provided is expected during the procedure to saving data. This error can be caused by internet connection absence and it involve that is impossible to update the changes on the account.  The system shows message error and ask to the user to retry to save the changes. |

|  |  |
| --- | --- |
| **Name** | **View movement details** |
| **Actors** | User |
| **Entry conditions** | 1. User is on the page that shows travel details. 2. User clicks on the icon that figure the vehicle related to the movement. |
| **Flow of events** | 1. User visualizes all information about the selected movements, such as the vehicle to reach the appointments, the trace route and the travel estimated time. |
| **Exit conditions** | User clicks on back arrow and return to travel details. |
| **Exceptions** | No exceptions expected. |

|  |  |
| --- | --- |
| **Name** | **View movement alternatives** |
| **Actors** | User |
| **Entry conditions** | 1. User is on the page that shows travel details. 2. User clicks on the icon that figure the vehicle related to the movement. |
| **Flow of events** | 1. User clicks on one of available alternatives and changes the movement way. 2. The system receives the user request and re-route the movement on the user choice. 3. The system replaces the old movement screen with new movement and shows to the user all details regarding it. |
| **Exit conditions** | The system shows the new movement way on the user choice. |
| **Exceptions** | Only exception provided is the absence of movement alternatives. This is handled with a message that report to the user the lack of alternatives. |

|  |  |
| --- | --- |
| **Name** | **Choose movement alternative** |
| **Actors** | User |
| **Entry conditions** | User have been selected one among the movement alternative and it exists. |
| **Flow of events** | 1. The system shows movement details to the user. 2. User, clicking on save button, overwrites the old movement with the new. 3. Application stays on the same page, thus to allow to the user to visualize movement details and wait a user action. |
| **Exit conditions** | User clicks on back arrow and return to daily schedule. Now the old movement has been overwritten with the new choice. |
| **Exceptions** | The only exception provided is expected during the procedure to saving data. This error can be caused by internet connection absence and it involve that is impossible to update the changes on the account.  The system shows message error and ask to the user to retry to save the changes. |

|  |  |
| --- | --- |
| **Name** | **Buy travel ticket** |
| **Actors** | User, Ticket manager |
| **Entry conditions** | User have selected one movement to see it details and clicks on “buy ticket” button. |
| **Flow of events** | 1. The system asks to the user payment information. 2. The system sends the payment information to the ticket manager that complete the operation “buy ticket”. 3. The system shows, on the same page where are contained the movement details, the ticket bought by user. |
| **Exit conditions** | The system shows bought ticket to the user. |
| **Exceptions** | * Not sufficient credit to buy the ticket. * Payment information not valid. * Internet connection error.   All these exceptions are handled with a message error that advise the user to repeat the operation. |

|  |  |
| --- | --- |
| **Name** | **Delete appointment** |
| **Actors** | User |
| **Entry conditions** | The user selects a specific appointment in his calendar (through daily or weekly view) and clicks on “Edit” button. |
| **Flow of events** | 1. A pop-up shows up, asking the user to confirm the deletion. 2. The user confirms the deletion. 3. The system removes all the appointment information from the memory, alert included. |
| **Exit conditions** | The appointment has been deleted and removed from the system. The user is redirected to his calendar page, which has been updated with the removal of the appointment. |
| **Exceptions** | The user does not confirm the deletion. The appointment has not been deleted and the user is redirected to his calendar page. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Create appointment** |
| **Actors** | User |
| **Entry conditions** | The user is checking his calendar and clicks on “Create appointment” button. |
| **Flow of events** | 1. The user is redirected to the appointment creation page, which contains all the fields required to perform the creation. 2. The user fills out all mandatory fields.   The following steps aren’t mandatory:   * 1. User chooses among the available icons.   2. User adds an alert to remember the appointment.   3. User clicks on “More options” and provides more detailed options.  1. User confirms the creation. 2. The system saves the appointment information. |
| **Exit conditions** | The appointment is created and inserted in the system. The user is redirected to his calendar page, which has been updated with the new appointment. |
| **Exceptions** | * The user has provided incorrect information: the appointment is not created and the user must repeat the procedure. * The user clicks on “back” without having confirmed the creation. The appointment is not created and the application returns to the calendar page. It is possible to repeat the procedure. * The appointment overlaps with other previously created appointments. The appointment is created and inserted anyway, but the user receives a notification of the overlapping. |

|  |  |
| --- | --- |
| **Name** | **Edit appointment** |
| **Actors** | User |
| **Entry conditions** | The user selects a specific appointment in his calendar (through daily or weekly view) and clicks on “Edit” button. |
| **Flow of events** | 1. The user is redirected to the appointment editing page which contains all the previously inserted information. 2. The user can perform the following actions:    1. Changing the appointment icon.    2. Editing the information    3. Adding an alert to the appointment.    4. Clicking on “More options” and providing more detailed options. 3. The user saves the changes. 4. The system saves the updated appointment information. |
| **Exit conditions** | The changes are saved and the appointment has been modified. The user is redirected to his calendar page, which has been updated with the new information provided. |
| **Exceptions** | * The user has provided incorrect information: the changes are not saved and the user must repeat the procedure. * The user clicks on “back” without having saved the changes. The appointment has not been modified and the application returns to the calendar page. It is possible to repeat the procedure. * The appointment now overlaps with other previously created appointments. The changes as saved anyway, but the user receives a notification of the overlapping. |

|  |  |
| --- | --- |
| **Name** | **Create flexible appointment** |
| **Actors** | User |
| **Entry conditions** | The user is creating/editing an appointment, clicks on “More options” and clicks on the “Flexible” field. |
| **Flow of events** | 1. A pop-up shows up, containing the fields to be filled. 2. The user fills out the fields, specifying the time range of the appointment. 3. The user confirms the choice. |
| **Exit conditions** | The choice is saved for later and the user is back on the More option page. At the end of the creation process, the appointment will be created at a time compatible with the time interval provided. |
| **Exceptions** | The user clicks on “back” without having confirmed the choice. The appointment has not been modified and the application returns to the More options page. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Create repeatable appointment** |
| **Actors** | User |
| **Entry conditions** | The user is creating/editing an appointment, clicks on “More options” and clicks on the “Flexible” field. |
| **Flow of events** | 1. A pop-up shows up, containing the fields to be filled. 2. The user fills out the fields, specifying the days in which the appointment is wanted to be created. 3. The user confirms the choice. |
| **Exit conditions** | The choice is saved for later and the user is back on the More option page. The appointment is now a repeatable appointment. |
| **Exceptions** | The user clicks on “back” without having confirmed the choice. The appointment has not been modified and the application returns to the More options page. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Create alert** |
| **Actors** | User |
| **Entry conditions** | The user is on the appointment creation/editing page and clicks on the “Add alert” button. |
| **Flow of events** | 1. The user is redirected to the alert creation/editing page. 2. The user fills the field specifying the time of the alert. 3. The user confirms the creation. 4. The system saves the alert information. |
| **Exit conditions** | The alert has been created and inserted in the system. The application returns to the appointment creation/editing page and it is now possible to edit the alert. |
| **Exceptions** | The user clicks on “back” without having confirmed the creation. The alert is not created and the application returns to the event creation page. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Edit alert** |
| **Actors** | User |
| **Entry conditions** | The user is on the appointment creation/editing page and clicks on the “Edit alert” button. |
| **Flow of events** | 1. The user is redirected to the alert creation/editing page. 2. The user replaces needed information with updated ones. 3. The user saves the changes. 4. The system saves the alert information. |
| **Exit conditions** | All the changes have been saved and inserted in the system. The application returns to the appointment creation/editing page and it is possible to repeat the procedure. |
| **Exceptions** | The user clicks on “back” without having saved the changes. The changes have not been saved and the application returns to the appointment creation/modification page. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Delete alert** |
| **Actors** | User |
| **Entry conditions** | The user is on the appointment creation/editing page and clicks on the “Edit alert” button. |
| **Flow of events** | 1. The user is redirected to the alert creation/editing page. 2. The user clicks on “Delete” 3. The user confirms the deletion. 4. The system removes all the alert information from the memory. |
| **Exit conditions** | The alert has been deleted and removed from the system. The application returns to the appointment creation/editing page and it is now possible to add a new alert. |
| **Exceptions** | The user does not confirm the deletion. The alert has not been deleted and the application returns to the alert editing page. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Check appointments on calendar** |
| **Actors** | User |
| **Entry conditions** | The user is logged in and he is on the homepage of the application. |
| **Flow of events** | 1. The system shows an overview of the existing appointments. 2. The user moves between the appointments and collects the needed infomation. |
| **Exit conditions** | The user has collected the needed information and moves to a different page. |
| **Exceptions** | No exception expected. |

|  |  |
| --- | --- |
| **Name** | **Change calendar view** |
| **Actors** | User |
| **Entry conditions** | The user is on the homepage of the application and he is checking his appointments. |
| **Flow of events** | 1. The user opens the lateral menu. 2. The user clicks on one of available view (daily, weekly or monthly). 3. The system changes the layout according to the user’s choice. |
| **Exit conditions** | The layout is changed and the user is back on the calendar view. |
| **Exceptions** | No exceptions expected. |