|  |  |
| --- | --- |
| **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 adjusts the preferences settings. 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 |
| **Entry conditions** | The user clicks on “Preferences” from the side menu on the homepage or during the registration process. |
| **Flow of events** | 1. The system shows the preferences settings, that include transport means owned, favorite kind of travel and other advanced options. 2. The user accesses to the desired settings and adjusts the preferences as wanted. 3. The user saves the changes. |
| **Exit conditions** | The changes are saved and remembered from the system. The preferences have been updated. |
| **Exceptions** | The user clicks on “back” without having confirmed the creation. The new preferences are not saved and the application returns to the homepage. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **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 receives data from different external APIs about routes, traffic, weather and available transport means and computes the best travel option according to the user preferences. 2. The system provides a graphic overview of all the travels scheduled for the day, with the relative movements. 3. The user checks all the information needed and eventually clicks on a specific travel or movement to get further information. |
| **Exit conditions** | User has obtained all the information needed and has clicked “back” to return to the homepage or has selected a specific travel/movement to get further information. |
| **Exceptions** | * Some of the appointments overlap. The system is not able to compute travels between the appointments and forces the user to choose a prior appointment. * Some of the appointments are not reachable in the allotted time. The system doesn’t show the travel and signals the problem to the user with a warning. |

|  |  |
| --- | --- |
| **Name** | **View travel details** |
| **Actors** | User, External APIs |
| **Entry conditions** | The user is checking his daily schedule and clicks on a specific travel. |
| **Flow of events** | 1. The system receives data from different external APIs and collects detailed information about the travel, such as the specific itinerary on the map and the weather conditions. 2. The user is redirected to a new page that displays all the information about the selected travel. 3. The user checks all the information needed and eventually clicks on a specific movement to get further information or buy a ticket. |
| **Exit conditions** | User has obtained all the information needed and has clicked “back” to return to the daily schedule or has selected a specific movement to get further information. |
| **Exceptions** | No exceptions expected. |

|  |  |
| --- | --- |
| **Name** | **Edit travel** |
| **Actors** | User |
| **Entry conditions** | The user is checking a specific travel. |
| **Flow of events** | 1. The user provides modifications to the travel in the following possible ways:    1. Clicking on the travel preferences and switching to a different alternative.    2. Clicking on a specific movement and changing the transport mean. 2. The user saves the changes. |
| **Exit conditions** | The changes are saved and the system displays the new modified travel. |
| **Exceptions** | There are no possible alternatives for the selected travel. The system displays a warning to notify the user. |

|  |  |
| --- | --- |
| **Name** | **View specific movement details** |
| **Actors** | User, External APIs |
| **Entry conditions** | User is checking a specific travel and selects a specific movement. |
| **Flow of events** | 1. The system receives data from different external APIs and collects detailed information about the movement, such as the specific itinerary on the map, the weather conditions and the eventual availability of tickets. 2. The user is redirected to a new page that displays all the information about the selected movement. 3. The user checks all the information needed and eventually proceeds in buying a ticket. |
| **Exit conditions** | User has obtained all the information needed and has clicked “back” to return to the travel page or has proceeded in buying a ticket. |
| **Exceptions** | No exceptions expected. |

|  |  |
| --- | --- |
| **Name** | **Choose prior appointment** |
| **Actors** | User |
| **Entry conditions** | The user is checking the schedule and two appointments overlap. |
| **Flow of events** | 1. The system signals the user the exactly time overlapping of the appointments. 2. The user clicks on the preferred appointment. 3. The system re-computes the daily schedule giving priority to the chosen appointment and shows the updated schedule. |
| **Exit conditions** | No more appointments overlap. The system shows the daily schedule without errors. |
| **Exceptions** | No exceptions expected. |

|  |  |
| --- | --- |
| **Name** | **Buy ticket** |
| **Actors** | User, Ticket provider |
| **Entry conditions** | User is checking a specific movement and clicks on “Buy ticket” |
| **Flow of events** | 1. The system connects trough APIs to the external system of the ticket provider. 2. The system shows the user a form to fill with all the payment information. 3. The user fills out all the fields and confirm the payment. 4. The system sends information to the ticket provider and waits for confirmation and ticket data. |
| **Exit conditions** | Payment is successfully fulfilled and bought tickets are available for the view. The user is brought back to the movement page. |
| **Exceptions** | Payment is rejected. The system notifies the user with a warning. It is possible to repeat the procedure. |

|  |  |
| --- | --- |
| **Name** | **Delete appointment** |
| **Actors** | User |
| **Entry conditions** | The user selects a specific appointment in his calendar (through daily or weekly view) and clicks on “Delete” 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 information. |
| **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. |

|  |  |
| --- | --- |
| **Name** | **View bought tickets** |
| **Actors** | User |
| **Entry conditions** | The user is logged into the system and clicks on “My tickets” button on the side menu. |
| **Flow of events** | 1. The system loads the tickets saved in memory and displays the list to the user. 2. The user chooses a specific ticket and clicks on it. 3. The system displays the full screen ticket. |
| **Exit conditions** | The user has checked the needed tickets and clicks “back” to return to the homepage. |
| **Exceptions** | No tickets saved in memory. The system notifies the absence of saved tickets to the user. |