|  |  |
| --- | --- |
| **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 popup is showed and ask to the guest if he want to connect an existing account, such as Google or Facebook, or if he want to create a new account.    1. If guest select to connect an existing account the system accepts the request and creates a new Travlendar+ account based on selected account and procedure ends.    2. If guest select to create a new account the system initializes the registration procedure. 2. The guest fills all the mandatory fields (first name, last name, email address and password), can insert information to manage a payment account and set the preferences (these two steps aren’t mandatory). 3. The guest clicks on button “Confirm”. 4. The system controls received data and saves it. 5. The guest must confirm registration through a link sent to email address previous inserted. |
| **Exit conditions** | * Guest has selected to connect an existing account to Travlendar+ and the system creates the account. * Guest after has confirmed the registration become new user.   From now he can log in the application ad start to use the system with all functionality. |
| **Exceptions** | * Connection with existing account is not possible. * The guest is already registered. * The information inserted inside mandatory fields are incorrect (ex. Email not valid, Password that not respect all requisites). * Username chosen by guest is already used. * Email inserted by guest is associated to another user.   For the first exception an error message shows to the user that is impossible to connect with the existing account and advices to create it without connecting on existing one.  When the other exceptions occur, an error message appears and displays the reason that caused the failure in the registration procedure. Then the procedure restart from the second steps. |

|  |  |
| --- | --- |
| **Name** | **Log in** |
| **Actors** | User |
| **Entry conditions** | The User is on the log in page of the application. |
| **Flow of events** | 1. The user inserts his credentials (username and password) inside appropriate fields. 2. The user clicks on “Log In” button. 3. The system receives the user’s request and allow him to view his calendar. |
| **Exit conditions** | The user is successfully redirected to view his calendar. |
| **Exceptions** | * The User inserts not valid Username. * The User inserts not valid Password.   All exceptions are handled with an error message that notify to the user the occurred issue and the procedure restarts to the first step. |

|  |  |
| --- | --- |
| **Name** | **Manage preferences** |
| **Actors** | User, guest |
| **Entry conditions** | * The user, with a right slide on the screen, open the menu that contains a preferences button and clicks on it. * The guest, during registration procedure, clicks on preferences button. |
| **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 calendar** |
| **Actors** | User |
| **Entry conditions** | The user must be logged in. |
| **Flow of events** | 1. The user on the homepage of the application can see the state of his calendar. |
| **Exit conditions** | The user controls all his appointments. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **Change view calendar** |
| **Actors** | User |
| **Entry conditions** | The user is on the application homepage. |
| **Flow of events** | 1. The user, with a right slide on the screen, open the menu that contains all possible views. 2. User clicks on one of them. 3. The system receives the request to change the view and closes the menu. |
| **Exit conditions** | * User open the lateral menu but closes it through left slide without change the view. * System has changed the view and has closed the menu. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **View daily schedule** |
| **Actors** | User |
| **Entry conditions** | The user selects daily view, and clicks on “view daily schedule” button. |
| **Flow of events** | 1. User visualizes in sequence all appointments scheduled for the selected day and the relative travel to reach them, with the estimated times. |
| **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. The user visualizes little bar that shows different transports. 2. User, clicking on one of these transports icon, changes the travel way. 3. The system receives the user request and re-route the travel on the user choice. 4. 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 button which have “floppy disk” shape, overwrite 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 into the memory. 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. The user visualizes little bar that shows different transports. 2. User, clicking on one of these transports icon, changes the movement way. 3. The system receives the user request and re-route the movement on the user choice. 4. 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 button which have “floppy disk” shape, overwrite the old travel 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 into the memory. 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 receives the user request and takes the payment information which this it can complete the transition. 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 account 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** | 1. User selects a specific appointment in his calendar (through daily or weekly view). 2. User selects “Delete” command present in an option panel that contains all possible commands. |
| **Flow of events** | 1. The user confirms the deletion. 2. The system receives the request and remove the appointment from the calendar. 3. The system removes also the travel that are associated to deleted appointment. 4. The system removes all alerts that are associated to deleted appointment. |
| **Exit conditions** | The appointment selected by user has been cancelled and the system redirects the user to the previous screen displayed by him before delete the appointment. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **Create appointment** |
| **Actors** | User |
| **Entry conditions** | The user clicks on “Plus” button laid on homepage. |
| **Flow of events** | 1. The system loads a new page that contains all fields required to create new event. 2. The user fills all mandatory fields (name, date, time and location).   The following steps aren’t mandatory:   * 1. User can choose among the available icons.   2. User can add alert related to the new event.   3. User can create a more complex event with “more option”.  1. User clicks on “Save” button. 2. The system controls if the received data are correct. 3. System collects all information inserted by the user and creates a new event, saves it and updates the calendar. |
| **Exit conditions** | * User clicks on “Confirm” button and the event created is added to the calendar and application return to calendar view. * User clicks on back arrow and the procedure to create a new event is interrupted without save. |
| **Exceptions** | The only exception is occurred when user inserts wrong information inside the mandatory fields. It is handled notifying the error to the user and restart the execution from second steps. |

|  |  |
| --- | --- |
| **Name** | **Edit appointment** |
| **Actors** | User |
| **Entry conditions** | 1. User selects a specific appointment in his calendar (through daily or weekly view). 2. User selects “Edit” command present in an option panel that contains all possible commands. |
| **Flow of events** | 1. System receives the request and loads all information related to the selected appointment and these are presented to the user. 2. User edit the information needed. 3. User clicks on “Confirm” button. 4. System check if the information inserted by the user violate any constraints. 5. The system saves the changes and update the calendar. |
| **Exit conditions** | he appointment selected by user has been updated and the system redirects the user to the previous screen displayed by him before edit the appointment. |
| **Exceptions** | The only exception is occurred when user inserts wrong information inside the mandatory fields. It is handled notifying the error to the user and restart the execution from second steps. |

|  |  |
| --- | --- |
| **Name** | **Create flexible appointment** |
| **Actors** | User |
| **Entry conditions** | 1. The user initializes the procedure of event creation. 2. User clicks on “more options” button and a panel that contains all the advanced options is opened. |
| **Flow of events** | 1. User clicks on dropdown menu related to flexible setting. 2. User select the preferred option. 3. User clicks on “Save” button. 4. The system receives the request and saves the changes. |
| **Exit conditions** | * User clicks on back arrow before to save the changes and the system without considering the user changes return to event creation page. * User clicking on “save” button, actives the saving procedure. After this step the system return to event creation page. |
| **Exceptions** | No exception provided. |

|  |  |
| --- | --- |
| **Name** | **Create repeatable appointment** |
| **Actors** | User |
| **Entry conditions** | 1. The user initializes the procedure of event creation. 2. User clicks on “more options” button and a panel that contains all the advanced options is opened. |
| **Flow of events** | 1. User clicks on dropdown menu related to repeatable setting. 2. User select the preferred option. 3. User clicks on “Save” button. 4. The system receives the request and save the changes into the memory. |
| **Exit conditions** | * User clicks on back arrow before to save the changes and the system without considering the user changes return to event creation page. * User clicking on “Save” button, actives the saving procedure. After this step the system return to event creation page. |
| **Exceptions** | No exception provided. |

|  |  |
| --- | --- |
| **Name** | **Create alert** |
| **Actors** | User |
| **Entry conditions** | * User creates a new appointment. * User edit an existing appointment. |
| **Flow of events** | 1. User clicks on “add alert” button. 2. System provide to open new page that contains all fields to create a new alarm. 3. User fills all fields and clicks “Confirm” button. 4. System saves the new alarm. |
| **Exit conditions** | The alarm has been created and the application return to the event creation/modification page, and it is possible to see the created alert. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **Delete alert** |
| **Actors** | User |
| **Entry conditions** | 1. The user selects a specific alarm related to a specific appointment previous selected. 2. User clicks on “Delete” command. |
| **Flow of events** | 1. The user confirms the deletion. 2. System delete the alarm from the memory. |
| **Exit conditions** | The selected alarm has been deleted and the application returns to appointment edit page. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **Modify alert** |
| **Actors** | User |
| **Entry conditions** | 1. The user selects a specific alarm related to a specific appointment previous selected. 2. User clicks on “Edit” command. |
| **Flow of events** | 1. The user edits the alarm settings needed. 2. The user clicks on “Confirm” button. 3. The system receives the request and update the alarm and save all changes. |
| **Exit conditions** | The selected alarm has been updated and the application returns to appointment edit page. |
| **Exceptions** | No exceptions provided. |