|  |  |
| --- | --- |
| **Name** | **Sign up** |
| **Goals** | [G1] |
| **Actors** | Guest |
| **Entry conditions** | The Guest is on the home page of the application. |
| **Flow of events** | 1. Guest clicks on “Register” button to initialize the registration process. 2. The Guest fills all the mandatory fields, that includes personal information and payment account. 3. The Guest clicks on button “Confirm”. 4. The system controls received data and saves it inside appropriate database. 5. The Guest must confirm registration through a link sent to email address previous inserted. |
| **Exit conditions** | The Guest after has confirmed the registration become new User.  From now he/she can log in the application ad start to use the system with all functionality. |
| **Exceptions** | * The Guest is already an User. * The information inserted inside mandatory fields are incorrect (ex. Email not valid, Password that not respect all requisites). * Username chose by Guest is already used. * Email inserted by Guest is associated to another User.   When these exceptions occur (except the first) an error message appears and displays the reason that caused the error in the registration procedure. Then the procedure restart from the second steps. |

|  |  |
| --- | --- |
| **Name** | **Log in** |
| **Goals** | [G2] |
| **Actors** | User |
| **Entry conditions** | The User is on the home page of the application. |
| **Flow of events** | 1. The User inserts his/her 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/her to view his/her calendar. |
| **Exit conditions** | The User is successfully redirected to view his/her 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** | **View calendar** |
| **Goals** | [G3] [G4] [G5] [G6] [G7] |
| **Actors** | User |
| **Entry conditions** | The User must be logged in. |
| **Flow of events** | 1. The User with left swipe on the screen can open a panel that contains three possible calendar views:  * Monthly; * Weekly; * Daily.  1. The User clicks on one among three alternatives. 2. The system changes the view as requested by User. |
| **Exit conditions** | The calendar view has been changed as the User has required. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **View daily schedule** |
| **Goals** | [G7] |
| **Actors** | User |
| **Entry conditions** | The user must be logged in |
| **Flow of events** | 1. User visualizes in sequence trace route and the estimated time of all travel scheduled in that day 2. User can click on one of these travel to view more details 3. User can come back to the previous screen 4. The system acquires input of user and change screen |
| **Exit conditions** | User wanted to visualize the details of one travel or User decided to come back to previous screen |
| **Exceptions** | No exceptions expected. |

|  |  |
| --- | --- |
| **Name** | **View travel alternatives** |
| **Goals** | [G2][G8] |
| **Actors** | User |
| **Entry conditions** | User must be logged in and click on travel to visualize details |
| **Flow of events** | 1. The user click on a transport service different from the one already on selection 2. The system shows possible alternative in sequence 3. The user can click on one of these, is they exists, to visualize more details 4. User can come back to the previous screen with the back button 5. The system acquire user input and change screen |
| **Exit conditions** | The system changes screen due to input of user |
| **Exceptions** | * There are no travel alternatives * There are no details about the travel   When an exception occurs, the system shows an error dialog frame that invites user to back to previous screen |

|  |  |
| --- | --- |
| **Name** | **Choose travel alternative** |
| **Goals** | [G2][G8] |
| **Actors** | User |
| **Entry conditions** | There almost is one choosable travel alternative |
| **Flow of events** | 1. The system shows in sequence the travel alternatives 2. User click on one alternative or skip back 3. The system shows details of all movements travel has 4. The system acquires user input |
| **Exit conditions** | The system changes screen due to input of user |
| **Exceptions** | No exceptions expected. |

|  |  |
| --- | --- |
| **Name** | **Delete appointment** |
| **Goals** | [G4] |
| **Actors** | User |
| **Entry conditions** | The user must select a specific appointment in his calendar. |
| **Flow of events** | 1. The system, after having detected the user’s click on a specific appointment, show an option panel with all possible commands. 2. The user selects “Delete” command. 3. The user confirms the deletion. 4. The system receives the request and remove from server and local memory the appointment. 5. The system removes all alerts that are associated to deleted appointment. |
| **Exit conditions** | The appointment selected by User has been cancelled. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **Create appointment** |
| **Goals** | [G3] |
| **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 fill all mandatory fields and click “Next” button. 3. The system controls if the received data are correct. 4. The system controls if the event overlaps with other events and eventually notify it to the User. 5. A little popup appears and requires position information to the User. 6. User select the desired option. 7. Another popup appears and requires the intention to active an alarm related to the appointment. 8. System collects all information inserted by the user and creates a new event, saves it inside memory and updates the calendar. |
| **Exit conditions** | The event created by the User has been added to the calendar. |
| **Exceptions** | * The information inserted inside mandatory fields are incorrect. * The new event overlaps with other existing events.   The first exception is handled notifying the error to the user, he have to click on ok button and restart the execution from second steps.  Instead, the second exception is handled by user’s will to keep equally the new appointment. |

|  |  |
| --- | --- |
| **Name** | **Modify appointment** |
| **Goals** | [G5] |
| **Actors** | User |
| **Entry conditions** | The user must select a specific appointment in his calendar. |
| **Flow of events** | 1. The system, after having detected the user’s click on a specific appointment, show an option panel with all possible commands. 2. The user selects “Modify” command. 3. System receives the request and loads all information related to the selected appointment and these are presented to the user. 4. User edit the information needed. 5. User click “Confirm” button. 6. System check if the information inserted by the user violate any constraints. 7. System check if the appointment now overlaps with other events. 8. The system saves the changes and update the calendar. |
| **Exit conditions** | The desired appointment has been updated. |
| **Exceptions** | * The information edited now are incorrect. * The new event overlaps with other existing events.   Both exceptions are handled as “Create appointment” case. |

|  |  |
| --- | --- |
| **Name** | **Create alert** |
| **Goals** | [G9] |
| **Actors** | User |
| **Entry conditions** | The user creates new appointment or edits an existing one. |
| **Flow of events** | 1. Different entry points: 2. After that user has created a new appointment, the system asks to the user if it wants to create a new alarm and the user chooses “Yes”. 3. If the user wants to edit an existing appointment, selects the alarm section and click to add another alarm. 4. System provide to open new page that contains all fields to create a new alarm. 5. User fills all fields and clicks “Confirm” button. 6. System saves the new alarm. |
| **Exit conditions** | The alarm has been created and added into the system. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **Delete alert** |
| **Goals** | [G9] |
| **Actors** | User |
| **Entry conditions** | The user must edit a specific alarm related to a specific appointment previous selected. |
| **Flow of events** | 1. The user clicks on a specific alarm. 2. System detects the user’s click and opens a little panel that contains all possible commands. 3. The user selects on “Delete” command. 4. The user confirms the deletion. 5. System delete the alarm from the memory. |
| **Exit conditions** | The selected alarm has been deleted. |
| **Exceptions** | No exceptions provided. |

|  |  |
| --- | --- |
| **Name** | **Edit alert** |
| **Goals** | [G9] |
| **Actors** | User |
| **Entry conditions** | The user must edit a specific alarm related to a specific appointment previous selected. |
| **Flow of events** | 1. The user clicks on a specific alarm. 2. System detects the user’s click and opens a little panel that contains all possible commands. 3. The user selects on “Edit” command. 4. The user edits the alarm needed. 5. The user clicks on “Confirm” button. 6. The system receives the request and update the alarm and save all changes. |
| **Exit conditions** | The selected alarm has been updated. |
| **Exceptions** | No exceptions provided. |