

|  |  |
| --- | --- |
| **Use Case ID** | UC - 01 |
| **Use Case Name** | Register |
| **Use Case**  **Description** | Allows Helia Users that do not have an account to create an account |
| **Participating**  **Actor(s)** | User |
| **Pre-Conditions** | 1. The User must not have an account 2. The User must not be logged in |
| **Post-Conditions** | 1. The User has created an account |
| **Normal Flow** | * 1. The User fills in (sys reply???) his Username, Password, Email, and other personal information   2. The User uploads an image for their avatar   AF-1: The User does not upload any image   * 1. The User press the Register button   2. The System checks if the Username and Email has been taken   AF-2: The Username has been taken  AF-3: The Email has been taken   * 1. The System saves the User information into the database |
| **Alternative Flows** | AF-1: The User does not upload any image   1. The System set the Users Avatar to the default avatar   AF-2: The Username has been taken   1. The System prompts the User that the Username has been taken. 2. The User acknowledges and is taken back to 1.3   AF-3: The Email has been taken   1. The System prompts the User that the Email has been used. 2. The User acknowledge and is taken back to 1.3 |
| **Exceptions** | - |
| **Includes** | - |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 02 |
| **Use Case Name** | Login |
| **Use Case**  **Description** | Allows Helia Users login to their account |
| **Participating**  **Actor(s)** | User |
| **Pre-Conditions** | 1. The User must have an account 2. The User must not be logged in |
| **Post-Conditions** | 1. The User is logged in |
| **Normal Flow** | * 1. The User enters (sys reply???) Their Username and Password   2. The User presses the Login button   3. The System checks（validate？） if the Username and Password match in the database   AF-1: The Username does not exist  AF-2: The Password is incorrect   * 1. The System logs the user in |
| **Alternative Flows** | AF-1: The Username does not exist   1. The System prompts the User that the Username does not exist. 2. The User acknowledge and is taken back to 1.1   AF-2: The Password is incorrect   1. The System prompts the User that the Password is incorrect. 2. The User acknowledge and is taken back to 1.1 |
| **Exceptions** | - |
| **Includes** | - |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 03 |
| **Use Case Name** | Modify Personal Particulars |
| **Use Case**  **Description** | Allows Helia Users to modify their personal particulars and avatar |
| **Participating**  **Actor(s)** | User |
| **Pre-Conditions** | 1. The User is logged in |
| **Post-Conditions** | 1. The User information is saved |
| **Normal Flow** | * 1. The User changes (sys reply???) any details of his particulars   2. The User click on the Upload button to upload a new image as their avatar   AF-1: The User does not upload any new image   * 1. The User presses the Save button   2. The System checks (validate??) for any invalid input   AF-2: There are invalid inputs   * 1. The System save the changes made |
| **Alternative Flows** | AF-1: The User does not upload any new image   1. The System keeps their current avatar   AF-2: There are invalid inputs   1. The System prompts the User of the invalid input 2. The User acknowledge and is taken back to 1.1 |
| **Exceptions** | - |
| **Includes** | - |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 04 |
| **Use Case Name** | Discover Activities |
| **Use Case**  **Description** | Allows Helia Users to discover various activities to promote a healthier lifestyle |
| **Participating**  **Actor(s)** | User, External data sets |
| **Pre-Conditions** | 1. The User is logged in |
| **Post-Conditions** | 1. A list of activities will be displayed |
| **Normal Flow** | * 1. The User clicks on the Discover Activities button   2. The System checks the current weather condition, UV Index, Pollutant Standard Index and generates a list of recommended activities. (How?? Other systems?)   3. The System displays the activities that would be appropriate for the user   4. The User selects preferred activity   5. The User swipes right to like the Activity   AF-1: The User swipes left   * 1. The System store the information that the User likes this activity   AF-2: The User clicks on Venue Recommendation  AF-3: The User clicks on Healthy Eateries Recommendation |
| **Alternative Flows** | AF-1: The User swipes left   1. The System stores the information that the user does not like this activity 2. The System removes the activity from the Users page   AF-2: The User clicks on Venue Recommendation   1. The flow continues in UC-05 Venue Recommendation   AF-3: The User clicks on Healthy Eateries Recommendation   1. The flow continues in UC-06 Healthy Eateries Recommendation |
| **Exceptions** | - |
| **Includes** | - |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 05 |
| **Use Case Name** | Venue Recommendation |
| **Use Case**  **Description** | Allows Helia Users to view recommended venue for the sports activity recommended to them. |
| **Participating**  **Actor(s)** | User, External data sets |
| **Pre-Conditions** | 1. This use case extends the UC-04 Discover Activities. It is initiated when the User wants to view recommended venue nearby. 2. The User is logged in 3. A sports activity has been recommended to the User in previous page |
| **Post-Conditions** | 1. Information about the recommended venue is displayed |
| **Normal Flow** | 1. User taps(click??) the “Recommended Venue” button on the page that displays the recommended activity 2. The system detects (How??去找谁？) User’s current location 3. The system gets(request??) information about nearby available facilities/venues from external data sets from data.gov.sg. 4. The system calculates the distances from the available facilities/venues.   AF-1: If all facilities/venues are within(?) 5.0 kilometers aware from the user’s current location   1. The system displays the information of the facility/venue that has the shortest distance from the User’s current location |
| **Alternative Flows** | AF-1: The system displays a message that informs the User there is no available facilities/venues nearby. |
| **Exceptions** | - |
| **Includes** | Nil |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 06 |
| **Use Case Name** | Healthier Eateries Recommendation |
| **Use Case**  **Description** | Allows Helia Users to view recommended healthier eateries. |
| **Participating**  **Actor(s)** | User, External data sets |
| **Pre-Conditions** | 1. This use case extends the UC-04 Discover Activities. It is initiated when the User wants healthier eateries recommended. 2. The User is logged in |
| **Post-Conditions** | 1. A map marked with locations of the nearby healthier eateries is displayed |
| **Normal Flow** | 1. User taps(clicks?) the “Recommended Eateries” button on the page that displays the recommended activity 2. The system detects User’s current location 3. The system gets information about nearby healthier eateries from external data sets from data.gov.sg. 4. Calculate? Generate? 5. The system displays a map marked with locations of the healthier eateries around the User’s current location |
| **Alternative Flows** |  |
| **Exceptions** | Nil |
| **Includes** | Nil |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 07 |
| **Use Case Name** | View Events |
| **Use Case**  **Description** | Allows Helia Users to view the upcoming events |
| **Participating**  **Actor(s)** | User |
| **Pre-Conditions** | 1. The User is logged in |
| **Post-Conditions** | 1. A list of events will be shown |
| **Normal Flow** | * 1. The User clicks on Events Page   2. The System generates the list of events in the database   3. The User clicks on (Find??) the Event that he is interested in   AF-1: The User filters the list of events   * 1. The System displays the event details   2. The User clicks on ‘Interested’   AF-2: The User clicks on ‘Register’   * 1. The System adds the event to the User list of interested events |
| **Alternative Flows** | AF-1: The User filters the list of events   1. The System displays a list of categories that the User can filter 2. The User chooses which category the User is interested in 3. The System displays the list of event base on the category   AF-2: The User clicks on Register   1. The System adds the event to the User Registered event |
| **Exceptions** | Nil |
| **Includes** | Nil |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 08 |
| **Use Case Name** | Create Events |
| **Use Case**  **Description** | Allows Helia Users to create events for other Users to go to |
| **Participating**  **Actor(s)** | User |
| **Pre-Conditions** | 1. The User is logged in |
| **Post-Conditions** | 1. The event will be created |
| **Normal Flow** | * 1. The User clicks on “Create Events”   2. The System redirects the User to the Create Events Page   3. The User enters the Information of the events and clicks on the Create button   4. The System validates the information   AF-1: There are invalid inputs   * 1. The System add the event to the database |
| **Alternative Flows** | AF-1: There are invalid inputs   1. The System prompts the User of the invalid input 2. The User acknowledge and is taken back to 1.1 (how to write in sequence diagram) |
| **Exceptions** | Nil |
| **Includes** | Nil |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 09 |
| **Use Case Name** | View My Events |
| **Use Case**  **Description** | Allows Helia Users view the Events that they have created |
| **Participating**  **Actor(s)** | User |
| **Pre-Conditions** | 1. The User is logged in |
| **Post-Conditions** | 1. A list of events that the User have created will be shown |
| **Normal Flow** | * 1. The User clicks on My Events   2. The System displays a list of events that the User has created   3. The User clicks on one of the events that he is interested in checking(eventDetails)   4. The System displays the events details   5. The User clicks on Delete Event button   6. The System removes the event from the database |
| **Alternative Flows** | - |
| **Exceptions** | Nil |
| **Includes** | Nil |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 10 |
| **Use Case Name** | View Interested and Going |
| **Use Case**  **Description** | Allows Helia Users view all the Events that they are interested in or have registered for |
| **Participating**  **Actor(s)** | User |
| **Pre-Conditions** | 1. The User is logged in |
| **Post-Conditions** | 1. A list of events that the User have is interest in or have registered for will be shown |
| **Normal Flow** | * 1. The User clicks on Interested and Going Events   2. The System gives the User a choice between Interested and Registered   3. The User clicks on Interested   AF-1: The User clicks on Registered   * 1. The System displays the list of events that the User is interested in   2. The User clicks on one of the events   3. The System displays the events details   4. The User clicks on Remove Interest   5. The System removes the event from the interested list in the database |
| **Alternative Flows** | AF-1: The User clicks on Registered   * 1. The System displays the list of events that the User have registered for   2. The User clicks on one of the events   3. The System displays the events details   4. The User clicks on Remove Registration   5. The System removes the event from the registered list in the database |
| **Exceptions** | Nil |
| **Includes** | Nil |

|  |  |
| --- | --- |
| **Use Case ID** | UC – 11 |
| **Use Case Name** | Events Moderation |
| **Use Case**  **Description** | Allows Helia Admin view and manage all the Events that has been created |
| **Participating**  **Actor(s)** | User |
| **Pre-Conditions** | 1. The User is logged in |
| **Post-Conditions** | 1. A list of events will be shown for the admin to moderate |
| **Normal Flow** | * 1. The Admin clicks on Moderate Events   2. The System displays a list of events that has not been approved   3. The Admin clicks on one of the events   4. The System displays the events details   5. The Admin clicks on Approved   AF-1: The Admin clicks on Reject  AF-2: The Admin clicks on Delete   * 1. The System adds the event to the Approved list in the database |
| **Alternative Flows** | AF-1: The Admin clicks on Reject   1. The System adds the event to the Rejection list in the database   AF-2: The Admin clicks on Delete   1. The System removes the event from the database |
| **Exceptions** | Nil |
| **Includes** | Nil |