Use- cases ProCp

check points add

log in

weather conditions with a slider

add plane to airfield

Precondition  
The following preconditions are valid for every use-case mentioned under:

1. The application is running.

If there is a precondition unique for the simulation application, it will be mentioned in the specific use-case.

Exceptions   
The following exceptions are valid for every use-case mentioned under:

1. fdkjfbhfbh

If there is an exception unique for an application, it will be mention in the specific use-case.

# Actors

The actors of every mentioned application are the users. If there is a different actor, it will be mentioned in the specific use-case.

|  |  |
| --- | --- |
| USER REQUIREMENTS | |
| User task: | Saving data |
| User task description: | User should be able to save the generated data from the simulation. |
| FUNCTIONAL REQUIREMENTS | |
| Actors: | User |
| Triggers: | User wants to save its progress |
| Flow of the events: | 1. User clocks “Save” button 2. System opens a window 3. User names file 4. User saves file on desired location 5. Window closes |
| Postcondition: | System awaits a command |
| Exceptions: | 1.1 There is no generated data  System shows warning  3.1 File name already exists  User needs to rename file |

|  |  |
| --- | --- |
| USER REQUIREMENTS | |
| User task: | Loading data |
| User task description: | User should be able to load the saved data from the machine. |
| FUNCTIONAL REQUIREMENTS | |
| Actors: | User |
| Triggers: | User wants to load its progress |
| Flow of the events: | 1. User clicks “Load” button 2. System opens a window 3. User finds desired file 4. User clicks “open” 5. Data is loaded |
| Postcondition: | System awaits a command |
| Exceptions: | 3.1 File does not exist  4.1 File is corrupted |

|  |  |
| --- | --- |
| USER REQUIREMENTS | |
| User task: | Weather condition manipulation |
| User task description: | User should be able to manipulate the weather conditions during the simulation |
| FUNCTIONAL REQUIREMENTS | |
| Actors: | User |
| Triggers: | User wants to modify the weather condition |
| Flow of the events: | 1. User chooses between weather and wind sliders 2. User slides default position left or right until desired weather condition 3. Changes to the weather are applied |
| Postcondition: | System awaits a command |
| Exceptions: | 1.1 No exceptions???? |

Use case: Log in

Name: File open

Goal: User logs in the system and gains access to the simulator’s main page.

Actor: User

Pre: App is up and running, the log in screen is being displayed.

MSS:

1. User inputs his credentials (username/email and password) in the appropriate fields
2. User selects option to log in
3. System acknowledges the user’s details are correct and logs him in
4. User is redirected to the simulation screen

Extensions:

3a: System checks the user’s details and they are incorrect

.1 An error message, informing the user of his incorrect details is displayed

.2 Return to step MSS 1

**Things such as a limited number of attempts or a forgotten password option are implementable, but do we need them?**

Name: Edit checkpoints

Goal: User adds new checkpoints to the map or removes already existing ones, the system accepts the changes and adapts the routing appropriately.

Actor: User

Pre: App is up and running, the user has logged in and a simulation is running.

MSS:

1. User selects the option to enter checkpoint edit mode
2. The simulation pauses and the checkpoint edit mode is entered
3. User selects where would they like new checkpoints by clicking on the spot on the map where the checkpoints should be
4. A new checkpoint is displayed
5. User selects to exit checkpoint edit mode
6. Application accepts the changes with the checkpoints, adapts the routing to the new environment and proceeds with the simulation

Extensions:

3a: User removes the checkpoints he wishes gone by clicking on them

.1 The checkpoints disappear from the map

.2 Proceed to MSS 5

3a.2a: Proceed to MSS 2

3a.2b: No checkpoints/less than n checkpoints are left on the map; the system displays a warning message, informing the user of that

.1 Proceed to MSS 2

**I have made a case for editing checkpoints instead of just for their addition. I think it makes more sense this way.**

Name: Add airplanes to the air space

Goal: After a prompt by the user, a new airplane is generated and enters the airspace.

Actor: User

Pre: App is up and running, the user has logged in and a simulation is running.

MSS:

1. User selects the option to add a new airplane to the air space
2. A new airplane is generated with random settings such as position, speed, etc. and shown on the verge of the airspace, heading towards the airport
3. **Should we mention that the system acknowledges it and handle sits approach to the airport?**

**That’s the simple way, we could also be giving the user the freedom to input the planes, location, heading, attitude, etc.**

MSS:

1. User selects the option to add a new airplane to the air space
2. A set of controls regarding the airplane’s attributes such as position, heading, speed, altitude, etc. are displayed
3. With those controls the users sets the airplane’s attributes
4. User selects to “release” the aircraft
5. The airplane is generated and shown on the verge of the airspace with all the properties, defined by the user

Extensions:

4.a The user has inputted something incorrectly

.1 The system displays a warning message and asks the user to correct his input

.2 Return to MSS 3

4.b The user decides to discard the aircraft

.1 The set of controls for new airplane definition are dismissed

.2 Simulation proceeds without a change