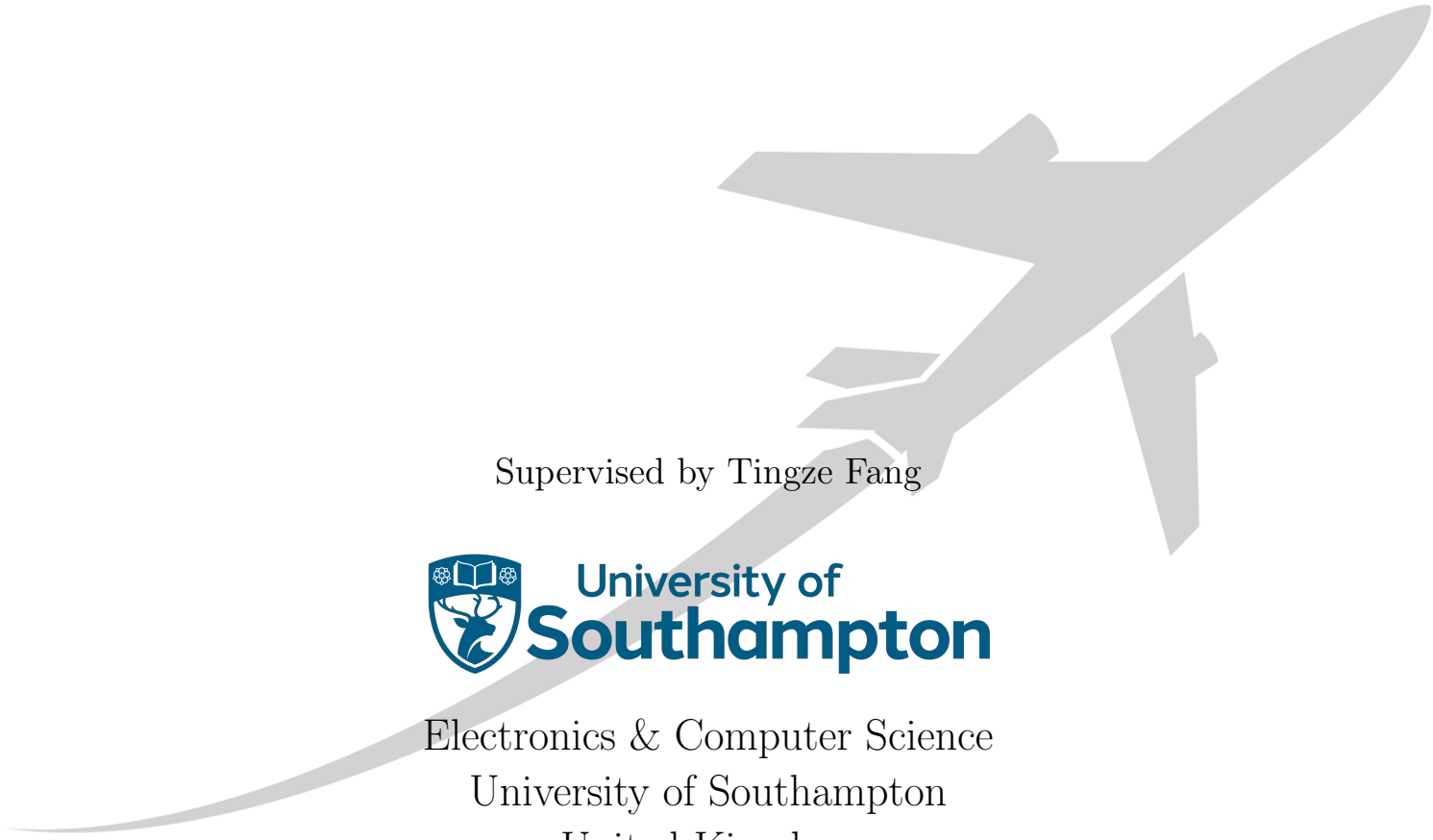
COMP2211
Software Engineering Group Project

Runway Re-declaration
Deliverable 4: Increment 3

Group 45

B. Jegatheeswaran (bj4g22@soton.ac.uk)
A. Geron (ag7g22@soton.ac.uk)
Y. Balasubramaniam (yb2e20@soton.ac.uk)
M. Gu (mg2n22@soton.ac.uk)
S. Zagrosi (sz10g21@soton.ac.uk)



Supervised by Tingze Fang

Electronics & Computer Science
University of Southampton
United Kingdom

Contents

Footnote: Cover page airplane graphic [1]

Deliverable 4: Increment 3

Introduction

1 Design

The following subsection details various elements that supported the design choices we made during the development of the application.

1.1 UML Diagrams

Unified Modelling Language (UML) is a visual modelling language that aids visualisation of a system. We decided to use three different UML diagrams to aid the development of our system.

1.1.1 Class Diagram

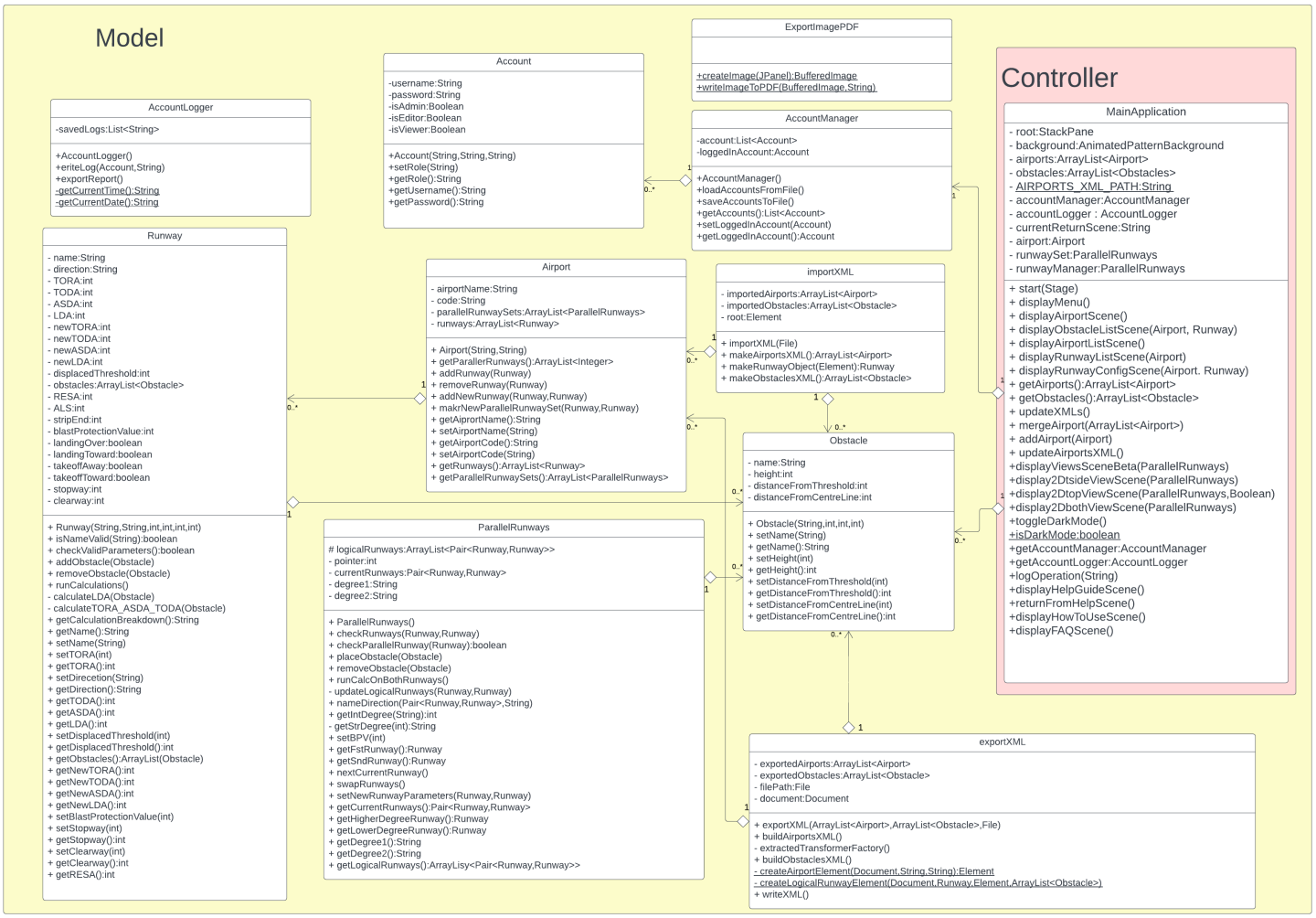


Figure 1: Model Controller class diagram

The class diagram shown above displays relationships between the model and controller classes. The majority of relationships associated in this class diagram is an aggregation relationship of one to many.

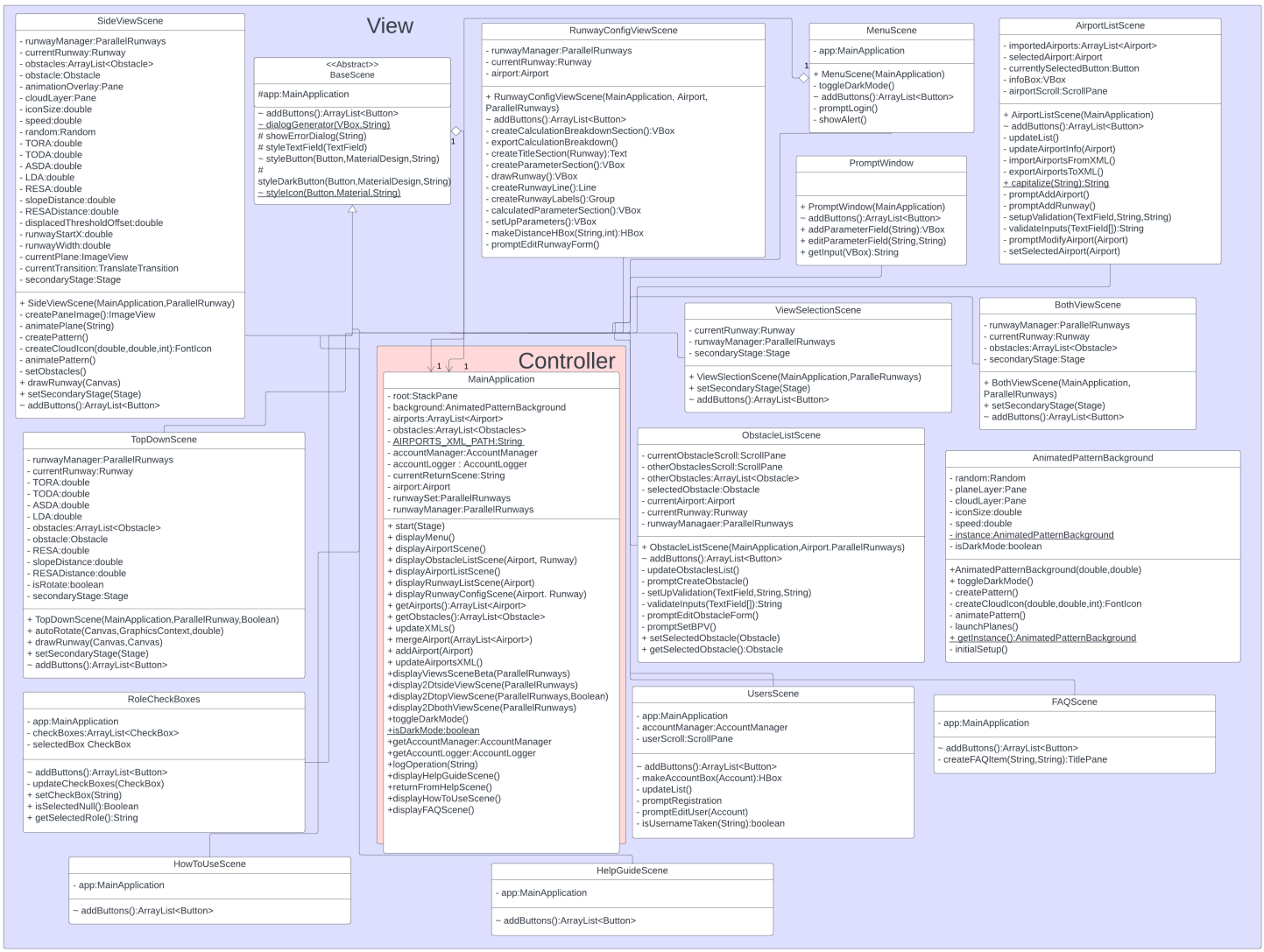


Figure 2: View Controller class diagram

The class diagram shown above displays relationships between the view and controller classes. The majority of relationships associated in this class diagram is an inheritance relationship between all classes ,excluding the controller class and AnimatedPatternBackground class, inheriting from the BaseScene class.

1.1.2 Use Case Diagram

Below is the Use Case Diagram created during increment 1.

Part 1 presents all the use cases for **Airfield Operational Manager**.

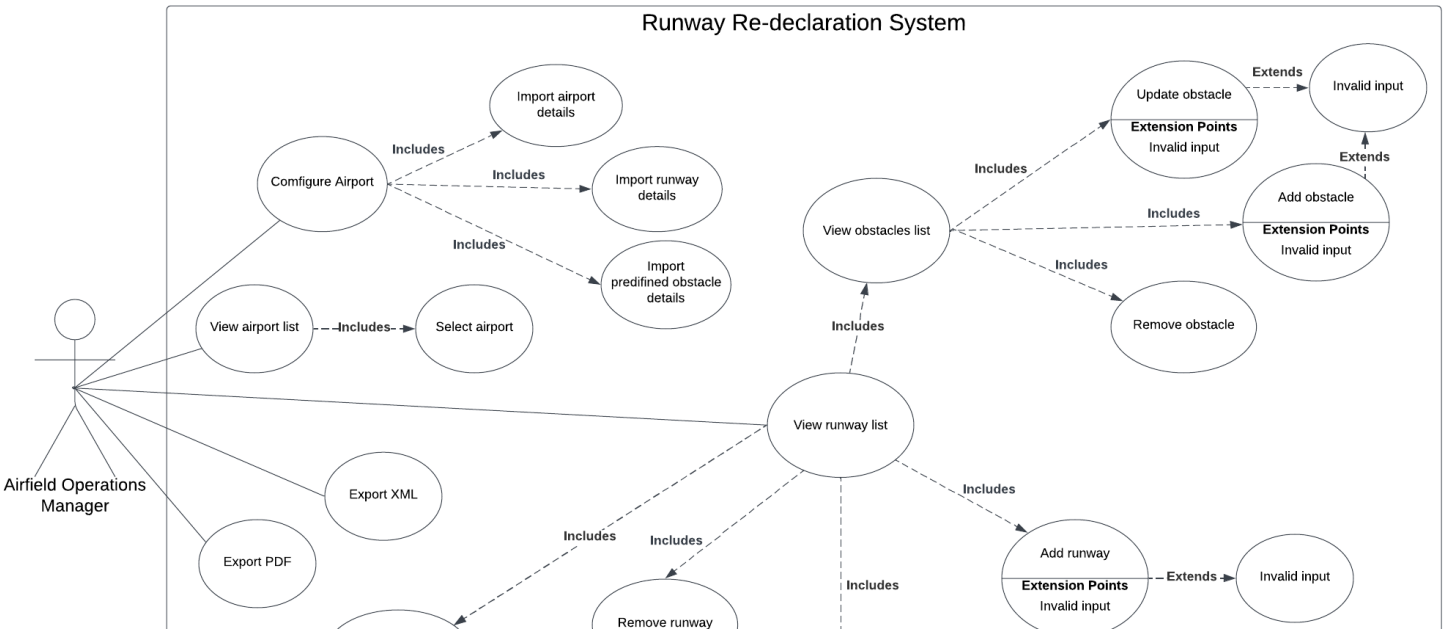


Figure 3: Use Case Diagram Part 1

Part 2 presents the rest of the use cases, including all the use cases for **Airfield Safety Officer and Air Traffic Controller**. The three stakeholders shares some common use cases. Please see attachment for the whole Use Case diagram.

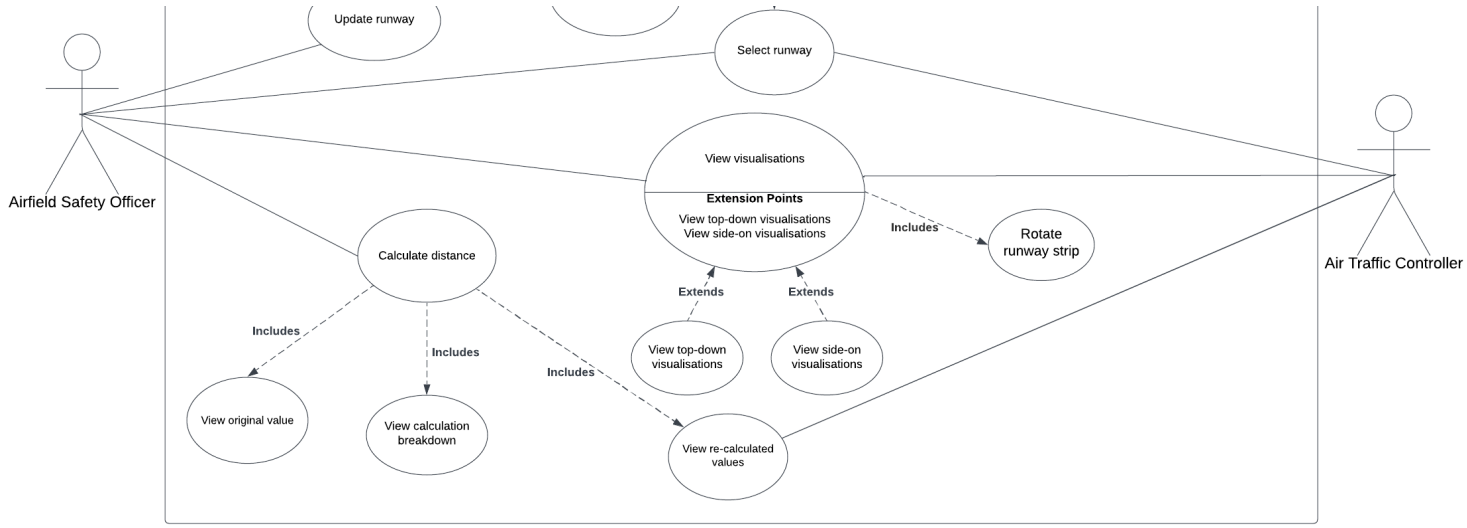


Figure 4: Use Case Diagram Part 2

1.1.3 State Machine Diagram

Below is the State Machine Diagram created during increment 1.

Part 1 presents the **Airport Configuring** state which contains sub-actions. The system enters this state when the system starts.

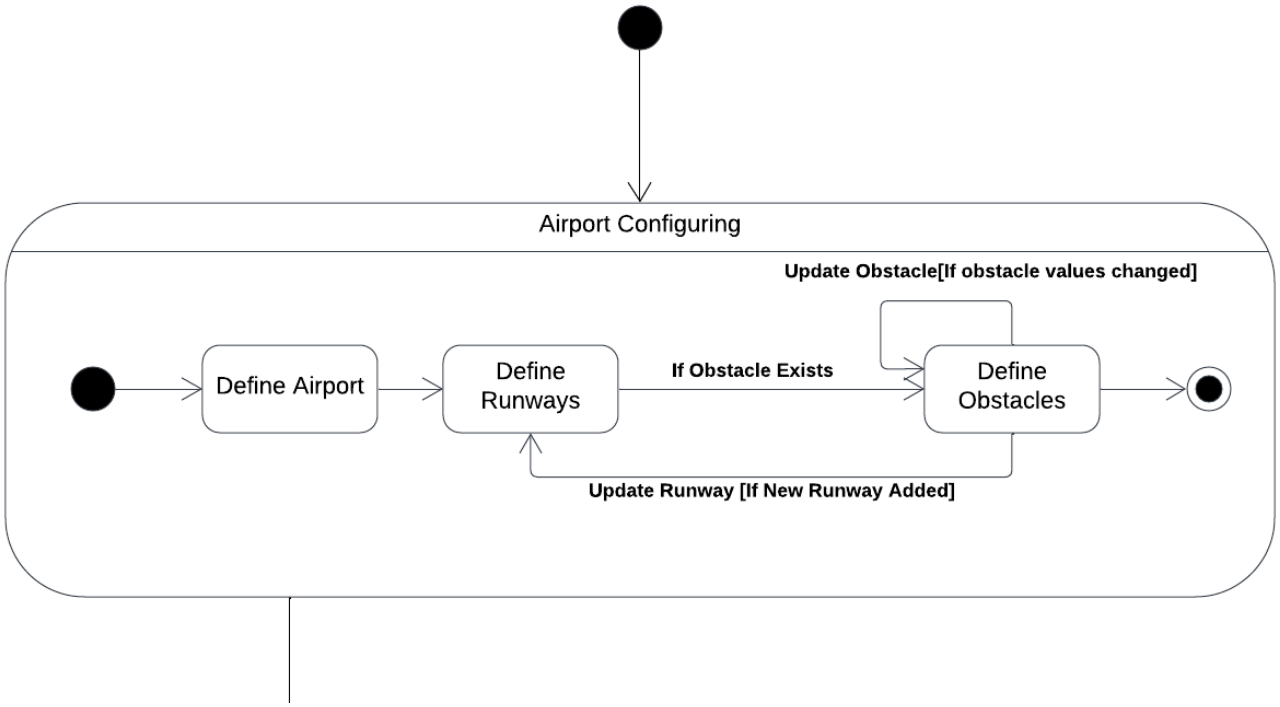


Figure 5: State Machine Diagram Part 1

Part 2 presents the rest of the states. The system enters these states when precondition satisfied.

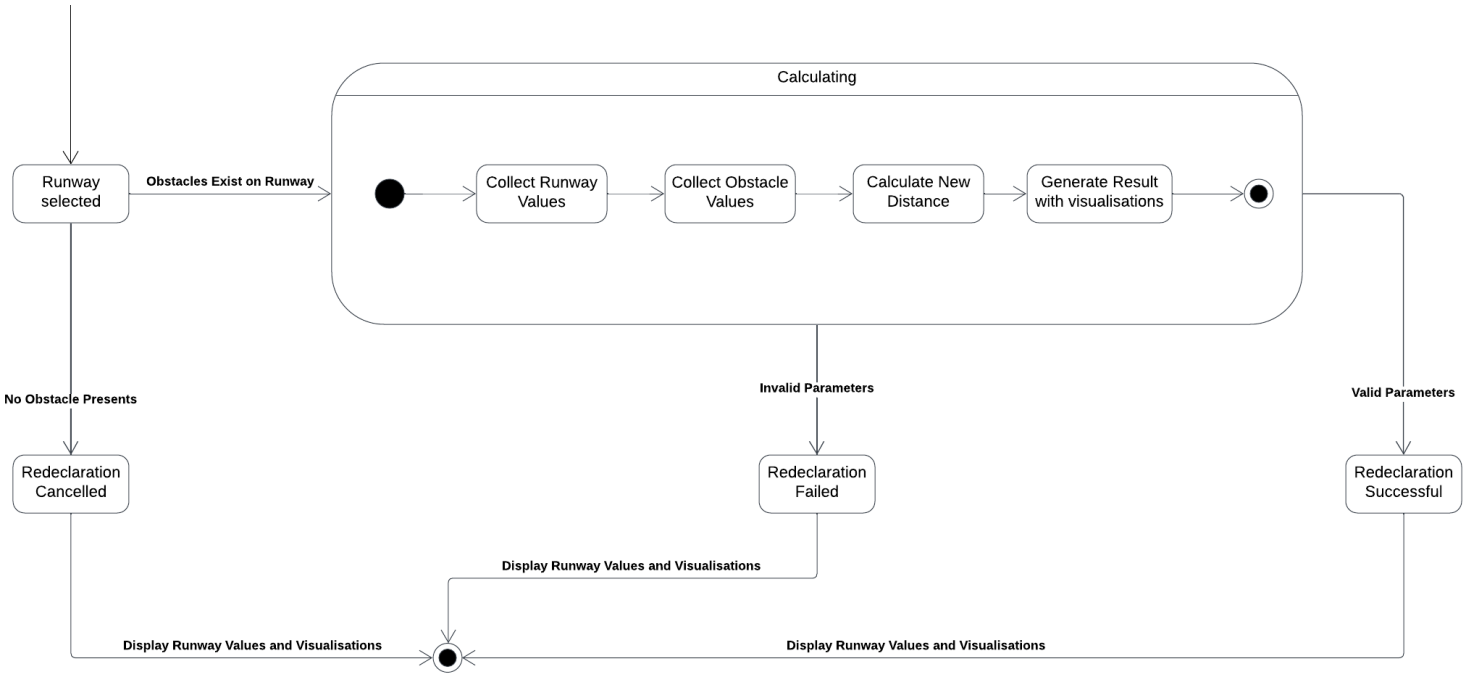


Figure 6: State Machine Diagram Part 2

Please see attachment for the whole State Machine diagram.

1.2 Scenarios

Scenarios are often used in software engineering to help with making design decisions. They describe specific instances of use, detailing the steps that the user and system take to achieve a goal. These narratives help in **understanding the context** of how the software will be used and are instrumental in user-centered design approaches.

Below are the scenarios that we created during Increment 1 (deliverable 2).

Scenario 1: Melisa (Airfield Operations Manager)

- Melisa opens the runway re-declaration software and login with her username and password.
- The access was authorized and the Airport Database page of the system appears with buttons to add and view list of airports.
- Melisa click on 'List', the screen displays an empty list of configured airport, with a warning message informing her that she need to add or import airport first.
- She clicks the 'Add' button from the list of options on the page, a 'Add Airport' dialog appears with text fields she need to fill in with airport name and code.
- Melisa only enters the name of the airport she's working at and clicks 'Add', an alert message appears informing her that she need to fill in all fields.
- Melisa clicks 'OK' and fills in the airport code, then clicked 'Add' again. The airport she entered appears on the airport list.
- She clicks on the airport, then clicks on 'Select' button, the list of runways in the airport appears on the screen.
- The list in empty, Melisa selects the option to 'Import' runway data from her airport. This quickly adds all the runways of the airport to the list.
- She clicks on a runway that recently has been reported to have an obstacle exist on it. This takes her to the runway display page, with all information about this runway listed.
- She then defines a new obstacle by navigating to 'Obstacles -¿ Create', this prompts a 'Create Obstacle' dialog.
- Melisa fills in all the fields, and pressed 'Create', the obstacle is created and added to the list of predefined obstacles.

Scenario 2: Captain Maxine (Airline Pilot)

- Maxine runs the runway re-declaration software clicks the "login" button.
- Maxine enters the username and password fields.
  - If the login credentials are incorrect, an alert window will appear and say "Incorrect Username or password".
- The airport database screen is shown on the application.
- Maxine selects the "Add" button which opens a prompt for the airport details.
- Maxine enters the airport details and clicks on "Add"
- The prompt closes, and the she clicks the "List" button
- The airport that she has inputted is now on display on the Airport List page.
- She clicks on the airport, as she made a mistake.
- Maxine clicks on the "Delete" button and the airport that was clicked is now removed from the airport list.
- The airport's runway list is updated on the screen and presents the updated list to Maxine.

Scenario 3: Stacey (Airfield Safety Officer)

- Stacey runs the runway re-declaration software and logs in with her username and password.
- Shes now given access to the Airport Database page of the system appears with buttons to add and view list of airports.
- Stacey clicks on the airport she works in labelled "LHR" the code for the Heathrow Airport.
- She clicks the "select" button and is now on the runway list page of that airport.
- She clicks on the runway that he is operating on which is the 27L runway and presses select.
- Stacey's screen now presents the configurations for the current runway.
- She clicks the obstacle button to get to the Obstacle list screen.
- She selects the "Car" Obstacle and it moves to the current Obstacles screen.
- She then clicks "Return" and prompt opens that requires Blast Protection Value to be inputted.
- Andrew inputs the appropriate value.
  - If the value inputted is invalid, such as inputting a negative value a messaged is displayed "Invalid measurements for Blast Protection Value"
- Stacey is now displayed the old and new parameters side by side along with a full breakdown of the calculations.
- Stacey wants to quickly change the Obstacle and goes back to the Obstacle screen.
- Stacey selects the "Airplane" obstacle in the other obstacles box.
- Stacey "clicks" add and the "Car" and "Airplane" obstacles swap places.
- Stacey clicks "Return" and inputs the Blast Protection Value in the prompt.
- Stacey is presented the NEW calculated values side by side by the default parameters.

Scenario 4: James (Air Traffic Controller)

- James opens up the runway re-declaration software and enters his login credentials.
- The airport database screen is shown, where she has the options to add new airports, view the list of airports or go back.
- James clicks on "List" to then get to the page that displays the list of available airports.
- James clicks on the "import" button which opens up his files directories.
- James selects the xml file he wants to import.
- The airport list screen updates with the airports that are in the xml file to then easily view them.
- He selects the Heathrow Airport and is presented the runway list page.
- He selects a runway from the runway list.
- James is presented with all runway information and options to view visualisations of the runway.
- James navigates to 'Visualisations -¿ side view', and is presented with the side view of the runway only.

Scenario 5: Andrew (Airfield Safety Officer)

- Andrew opens the runway re-declaration software and login with her username and password.
- The access was authorized and the Airport Database page of the system appears with buttons to add and view list of airports.
- Andrew clicks on the airport he works in labelled "LCY" the code for the London City Airport.
- He opens the list of runways by pressing the select button.
- He clicks on the runway that he is operating on which is the 09R runway and presses select.
- Andrew's screen now presents the configurations for the current runway.
- He clicks the obstacle button to get to the Obstacle list screen.
- He clicks on "create" and writes down the configurations for the obstacle.
- He writes down "Broken down airplane" and gives it's height, distance from threshold and centreline.
  - If the parameters are invalid, such as inputting a negative value a messaged is displayed "Invalid measurements for obstacle"
- Andrew then clicks "Return" and prompt pops up that requests the specific Blast Protection Value waiting for user input.
- Andrew inputs the appropriate value.
  - If the value inputted is invalid, such as inputting a negative value a messaged is displayed "Invalid measurements for Blast Protection Value"
- Andrew is now displayed the old and new parameters side by side along with a full breakdown of the calculations.

1.3 Storyboards

Since last increment, we have made some changes to our storyboards which have been used to develop our UI. Please see below for our updated storyboards.

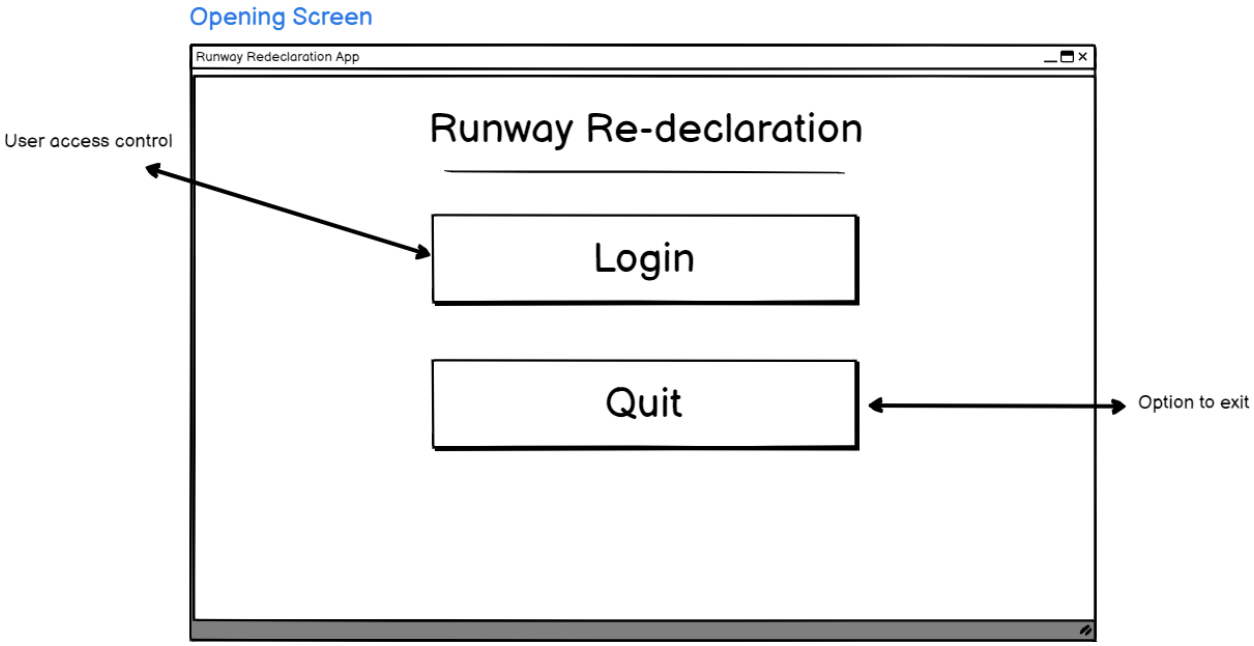


Figure 7: Login page 1 - User can log in to the system by clicking on the 'Login' button

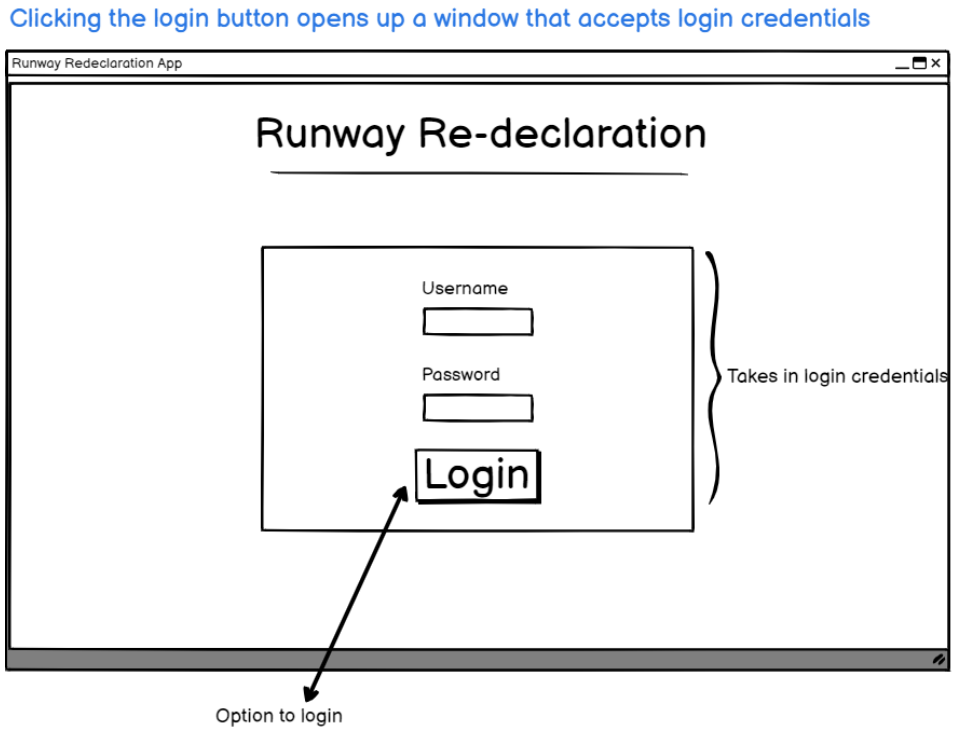


Figure 8: Login page 2 - Only correct credentials will be accepted

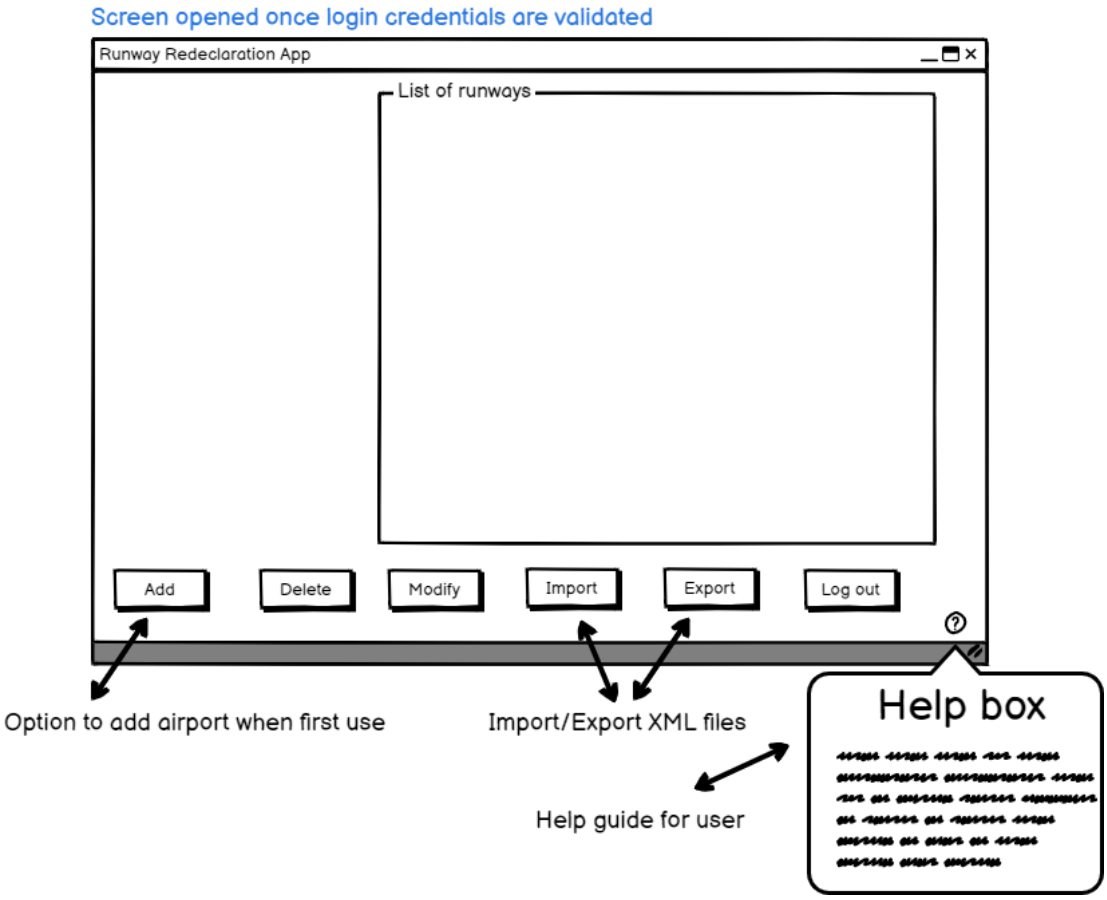


Figure 9: Empty Airport page - user needs to configure airport first

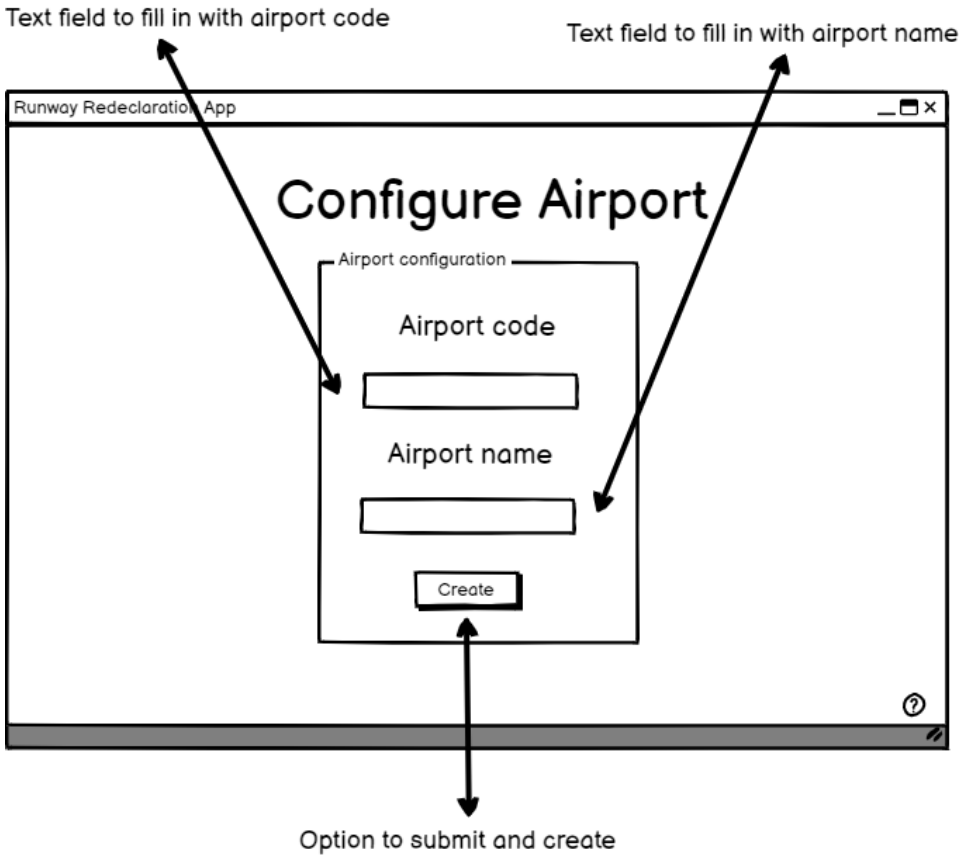


Figure 10: This page will pop up when user clicks on add airport for configuring airport

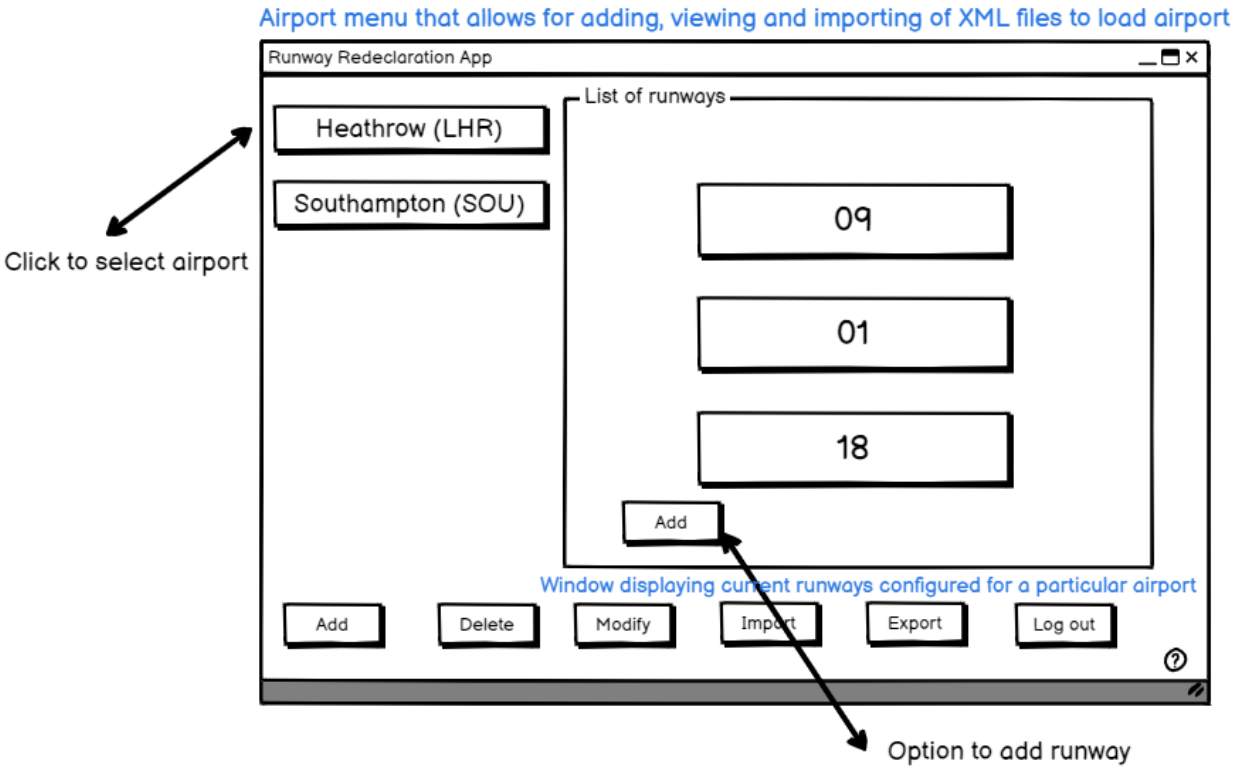


Figure 11: After selecting airport, user can now select runways to view specific runway

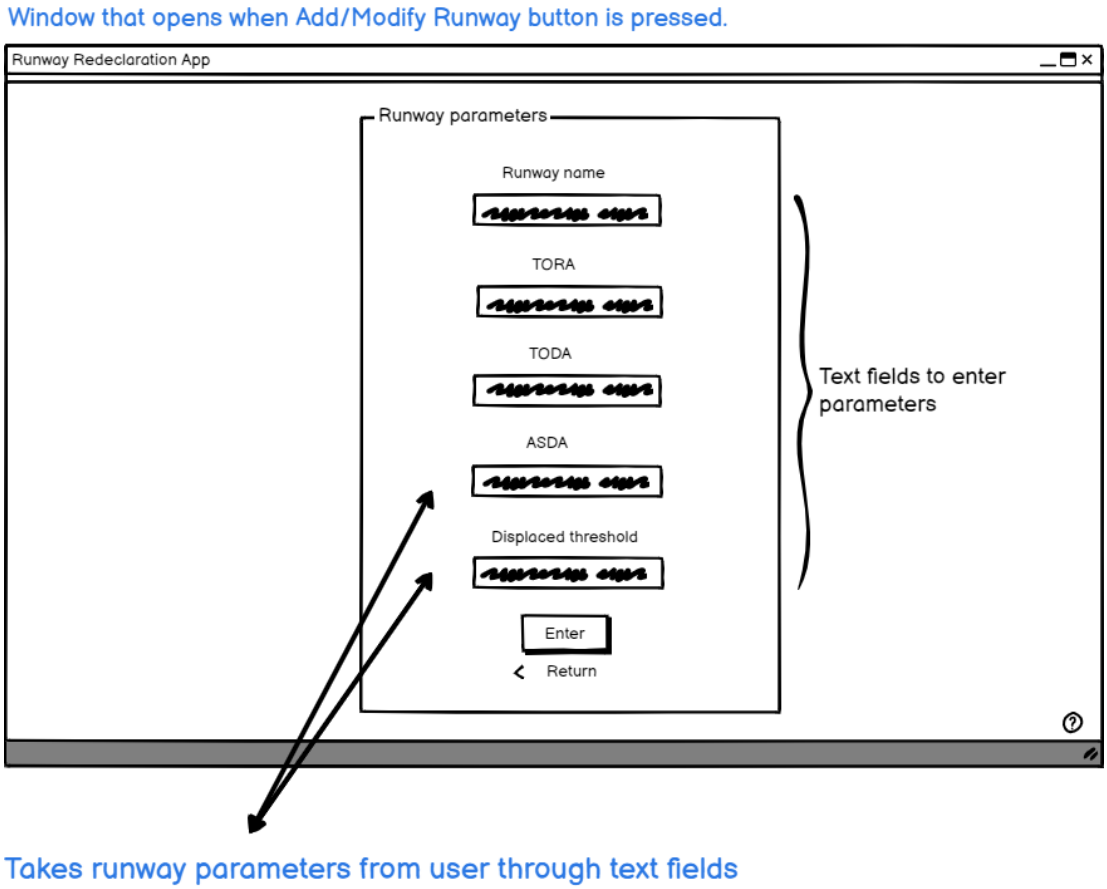


Figure 12: This will pop up when user clicks on add runway

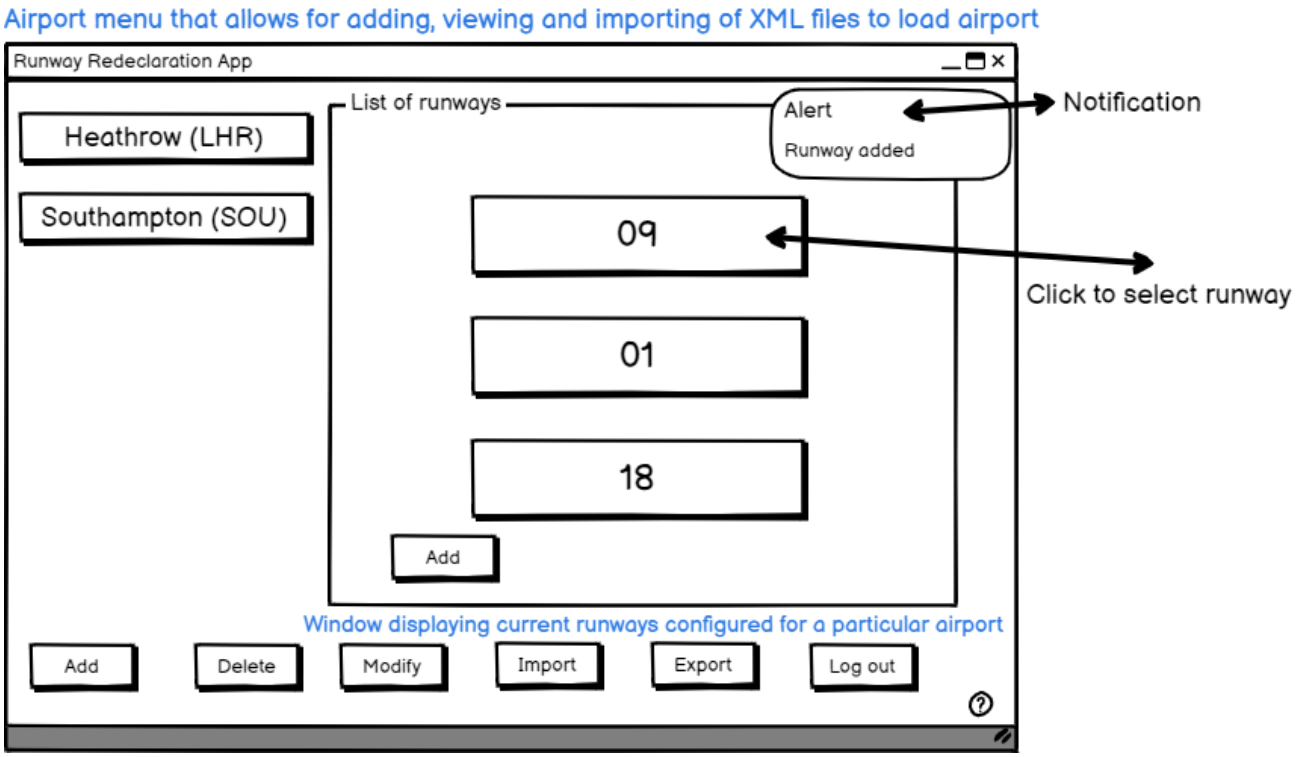


Figure 13: The list of runways will update according to user's input and a notification message will show indicating the operation.

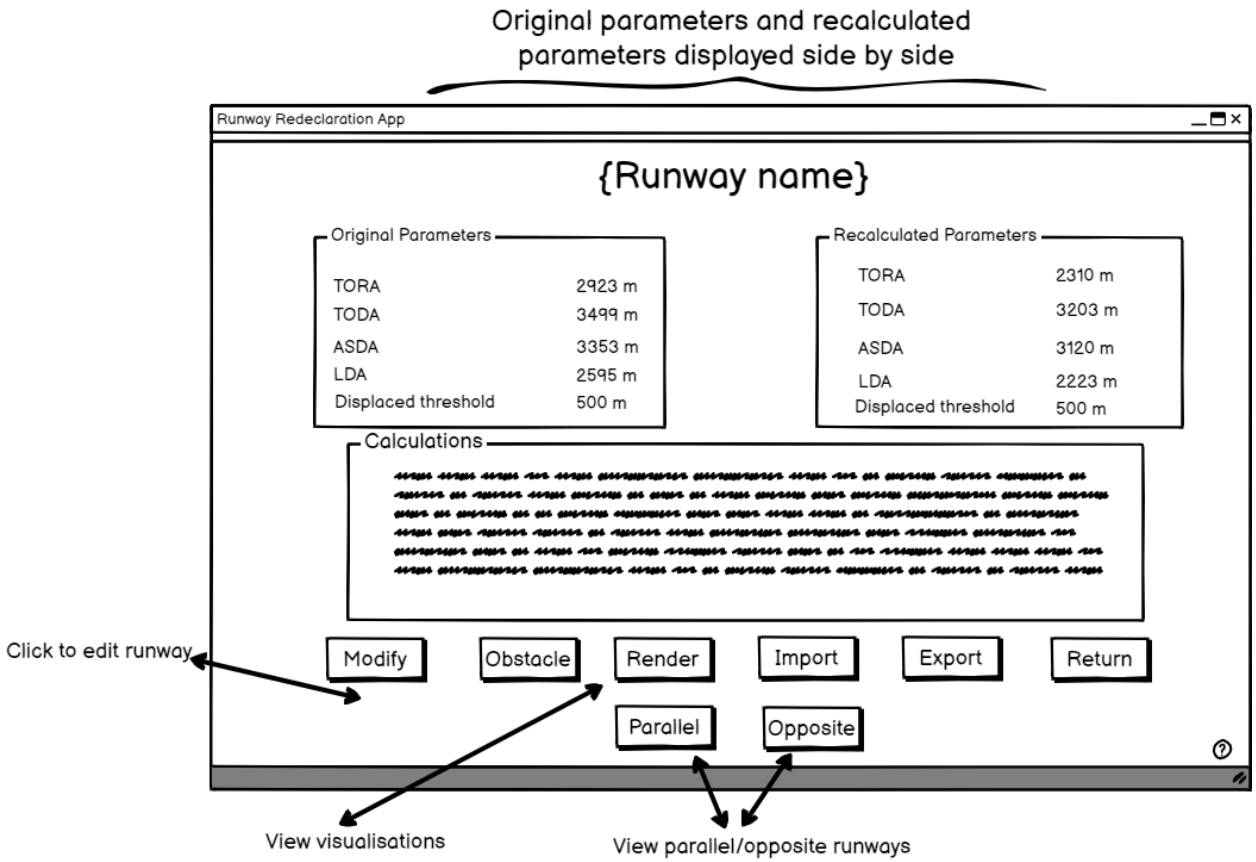


Figure 14: After selecting a runway, user can view details about the selected runway

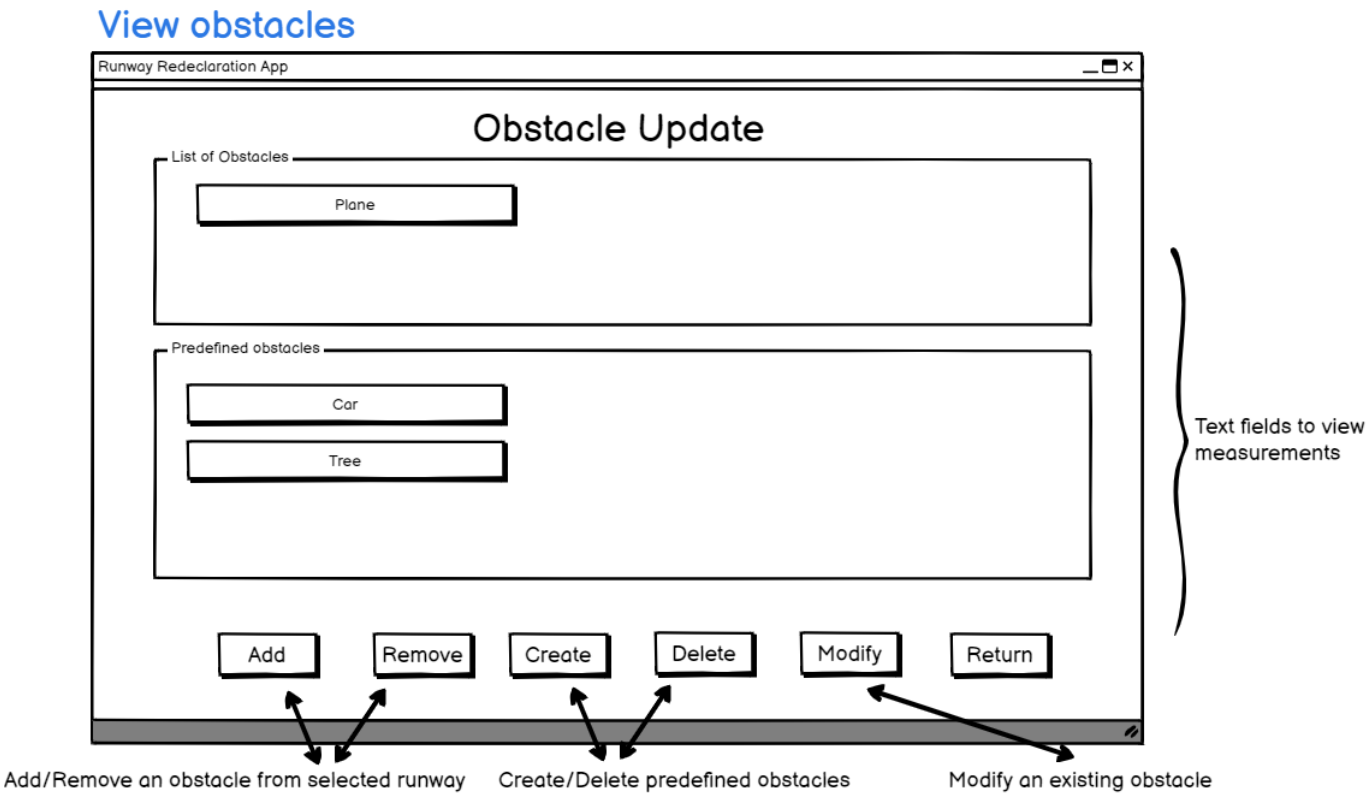


Figure 15: User have the option to view the obstacle on selected runway and any predefined obstacles

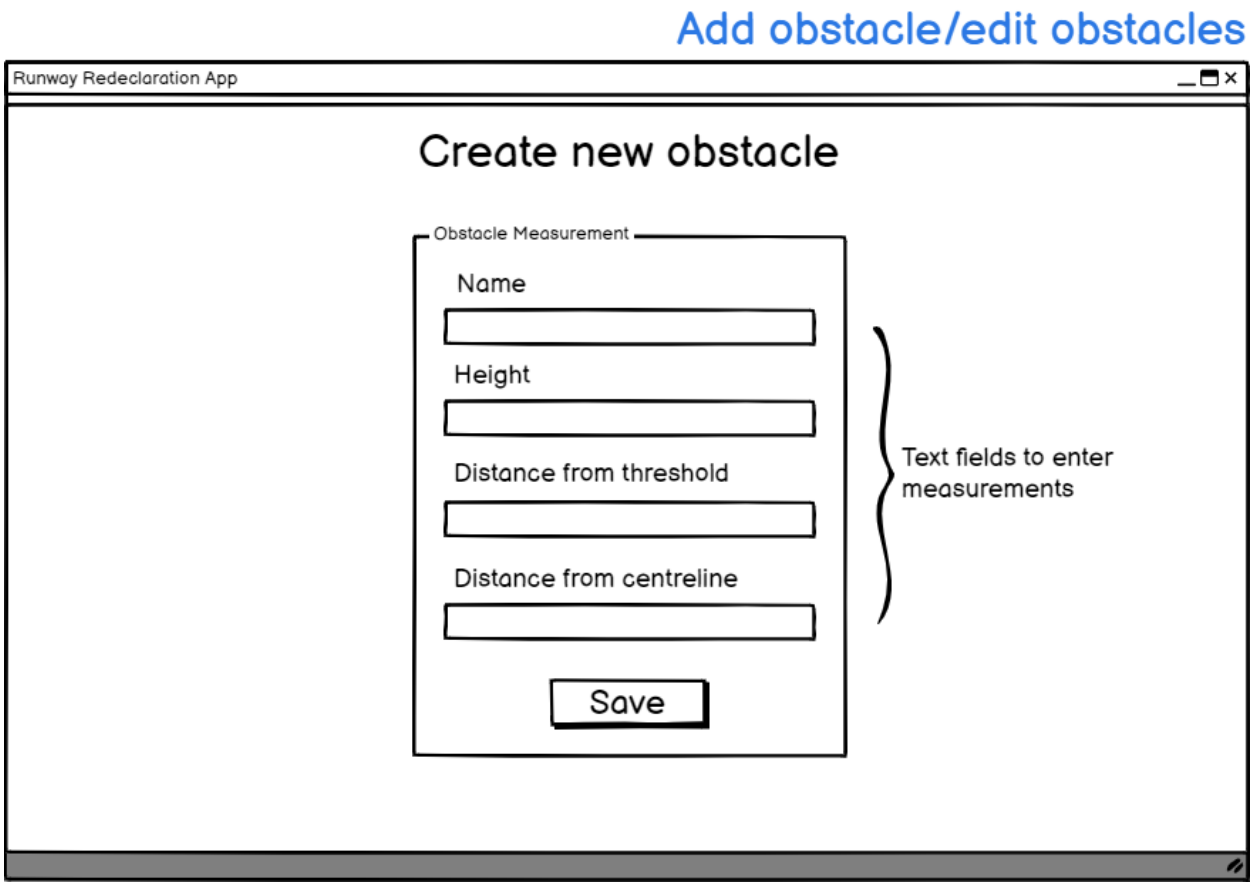


Figure 16: This page will pop up when user selects to create an obstacle

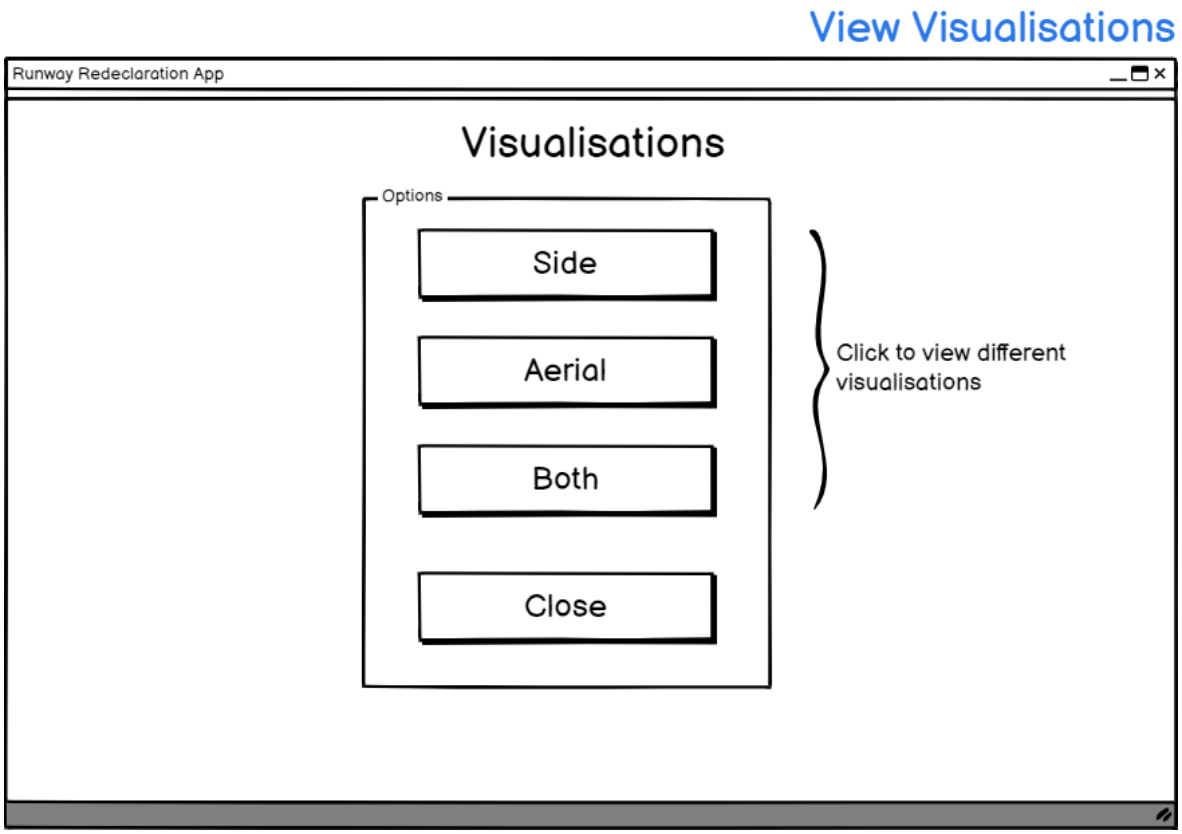


Figure 17: User can click on Render to view the options for visualisations

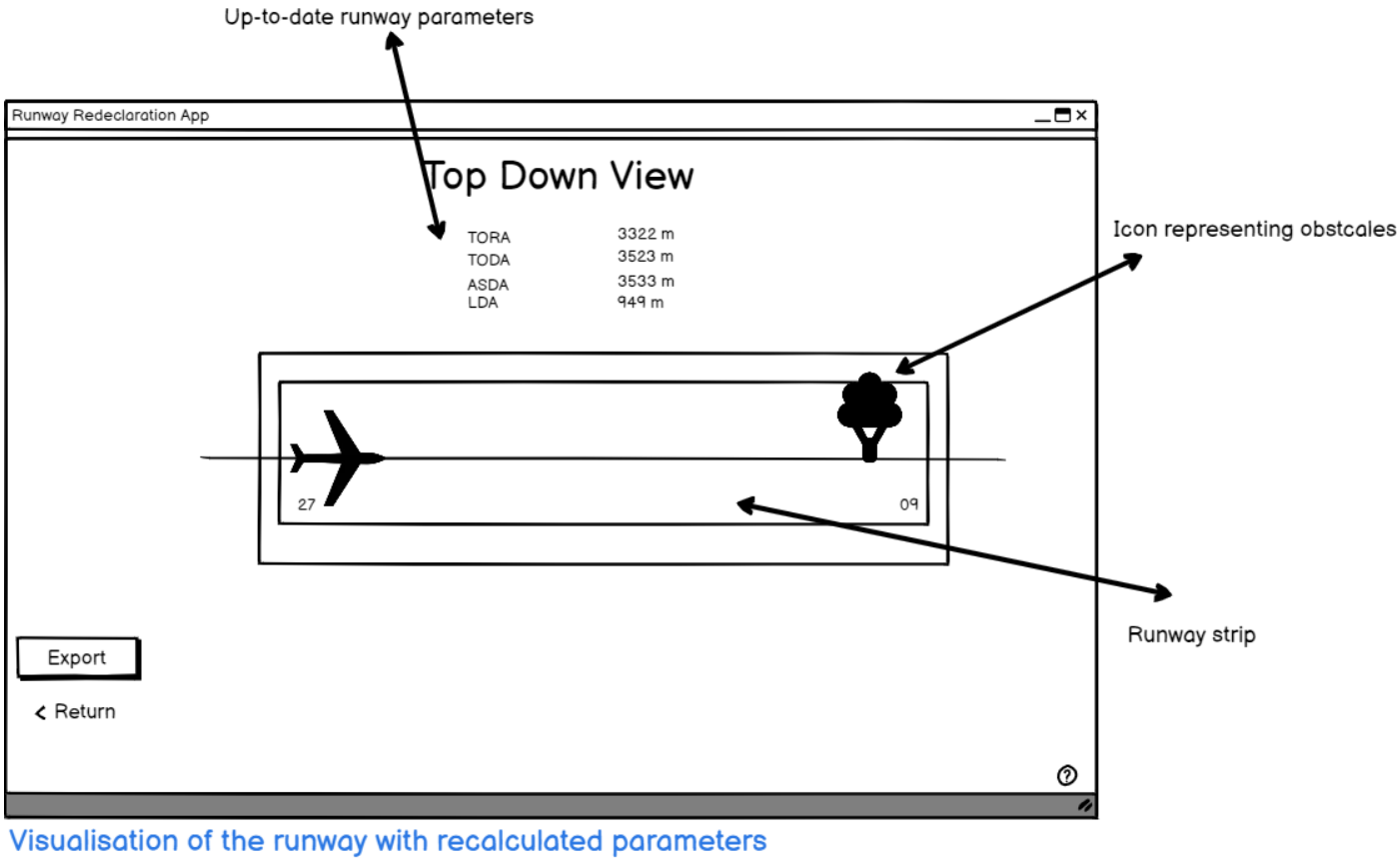


Figure 18: This displays the top down view of the selected runway

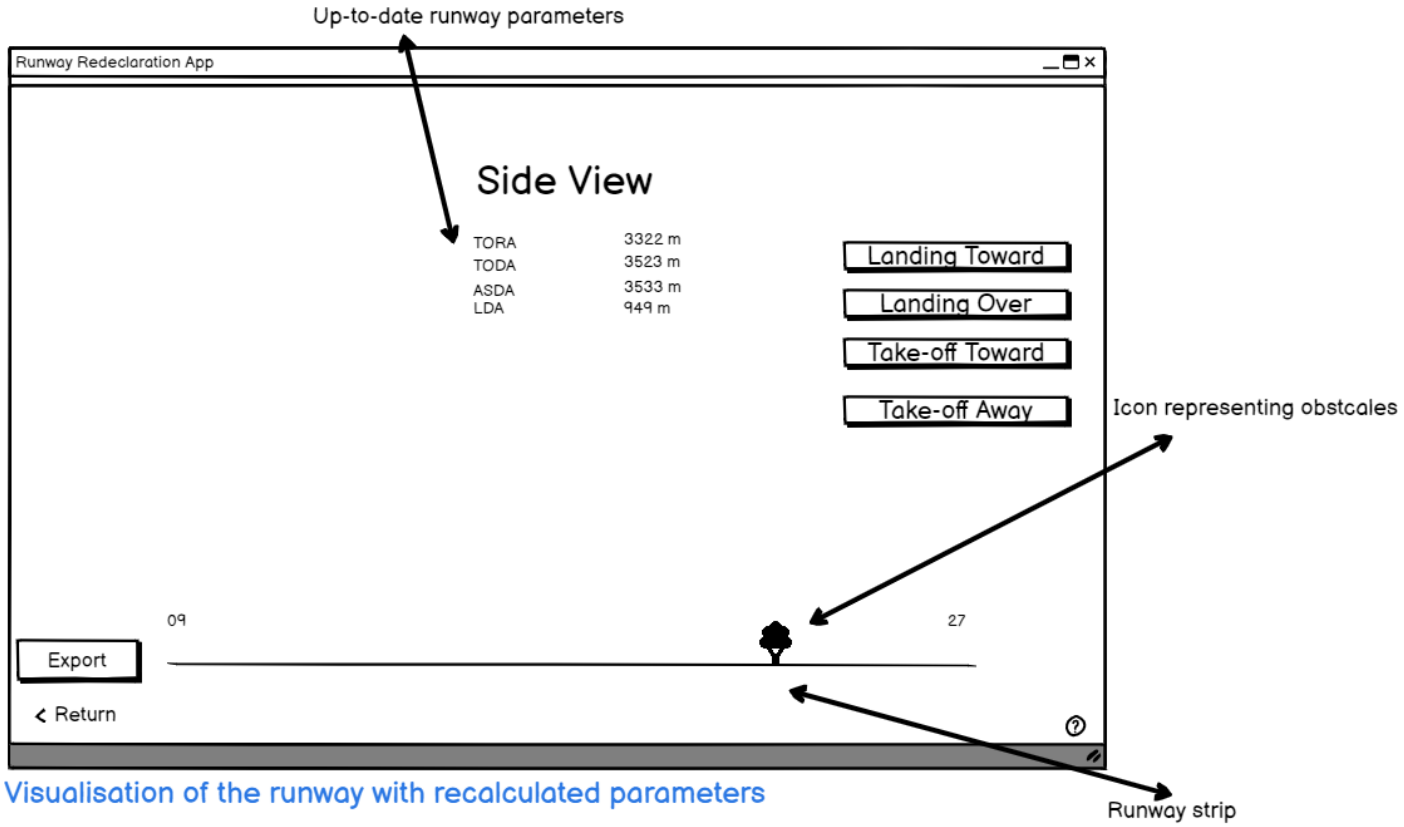


Figure 19: This displays the side on view of the selected runway

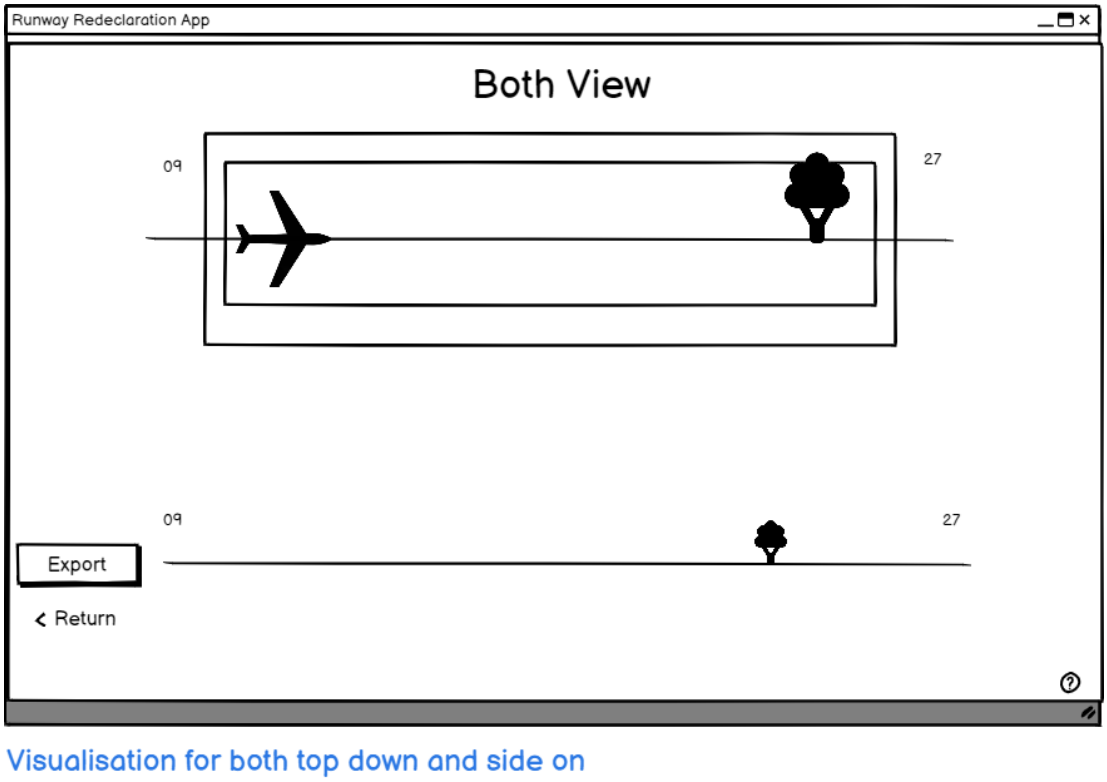


Figure 20: This displays both the side on and top down views of the selected runway

2 Testing

Testing is a critical phase that aims to evaluate the product's functionality, reliability, performance, and security.

To test our application, we used a range of techniques including **defect** and **validation testing**, also **boundary, partitioning, and regression testing**. These various testing techniques focuses on the functionality of our application, which will help us to thoroughly test our program to make sure it functions as required.

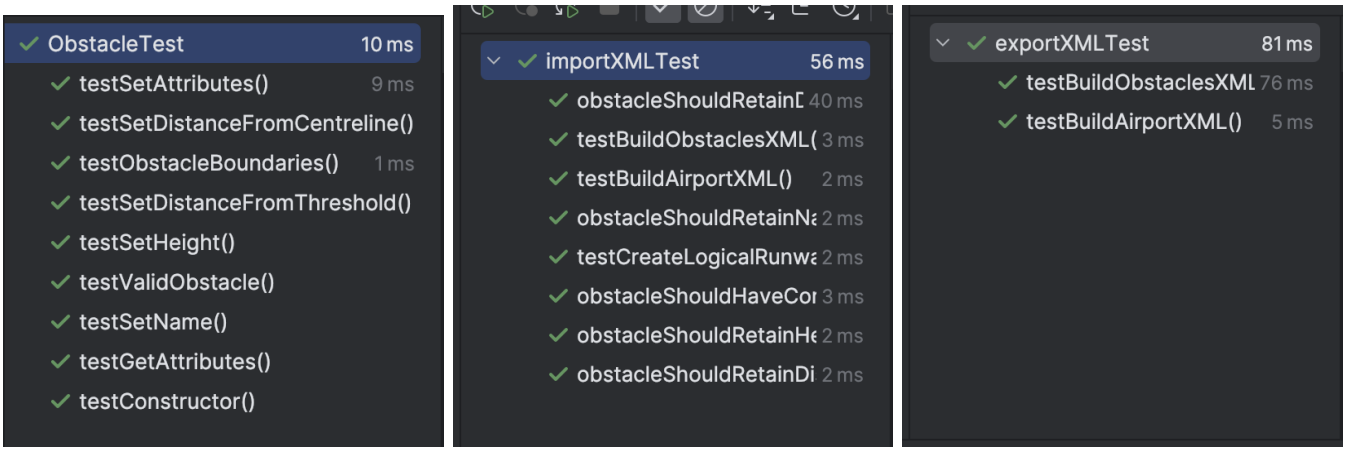
We also been using other testing techniques such as **Acceptance Testing** and **Scenario testing**, which will focuses on the usability and user experience of our application. We wish to use these techniques to test and ensure our program will accurately perform tasks that our users requires whist maintaining a joyful experience for them.

For each testing technique we will provide 1-2 test examples to demonstrate our result and usage of particular technique.

2.1 Unit tests

2.1.1 Defect testing

Test Definition: Defect testing, also known as bug testing, focuses on identifying defects in the software where the behaviour of the application deviates from the expected results. It involves executing a component or system to find any defects that could cause a failure in operation.
Reason for testing: Defect testing is crucial for identifying specific errors or bugs in the code. By focusing on finding defects, we can ensure that the individual components of the software function correctly in isolation, leading to more reliable and error-free software.
Test Examples: Identified that the cause of defects could potentially come from import XML, export XML as well as incorrectly inputted runway parameters.
Solved this by ensuring runway, obstacle classes correctly accept and store data.



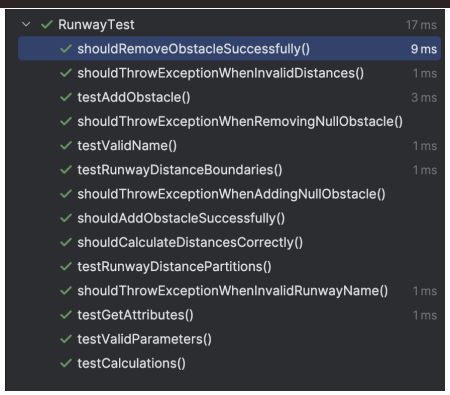
Test Result: ALL PASSED

2.1.2 Validation testing

Test Definition: Validation testing is the process of evaluating software at the end of the development process to determine if it meets the specified requirements. It answers the question, "Are we building the right product?" and ensures that the software fulfils the intended use and goals of the stakeholders.
Reason for testing: Validation testing is important to verify that the software meets all the user requirements. It ensures that the product fulfils its intended use and behaves as expected in the real-world scenario, thus guaranteeing customer satisfaction and usability.
Test Examples: The Obstacle and Runway Classes require user input so the following methods are tested:

```
14 usages  Anthony Geron +2 *
public boolean checkValidParameters(){ // Checks parameters, returns true if all cases are met
    boolean greaterThanZero = (TORA>0 && TODA>0 && ASDA>0 && LDA>0 && displacedThreshold>=0 && stopway>=0 && clearway>=0);
    boolean checkTODA = (TODA == TORA + clearway && TODA >= TORA);
    boolean checkASDA = (ASDA == TORA + stopway || ASDA == LDA + stopway);
    boolean checkMinimum = TORA >= 1800;
    boolean checkMaximum = TORA <= 4000;
    return (greaterThanZero && checkTODA && checkASDA && checkMaximum && checkMinimum);
}
```

```
20 usages  Anthony Geron
public boolean validObstacle(Runway runway) {
    boolean maxHeight = height < 35 && height >= 10;
    boolean distanceThreshold = (distanceFromThreshold < runway.getTORA() && distanceFromThreshold > -runway.getDisplacedThreshold());
    boolean distFromCent = (distanceFromCentreline < 100);
    return (maxHeight && distanceThreshold && distFromCent);
}
```



Test Result: ALL PASSED

2.1.3 Boundary testing

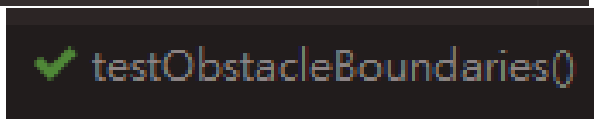
Test Definition: Boundary testing is a method where the focus is on the boundary or limit conditions of the software being tested. It involves testing at the extreme ends or boundaries of input values. The goal is to identify any errors that occur at the edges of the input domain, such as maximum, minimum, just inside/outside boundaries, typical values, and error values.

Reason for testing: Boundary testing is essential for ensuring that the software behaves correctly at the boundary limits of input values. This kind of testing checks for potential errors at the edges of input ranges, which are common points of failure in software. This can enhance the robustness and reliability of the product.

Test Examples: Boundary tests are

```
Anthony Geron +1
@Test
void testRunwayDistanceBoundaries() {
    assertThrows(IllegalArgumentException.class, () -> new Runway( name: "09", stopway: 0, clearway: 0, TORA: 4001, displacedThreshold: 307).checkValidParameters()); // Invalid distance testing > 4000m 4001
    assertTrue(new Runway( name: "09", stopway: 0, clearway: 0, TORA: 4000, displacedThreshold: 307).checkValidParameters()); // Valid distance within range of 1800-4000m 4000

    assertThrows(IllegalArgumentException.class, () -> new Runway( name: "09", stopway: 0, clearway: 0, TORA: 1799, displacedThreshold: 307).checkValidParameters()); // Invalid distance testing < 1800m 1799
    assertTrue(new Runway( name: "09", stopway: 0, clearway: 0, TORA: 1801, displacedThreshold: 307).checkValidParameters()); // Valid distance within range of 1800-4000m 1801

}
```

```
Anthony Geron
@Test
void testObstacleBoundaries() {
    Runway runway = new Runway( name: "09", stopway: 0, clearway: 0, TORA: 3660, displacedThreshold: 307);
    assertTrue(obstacle.validObstacle(runway));

    obstacle.setDistanceFromCentreline(99);
    assertTrue(obstacle.validObstacle(runway)); // Distance from centre < 100

    obstacle.setDistanceFromCentreline(100);
    assertFalse(obstacle.validObstacle(runway)); // Distance from centre > 100

    obstacle.setDistanceFromCentreline(50);
    obstacle.setHeight(34);
    assertTrue(obstacle.validObstacle(runway));  // Height < 35

    obstacle.setHeight(35);
    assertFalse(obstacle.validObstacle(runway)); // Height > 35

    obstacle.setHeight(10);
    assertTrue(obstacle.validObstacle(runway));  // Height < 10

    obstacle.setHeight(9);
    assertFalse(obstacle.validObstacle(runway)); // Height > 10

    obstacle.setHeight(20);
    obstacle.setDistanceFromThreshold(3659);
    assertTrue(obstacle.validObstacle(runway)); // Distance from threshold < TORA

    obstacle.setDistanceFromThreshold(3660);
    assertFalse(obstacle.validObstacle(runway)); // Distance from threshold > TORA

    obstacle.setDistanceFromThreshold(-306);
    assertTrue(obstacle.validObstacle(runway)); // Distance from threshold < - displaced threshold

    obstacle.setDistanceFromThreshold(-307);
    assertFalse(obstacle.validObstacle(runway)); // Distance from threshold > - displaced threshold
}
```



Test Result: ALL PASSED

2.1.4 Partition testing

Test Definition: Partition testing is a method where input data of a software application is divided into partitions of equivalent data from which test cases are derived. It reduces the testing workload by identifying a few test cases that are representative of larger groups of similar cases. So if one test case in a partition passes, all others will too, and similarly for failures.

Reason for testing: Partition testing is important for efficiently checking software functionality across a range of input values. This method helps in identifying class-level issues in the software, thereby ensuring thorough and efficient testing coverage.

Test Examples: We included partition tests in Obstacle class and Runway class:

```
Anthony Geron
est
id testValidParameters(){ // BOUNDARY TESTING FOR NEGATIVE VALUES
  assertTrue(runway.checkValidParameters());
  assertTrue(new Runway( name: "09", stopway: 0, clearway: 0, TORA: 3660, displacedThreshold: 307).checkValidParameters());

  assertThrows(IllegalArgumentException.class, () -> new Runway( name: "09", stopway: -1, clearway: 0, TORA: 3660, displacedThreshold: 307).checkValidParameters());
  assertThrows(IllegalArgumentException.class, () -> new Runway( name: "09", stopway: 0, clearway: -1, TORA: 3660, displacedThreshold: 307).checkValidParameters());
  assertThrows(IllegalArgumentException.class, () -> new Runway( name: "09", stopway: 0, clearway: 0, TORA: -1, displacedThreshold: 307).checkValidParameters()); //
  assertThrows(IllegalArgumentException.class, () -> new Runway( name: "09", stopway: 0, clearway: 0, TORA: 3660, displacedThreshold: -1).checkValidParameters()); //
```

```
@Test
void testRunwayDistancePartitions() {
    assertTrue(new Runway( name: "09", stopway: 0, clearway: 0, TORA: 3660, displacedThreshold: 307).checkValidParameters()); // Valid distance within range of 1
    assertThrows(IllegalArgumentException.class, () -> new Runway( name: "09", stopway: 0, clearway: 0, TORA: 4500, displacedThreshold: 307).checkValidParameters(
    assertThrows(IllegalArgumentException.class, () -> new Runway( name: "09", stopway: 0, clearway: 0, TORA: 1500, displacedThreshold: 307).checkValidParameters(
}
```

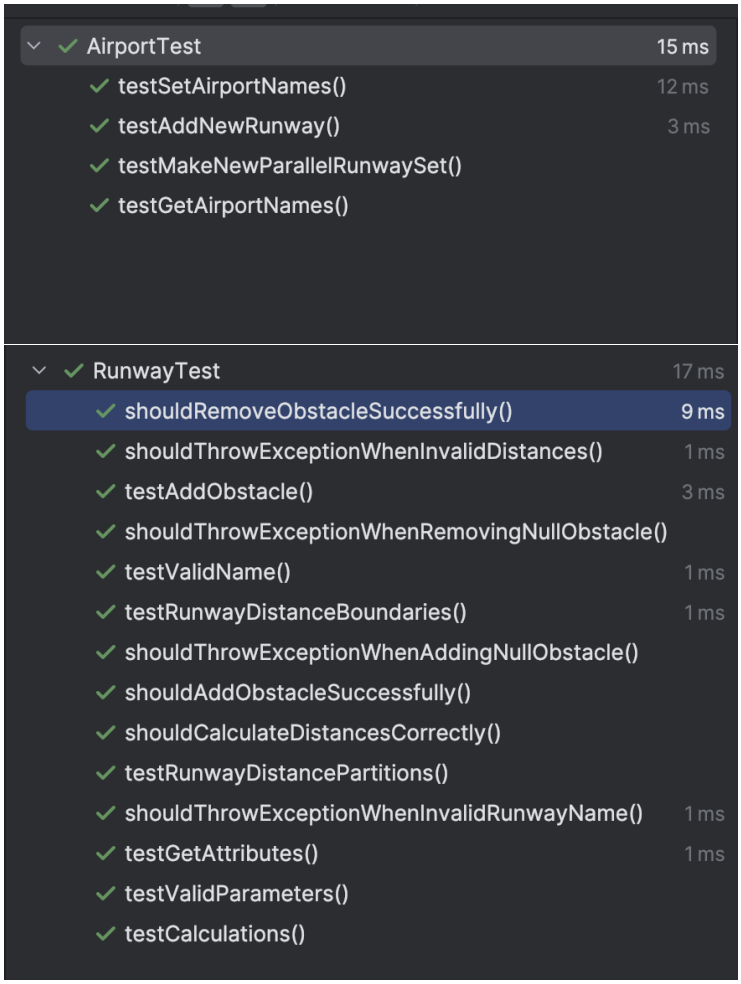
Refer section on Validation Testing and Defect testing to see passed tests. Test Result: ALL PASSED

2.1.5 Regression testing

Test Definition: Regression testing involves re-running functional and non-functional tests to make sure that previously developed and tested software still performs after a change. This ensures that when new features are added to the software application, the new changes don't affect the existing functionality.

Reason for testing: Regression testing is critical for ensuring that changes do not adversely affect the existing functionality of the software. It helps maintain the stability and quality of the software over time, especially as it evolves and grows, thereby safeguarding against unexpected side effects of modifications.

Test Examples: Runway, obstacle and airport classes were implemented in the previous increment. We ran JUnit tests to ensure they worked even with the new update:



Test Result: ALL PASSED

2.2 Acceptance Testing

Test Definition: Acceptance testing is usually the final phase of software testing where the system is tested for acceptability and to ensure that it meets all the requirements. This testing assesses whether the software is ready for release and if it fits for purpose from the user's perspective. It verifies both the functionality and usability of the application in a real-world scenario using the user stories.

Reason for testing: To determine if the application satisfies the user requirements and is ready for deployment. It's a user-focused test to verify if the system is acceptable for use.

Below are some example acceptance criteria created for tested user story. For more acceptance criteria on other user stories, please see the **Deliverable 3** report.

ID	Nickname	Acceptance Criteria
22	Export Info	1. The system must allow users to export visualizations, reports, and operational data into various formats. 2. Ensure that exported documents accurately reflect the current data and formatting as seen in the application.
23	Collab	1. Implement role-based access controls that restrict or allow functionalities according to user roles. 2. Ensure that all users see the latest data updates. 3. Use password to maintain high security standards to protect data integrity and privacy during multi-user access.
24	Error	1. The system must provide explicit and understandable error messages that guide users on how to resolve issues.
17	Clear Grade	1. Ensure that cleared and graded areas around the runway are distinctly visible in the top-down view.
18	TOC Slope	1. Accurately display the Takeoff Climb Surface (TOCS) slope over any obstacles in the side-on view. 2. Slopes should update dynamically with changes in obstacle data or runway configurations.
19	ALS Slope	1. Accurately display the Approach Landing Surface (ALS) slope over obstacles in the side-on view. 2. Slopes should update dynamically with changes in obstacle data or runway configurations.
8	Lower Threshold	1. The system should offer a comprehensive list of predefined obstacles that can be quickly added to the runway. 2. Users should find it straightforward to access and select obstacles from the list. 3. Provide options to customize or add new obstacles to the list as needed.
9	Calc Break-down	1. Provide a clear and detailed breakdown of runway distance calculations. 2. Ensure the accuracy of calculations
10	XML Import	1. System support importing details from XML files regarding obstacles and airport data.
11	XML Export	1. Allow users to export details of obstacles and airport data to XML files. 2. Ensure that exported XML files accurately reflect the current system data.
12	Side-View Display	1. The side-on view displays detailed information about the runway, including all necessary indicators and designators. 2. Information displayed must be accurate and clearly visible. 3. Updates to runway or obstacle information should be reflected immediately in the side-view display.
13	Top-View Display	1. Display detailed and accurate information in the top-down view, including the runway centerline. 2. Information displayed must be accurate and clearly visible. 3. Updates to runway or obstacle information should be reflected immediately in the Top-View display.
5	Both Views	1. The system must be able to display both 2D top-down and side-on views of the airport on the same screen or interface. 2. Ensure that both views are synchronized in real-time, showing up-to-date and consistent information across both visualizations. 3. Both views must be clear and accurately represent the airport's layout and current operational status
6	Auto Calc	1. When an obstacle is added or its specifics are changed, the system should automatically recalculate and update available runway distances. 2. The recalculated runway distances should be displayed in real-time 3. Ensure that all calculations are accurate

Table 1: User Stories Acceptance Criteria

Test Examples:

Example : Testing User Story 22 (Export Info)

As an *operational staff,* ***I want*** *the ability to export visualisations, reports, and operations in formats like PDF,* ***so that*** *I can distribute and archive data for stakeholders and regulatory compliance.*

Acceptance Criteria:

1. The system must allow users to export visualizations, reports, and operational data into various formats.
2. Ensure that exported documents accurately reflect the current data and formatting as seen in the application.

Test Cases:

1. **Test Case 1: Export Functionality Test**
   Objective: To verify that the system allows users to export visualizations, reports, and operational data in various format.
   Steps:
   (a) Navigate to the page where visualisations, calculation reports and user displayed.
   (b) Select the option export.
   (c) Choose where to store file on local computer
   (d) Complete exportation

   Expected Outcome: Files can be exported.
   Actual Outcome: Files was exported.

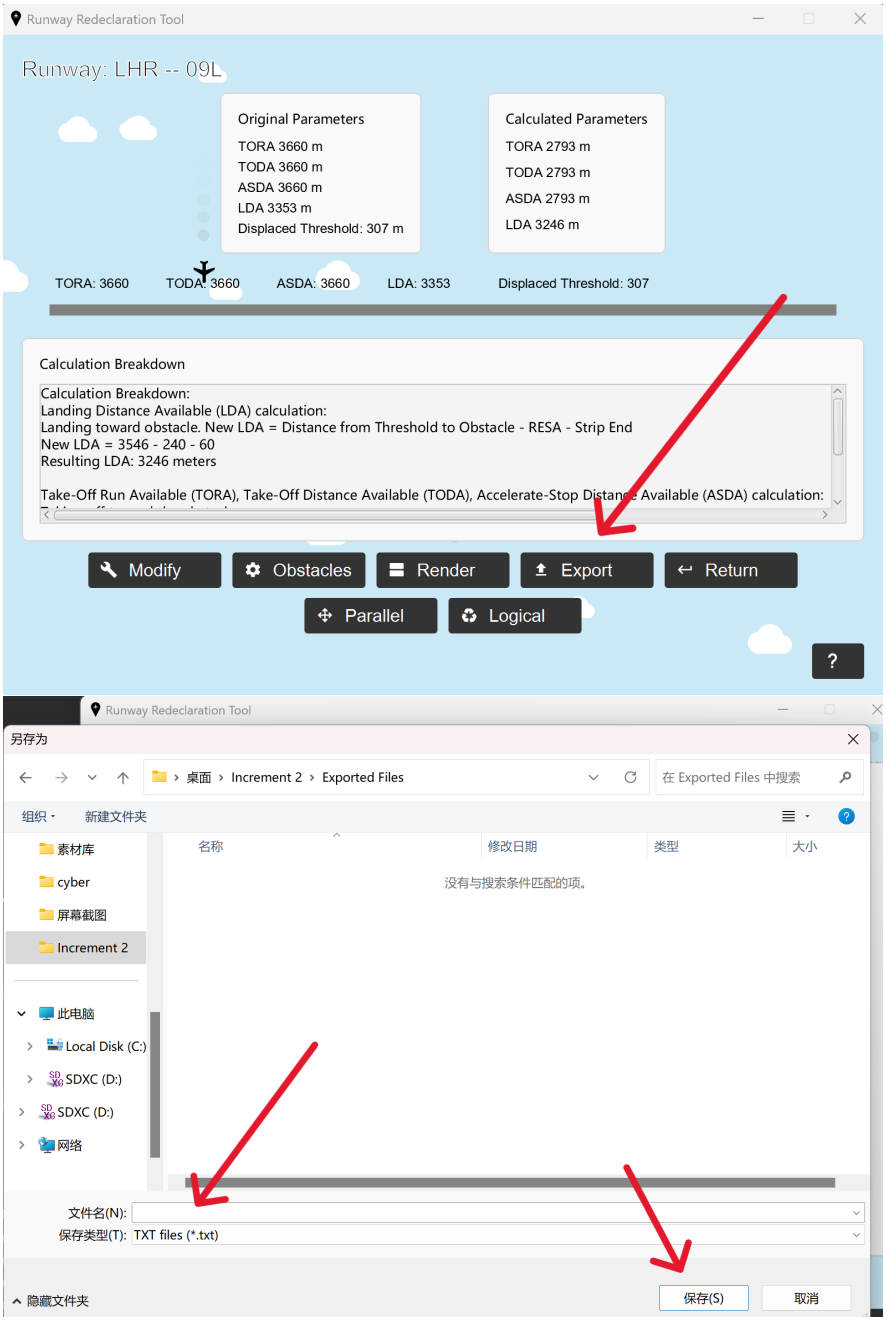


Figure 21: Evidence for test case 1

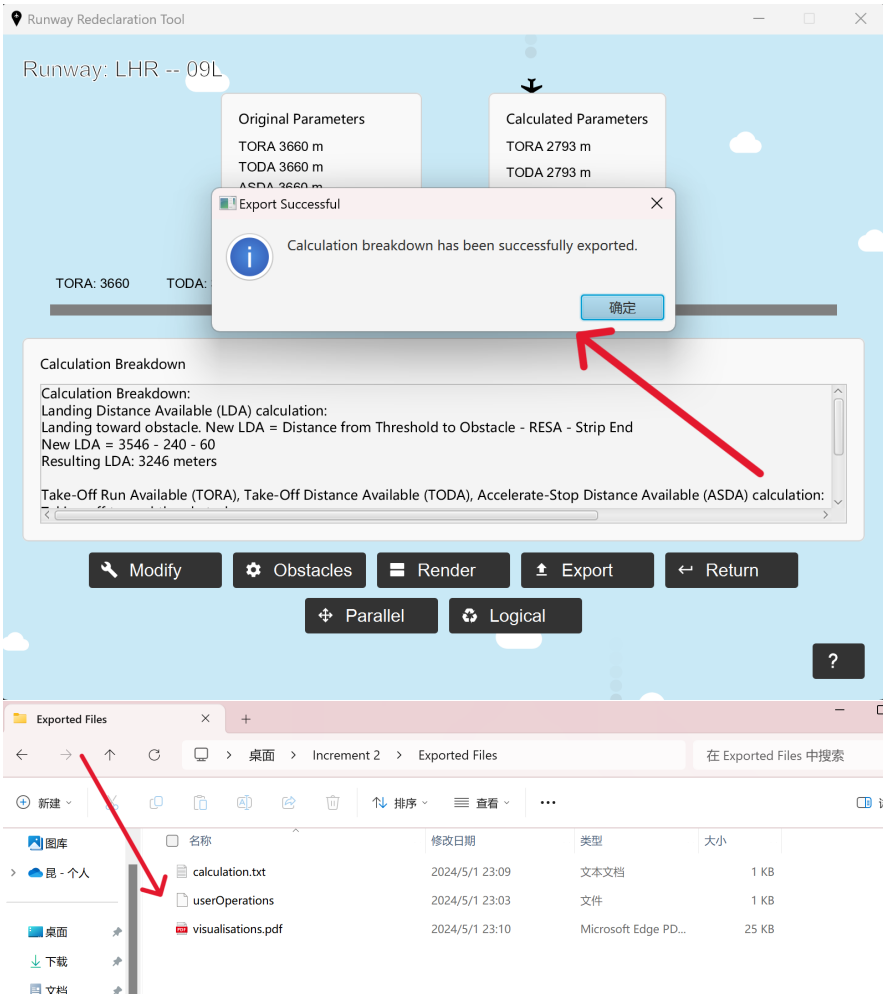


Figure 22: Evidence for test case 1

Test Result: PASSED

2. **Test Case 2: Accuracy of Exported Files**
Objective: To ensure that the exported document accurately reflects all current data.
Steps:
   (a) Open the exported files.
   (b) Compare with information displayed in the application.

Expected Outcome: Both exported content and displayed information are the same.
Actual Outcome: Both exported content and displayed information are the same.

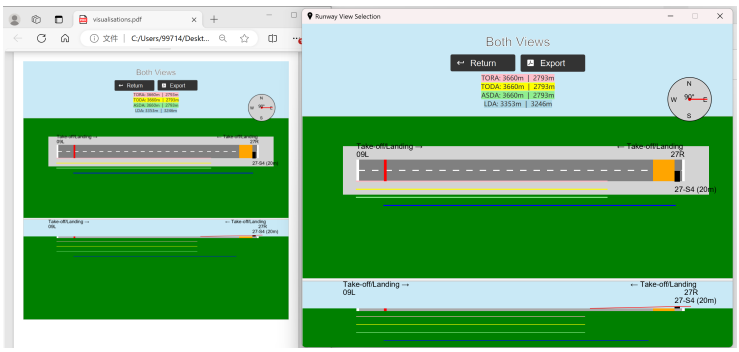


Figure 23: Evidence for test case 2: Export content matches with information in app

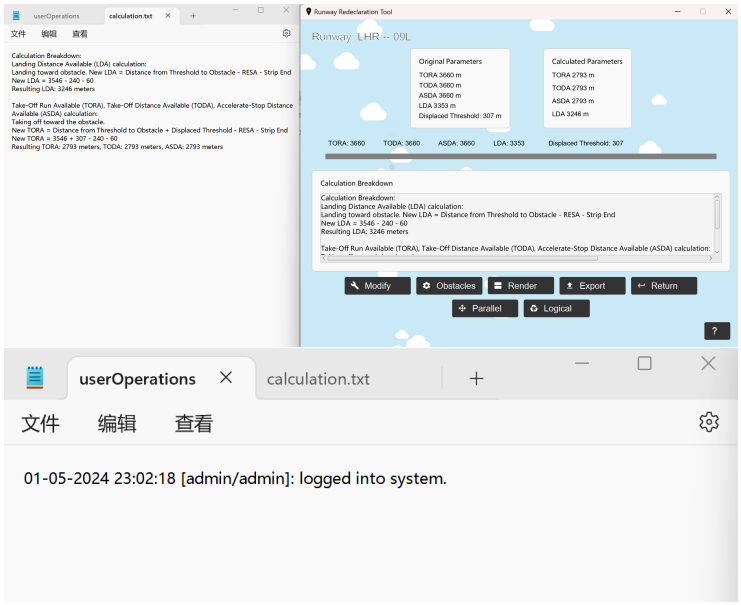


Figure 24: Evidence for test case 2: Export content matches with information in app

Test Result: PASSED

Test Result: ALL PASSED

2.3 Scenario testing

Test Definition: Scenario testing involves using realistic user scenarios to test the software's functionality. This type of testing is based on hypothetical stories to help testers think through a complex problem or system. It goes beyond traditional test cases by considering more complex and nuanced user behaviours.

Reason for testing: To ensure that the software functions correctly in realistic user scenarios, especially in complex or unusual situations. It also helps to understand how different components of the application interact in various scenarios.

For this increment we have finished testing scenario 1 and 2. We are planning to finish testing the rest of our scenarios in the next increment. Please see below for our test.

Test Examples:

Example 1: Testing Scenario 3 - Stacey (Airfield Safety Officer)

Objective: To validate the functionality of obstacle management and parameter recalculation for a runway as performed by an airfield safety officer, Stacey.

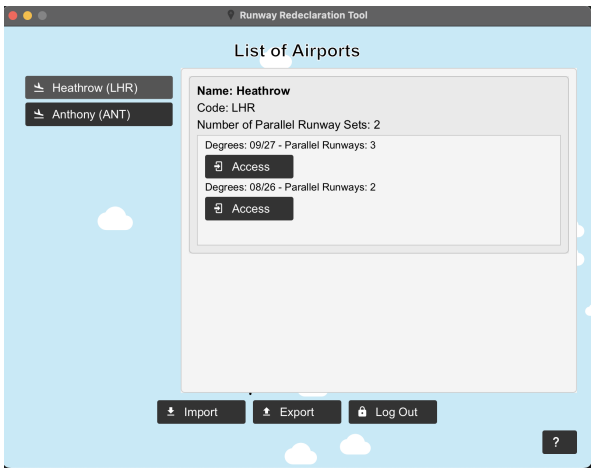
Test Steps:

1. Launch the runway re-declaration software.
2. logs in with her valid username and password.
3. navigates to the " LHR " airport and clicks the "Select" button.

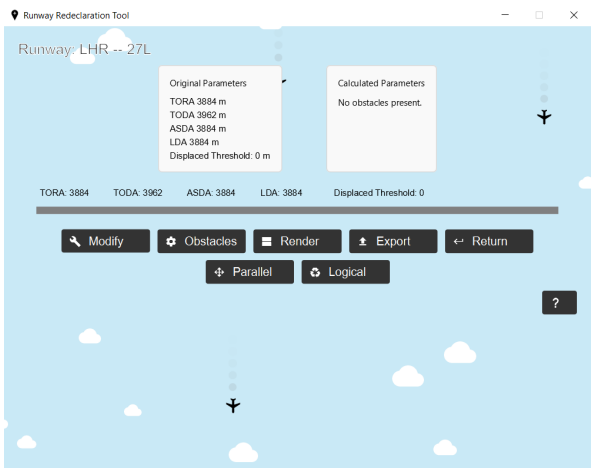
4. directed to the runway list page displaying runway 27L among others.
5. selects runway 27L and clicks "Select."
6. Verify that the runway configuration page is displayed, showing current settings for runway 27L.
7. navigates to the "Obstacle" section and selects the "Car" obstacle, moving it to the current obstacle configuration.
8. clicks "Return," and a prompt appears requesting the input of the Blast Protection Value.
9. Input a valid Blast Protection Value and verify that the system accepts the input and displays the recalculated runway parameters.
10. Intentionally input an invalid (negative) Blast Protection Value to test error handling.
11. Verify that an error message "Invalid measurements for Blast Protection Value" is displayed.
12. returns to the Obstacle list, selects the "Airplane" obstacle, and clicks "Add" to swap it with the "Car" obstacle.
13. Verify that the obstacles swap correctly without any issues.
14. inputs a valid Blast Protection Value again.
15. Confirm that the system recalculates and displays the new parameters side by side with the default parameters correctly.

Expected Outcome: Each step should occur without errors, and the system should react as described in the steps. The software should handle logins and navigate as expected. The correct airport and runway data should be accessible and selectable. Obstacles should be manageable with accurate updates and feedback on user actions. Error handling should work correctly by displaying appropriate messages for invalid inputs. Recalculations based on obstacle changes should update runway parameters accurately and display them as expected.

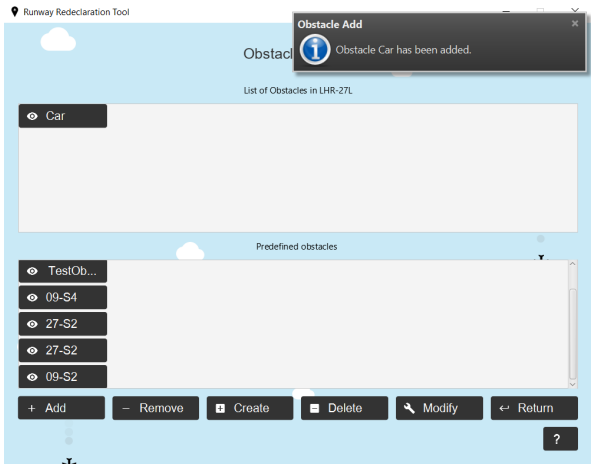
Actual Outcome: The test was carried out without errors, and the system reacted as described in the steps. The result meets all criteria stated in the expected outcome.



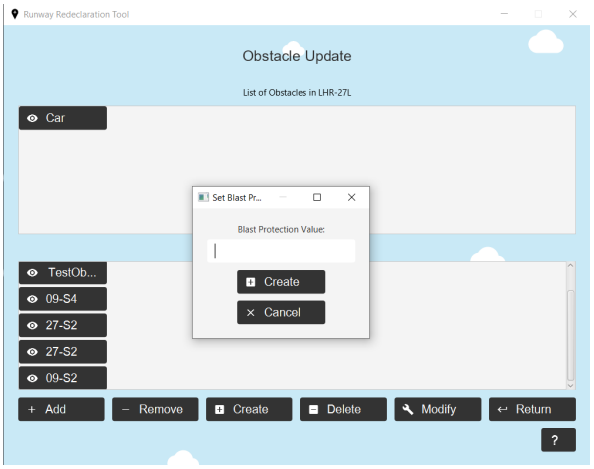
(a) Navigating to "LHR" airport.



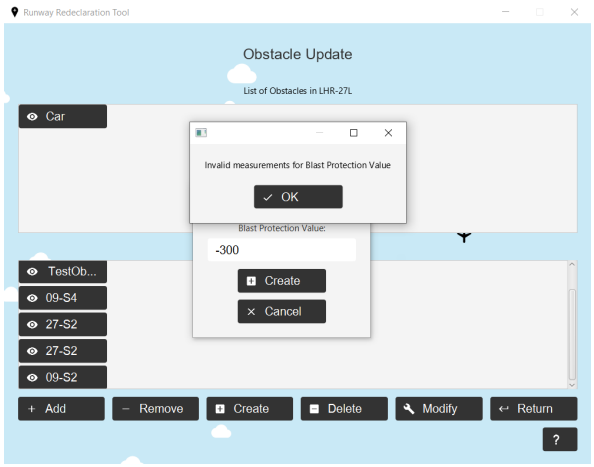
(b) Configuration scene for runway 27L.



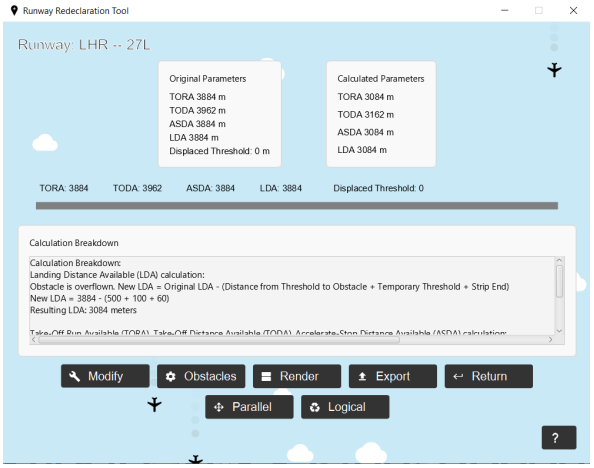
(a) Car added to obstacle configuration.



(b) Returning requires a BPV to be inputted.



(a) Invalid BPV shows the error message.



(b) Calculated values now displayed.

Test Result: PASSED

Example 2: Testing Scenario 4 - James (Air Traffic Controller)

Objective: To test the import functionality of the runway re-declaration software and the subsequent navigation through the airport and runway data, including viewing specific visualisations.

Test Steps:

(a) Launch the runway re-declaration software.

(b) enters username and password.

(c) Verify successful login and display of the airport database screen.

(d) Click the "List" button on the airport database screen.

(e) Verify that a page displaying the list of available airports is shown.

(f) Click the "Import" button.

(g) Verify that the system opens the file directory for file selection.

(h) Navigate to the location of "airport_data.xml".

(i) Select the file and confirm the selection to start the import process.

(j) Verify that the airport list is updated to include airports from the imported file, focusing on checking for Heathrow Airport.

(k) From the updated list, select Heathrow Airport and proceed.

(l) Verify that a runway list page is presented showing all runways associated with Heathrow Airport.

(m) Select a specific runway from the list.

(n) Verify that detailed information about the selected runway is displayed.

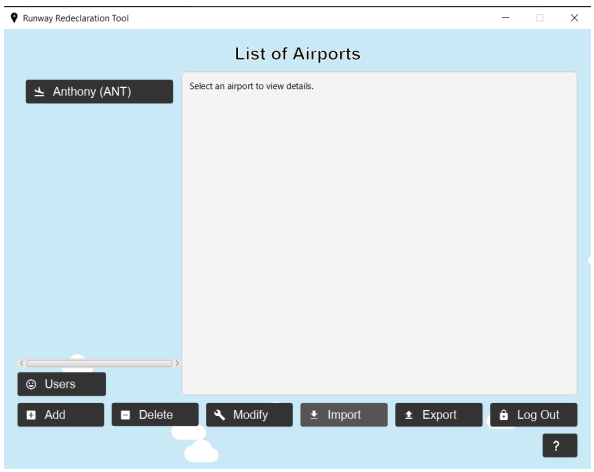
(o) Navigate to 'Visualisations -¿ Side View'.

(p) Verify that the side view visualization of the selected runway is correctly displayed.

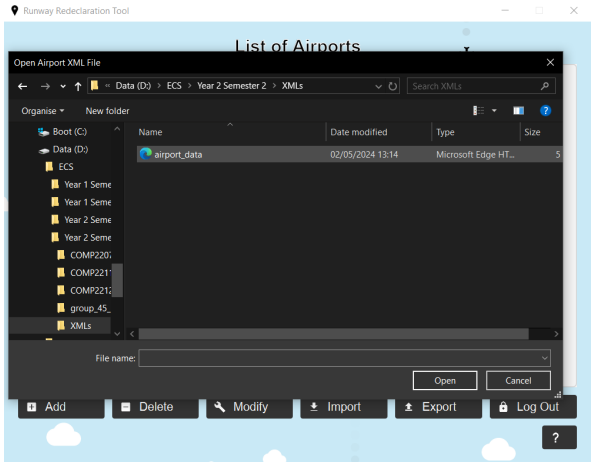
Expected Outcome:

- Each step performs as described without errors, and all system responses match the expected outcomes.
- The XML import function correctly populates the airport database.
- Visualizations accurately represent the selected runway's current configuration.

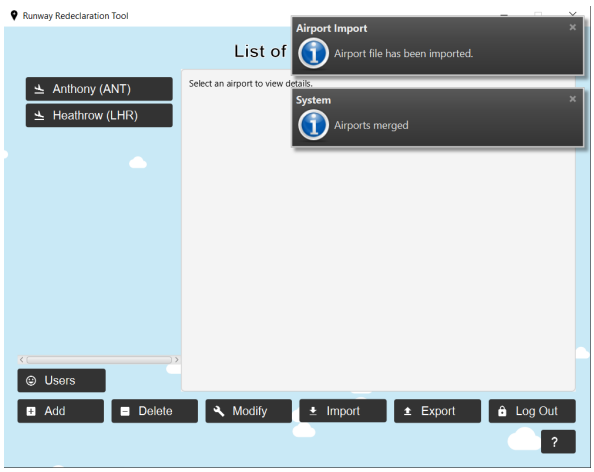
Actual Outcome: The system reacted as described in the steps. The XML import function correctly populates the airport database. Visualizations accurately represent the selected runway's current configuration.



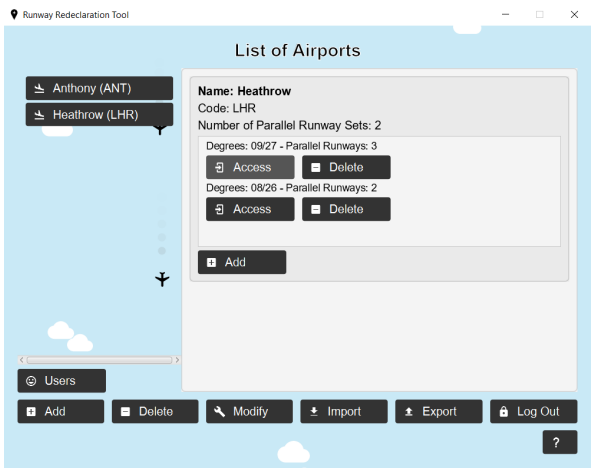
(a) James goes on the import button.



(b) Selecting a local xml file to import.



(a) Heathrow airport added.



(b) Go to Heathrow airport's runways.

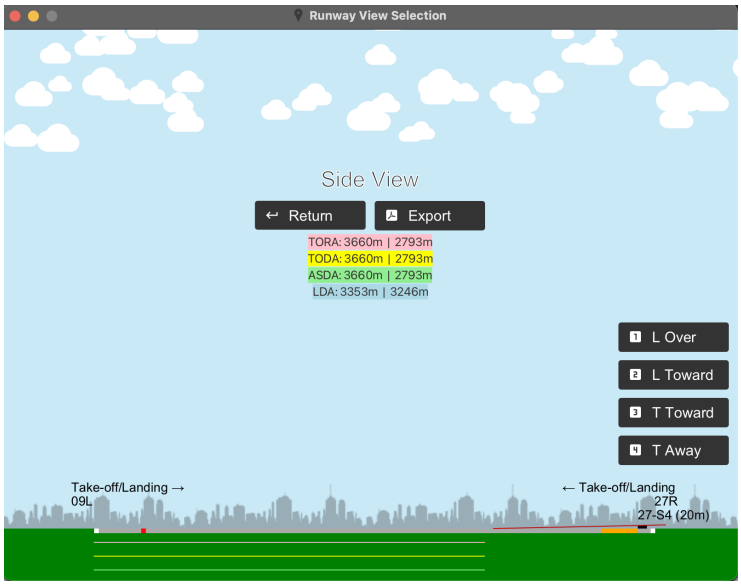


Figure 30: Access side-view of runway 27R example.

Test Result: PASSED

Example 2: Testing Scenario 5 - Andrew (Airfield Safety Officer)

Objective: To ensure that the system allows an Airfield Safety Officer to create and configure an obstacle on a runway and validate the input of blast protection values while checking for errors in data entry.

Test Steps:

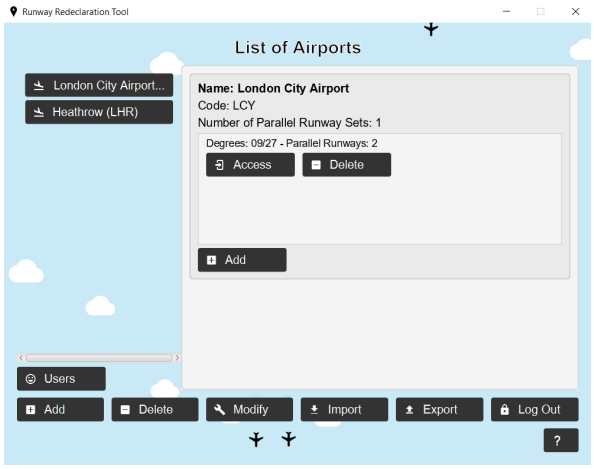
1. Open the runway re-declaration software.
2. Login with Andrew's username and password.
3. Verify that the login is successful and the Airport Database page is displayed.
4. Click on the airport "LCY" from the list of airports.
5. Press the 'Select' button to open the list of runways.
6. Click on runway "09R" and press 'Select' to view its configurations.
7. Access the Obstacle Management section by clicking the 'Obstacle' button.
8. Click on 'Create' to open the obstacle creation form.
9. Enter the details for a new obstacle:
    - Name: "Broken down airplane"
    - Height: 25 meters
    - Distance from threshold: 500 meters
    - Distance from centreline: 50 meters
10. Confirm the data entry by clicking 'Create'.
11. Verify that no error messages are displayed and the new obstacle is listed.
12. Click 'Return' to proceed to the next step, where blast protection values are requested.

13. Input a valid blast protection value (e.g., 300 meters).
14. Verify that the input is accepted without errors.
15. Check the display of old and new runway parameters side by side along with a breakdown of the calculations.

Expected Outcome:

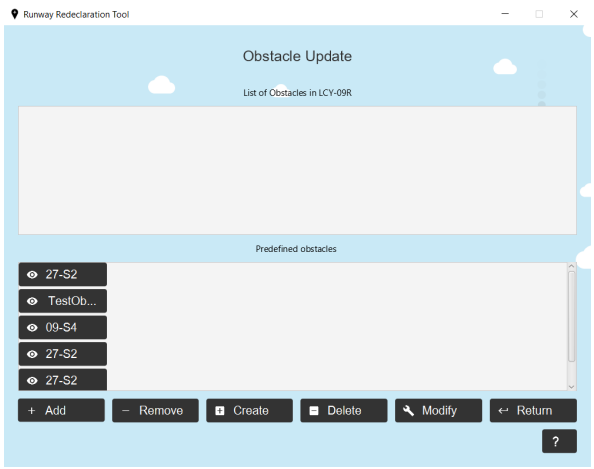
- Each step functions as expected, and all user inputs are processed correctly.
- Error messages for invalid inputs should display as appropriate, preventing the user from proceeding with incorrect data.
- After all correct inputs, the system should display a side-by-side comparison of old and new parameters with detailed calculations, confirming the system's responsiveness and accuracy in handling runway safety operations.

Actual Outcome: The system reacted as described in the steps. Error messages are displayed as appropriate. After all correct inputs, the system displayed a side-by-side comparison of old and new parameters with detailed calculations.

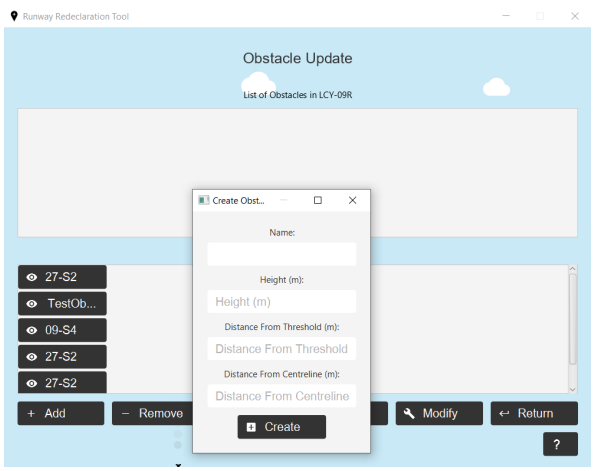


(a) Accessing LCY airport.

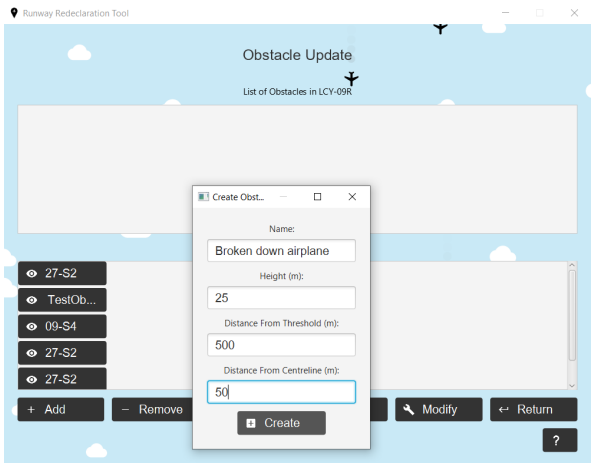
(b) Configuration screen for 09R runway.



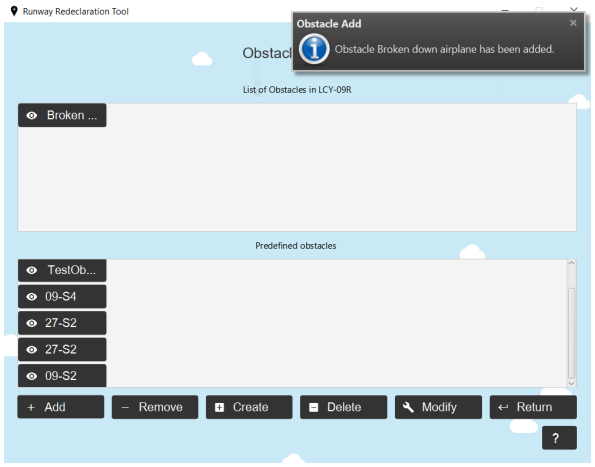
(a) Configuration screen for 09R runway.



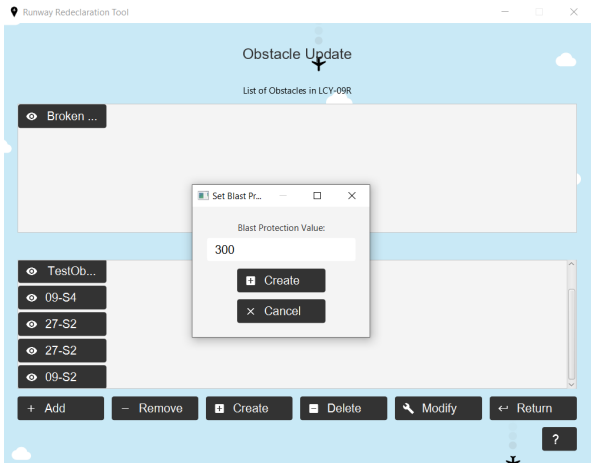
(b) New prompt for new obstacle.



(a) Entered the details of the broken airplane.



(b) Add the broken airplane to the runway.



(a) Add the BPV of this new obstacle.

(b) Shows new calculations.

Test Result: PASSED

3 Planning

3.1 Completed Sprint Plan for Deliverable 4

The following sprint plan outlines the tasks, owners, estimated hours, actual hours and priorities that are completed for Deliverable 4.

User Story	Sprint Backlog (Tasks)	Member	Est. Hours	Actual Hours
11: XML Export	1. creates the option to export data in XML files.	Anthony	4	4
17: Clear Grade	1. makes the top-down view scene display the clear and grade area around the runway strip.	Zagrosi	3	3
18: TOC Slope	1. displays the TOCS (Take—Off Climb Surface) slope caused over the obstacle when one is present in the side-on view scene.	Manlin	2	2
19: ALS Slope	1. displays the ALS (Approach / Landing Surface) slope caused over the obstacle when one is present in the side-on-view scene.	Manlin	2	2
20: Select Runway	1. creates an option to select different runways, with the views updating upon their selection.	Bav	3	3
21: Export Info	1. creates the option to export visualisations, reports and user operations in various formats, such as PDF.	Yash, Manlin	5	5
23: Collab	1. Implement user authentication and authorization features to allow multiple users (agency team members) to collaborate on the same dashboard. 2. Add user roles (e.g., admin, editor, viewer) with different levels of access to the data and functionalities. 3. All user information, including usernames, passwords, and permission levels, should be stored in the database	Zagrosi,Anthony	8	7
26: Help Guide	1. create a user help guide to help users learn how to use the system 2. The user guide should be user-centred, address frequently asked questions (FAQs) and outline known issues. 3. Create a help button within the app to view the digital help guide	Bav,Yash	5	5

Table 2: Completed Sprint Plan for Deliverable 4

3.2 Completed Burn-Down Chart for Deliverable 4

The burn-down chart is modified to include additional user stories we have decided to add and implement during increment 3 (deliverable 4).

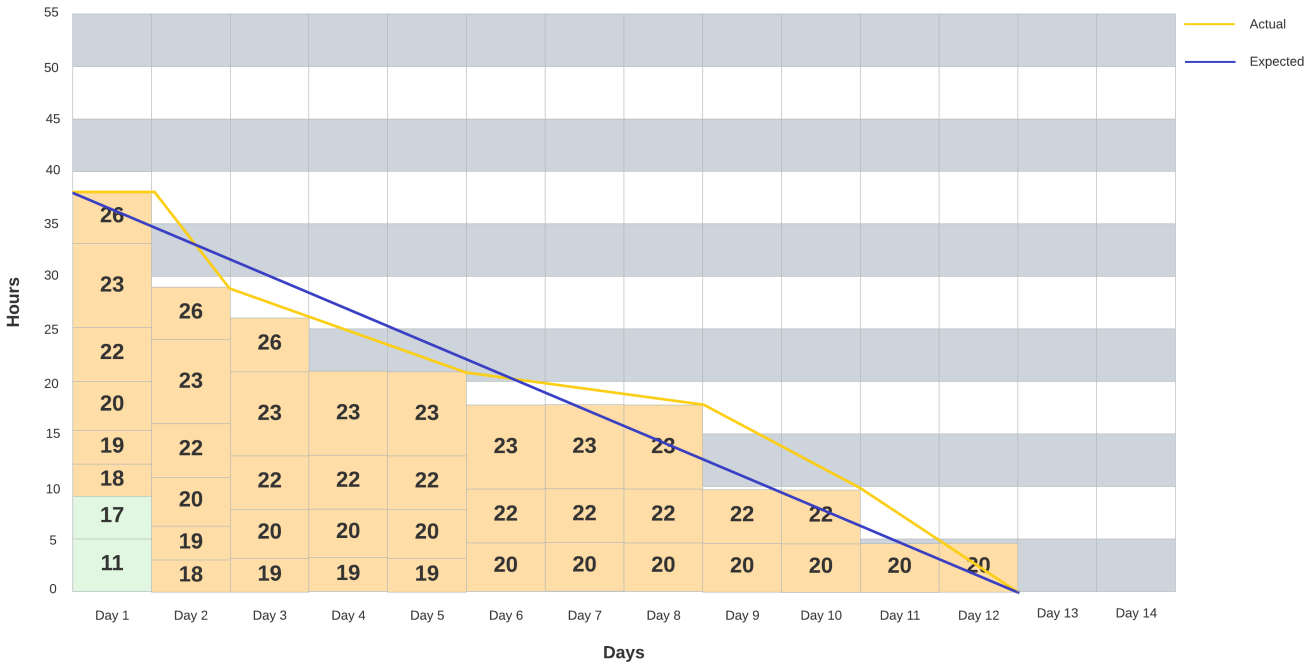


Figure 35: Completed Burn-down Chart for Increment 3 (Deliverable 4)

4 Response to feedback in Deliverable 3

Our feedback for improvement from Deliverable 3:

- "Extend or simplify existing features or any changes that reduce the amount of work for users."
- The Collab user story extends to now have a full account database, instead of only limited to a single account from previous deliverables.
- It restricts certain accounts' roles such as editor and viewer. This will give them less functionalities to worry about and learn to use.
- It's up for the admins to configure and setup new user's accounts for them.

5 Appendix

5.1 Product Backlog

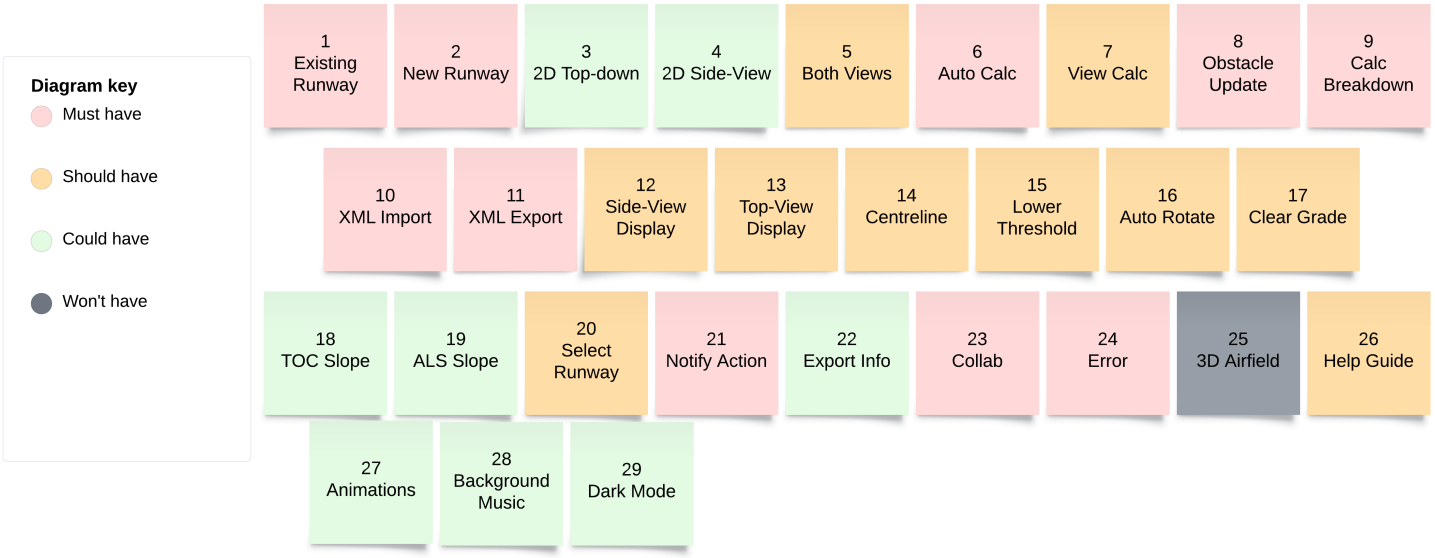


Figure 36: Product Backlog

5.2 Sprint Plans

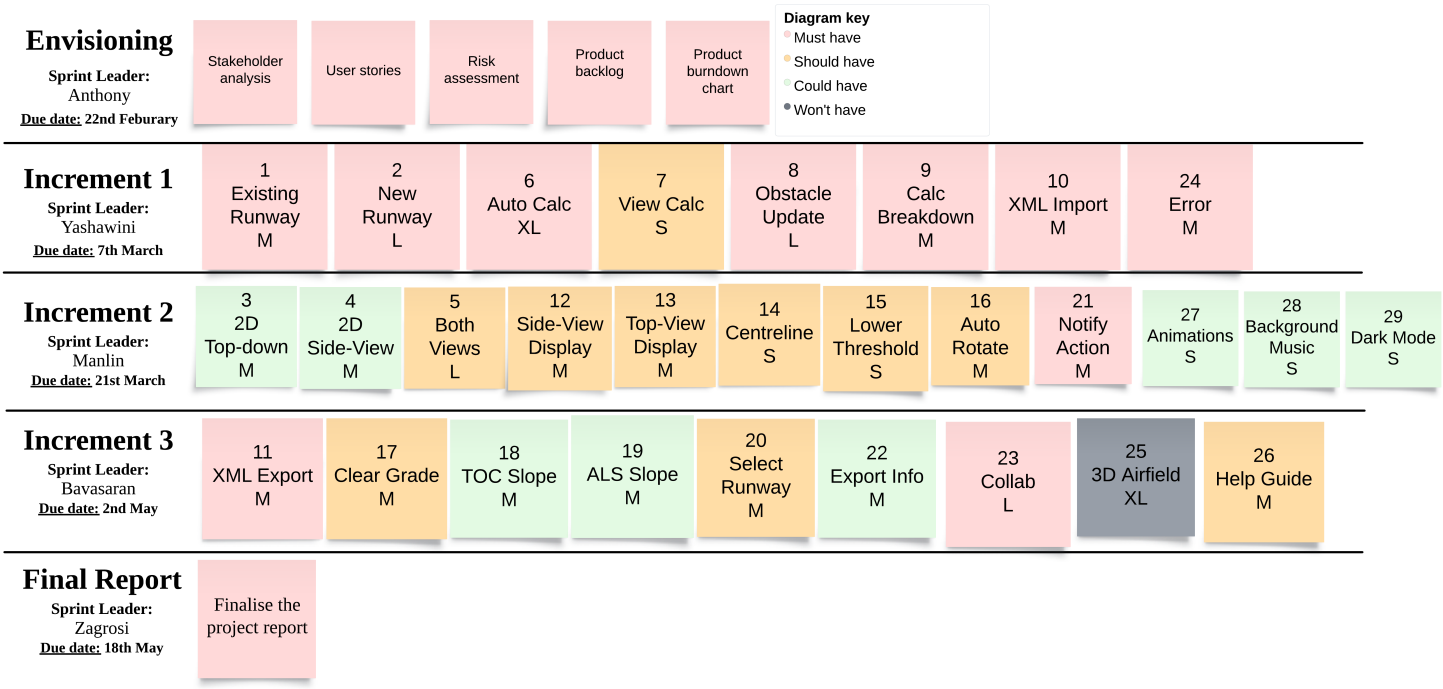


Figure 37: Sprint Plans

5.3 State Machine Diagram

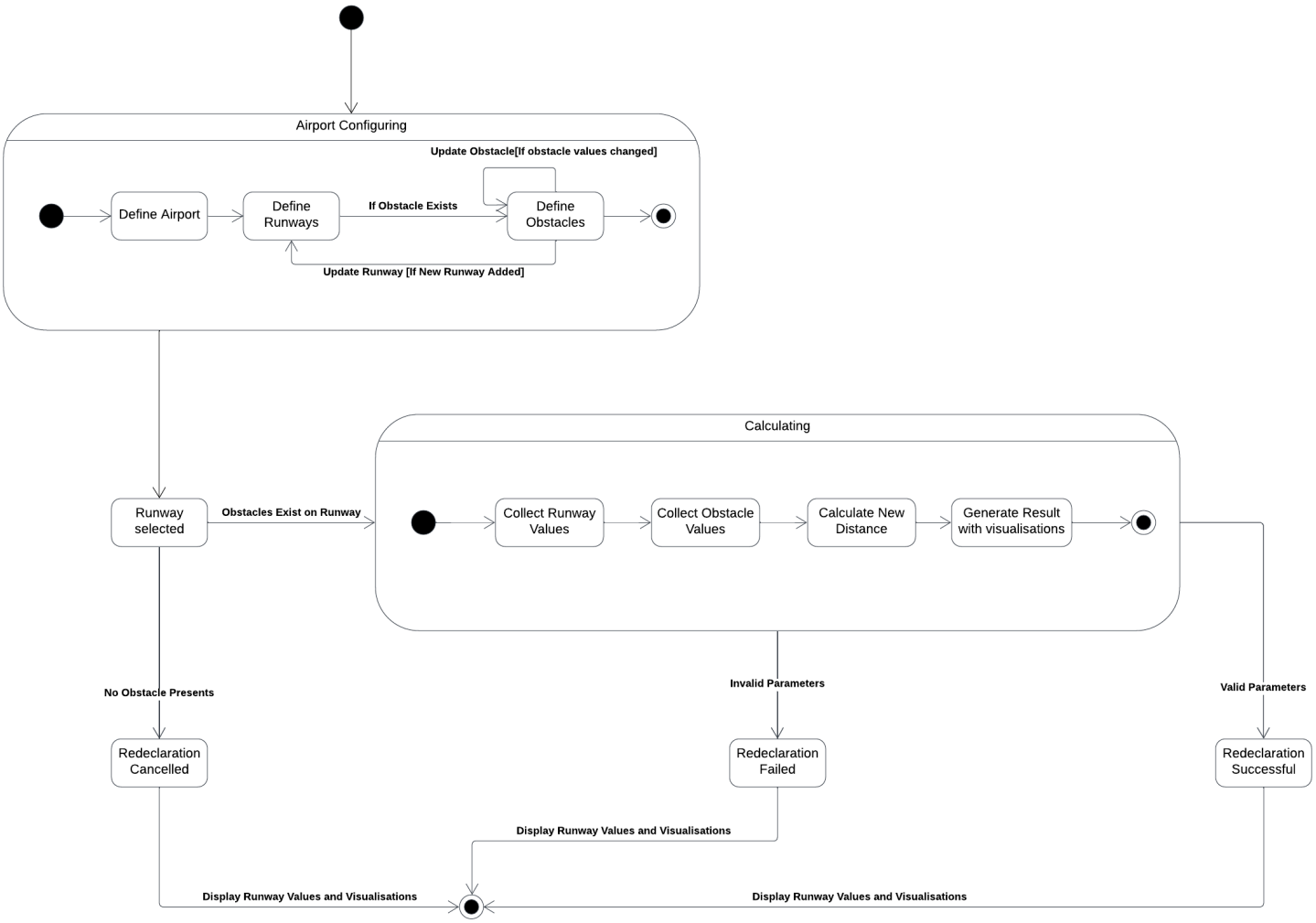


Figure 38: State Machine Diagram

5.4 Use Case Diagram

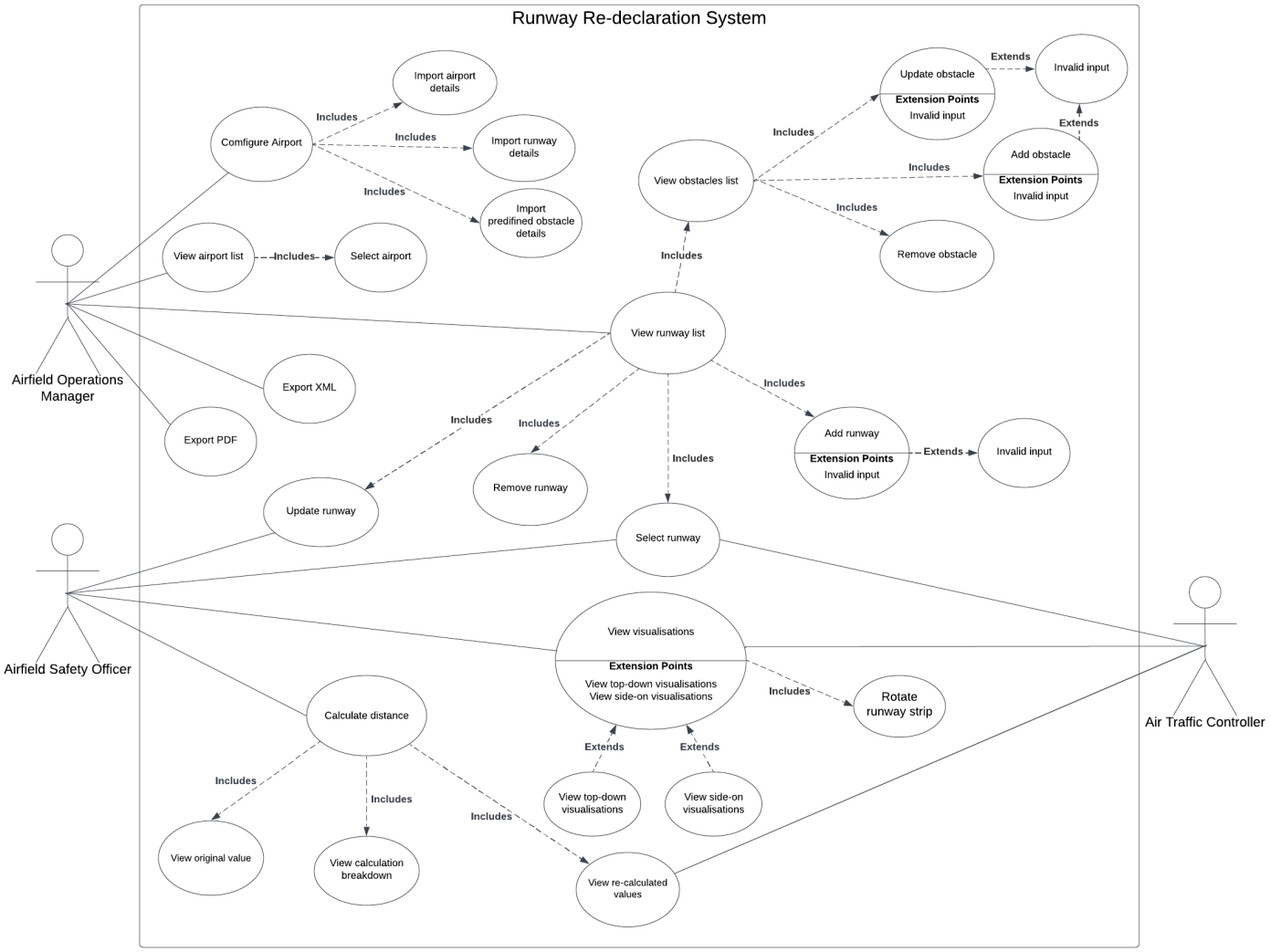


Figure 39: Use Case Diagram

Bibliography

References

[1] Creative fabrica - plane taking off graphic, 2023.