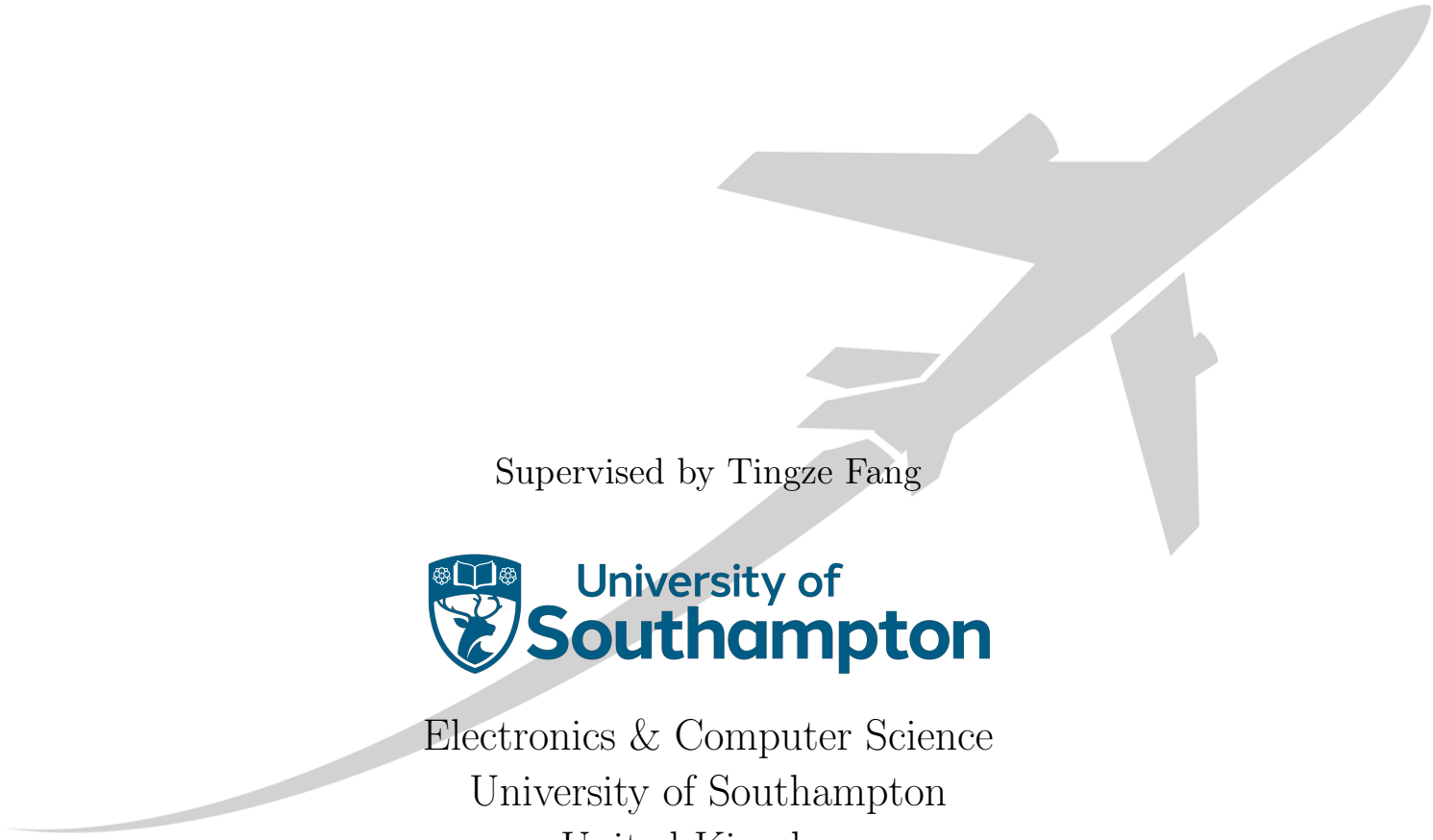
COMP2211
Software Engineering Group Project

Runway Re-declaration
Deliverable 3: Increment 2

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 3: Increment 2

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

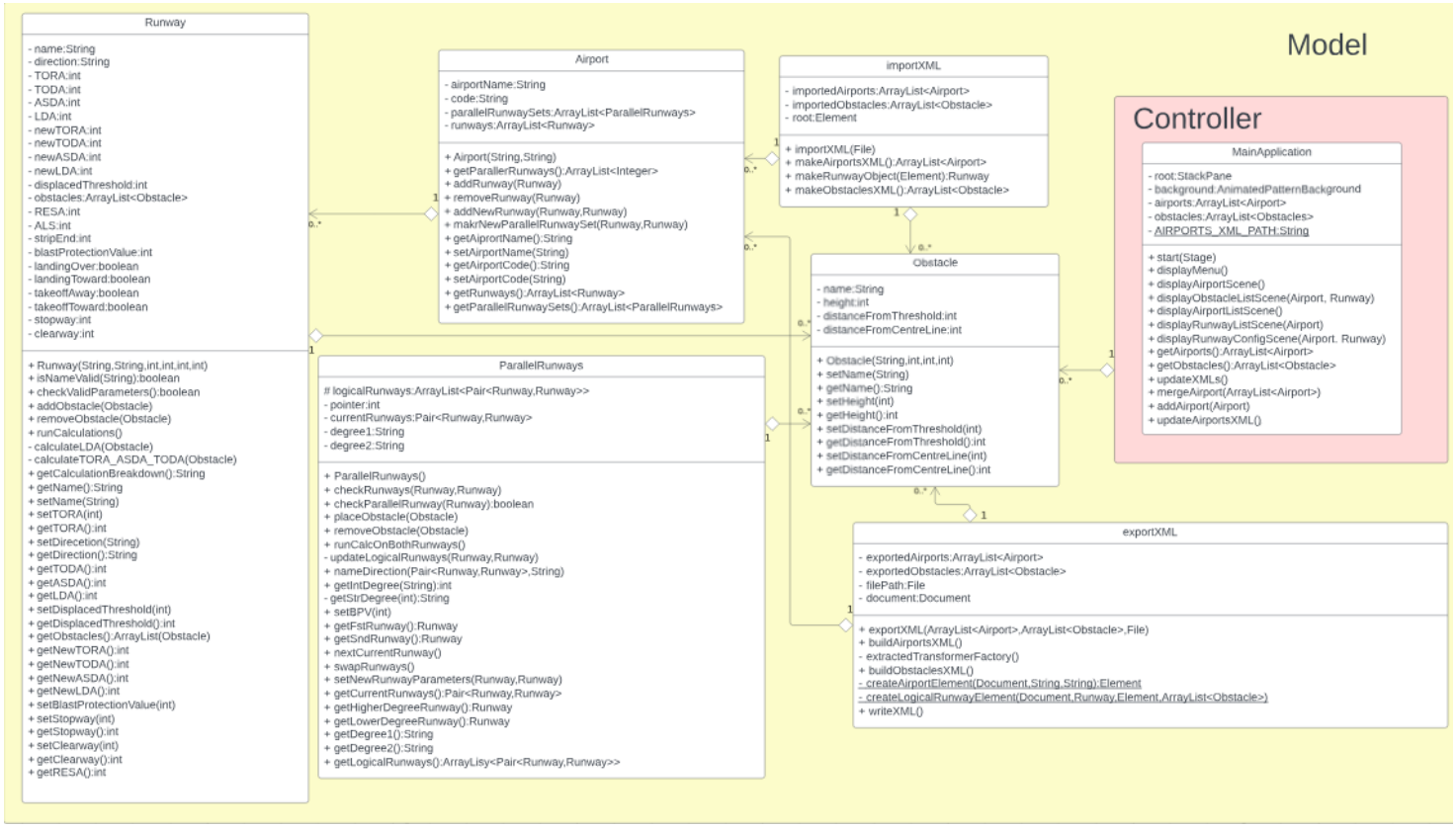


Figure 1: Model Controller class diagram

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.

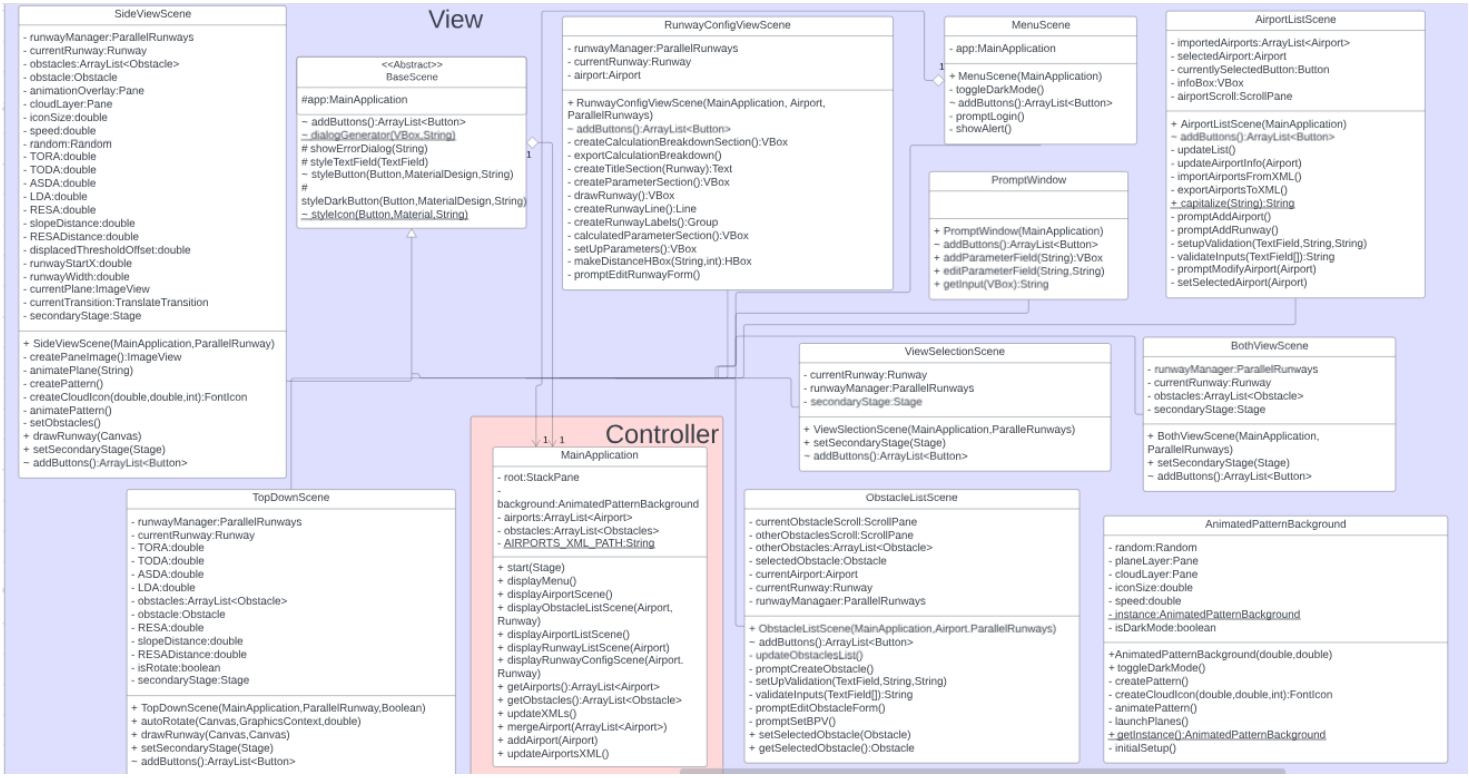


Figure 2: View Controller class diagram

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**.

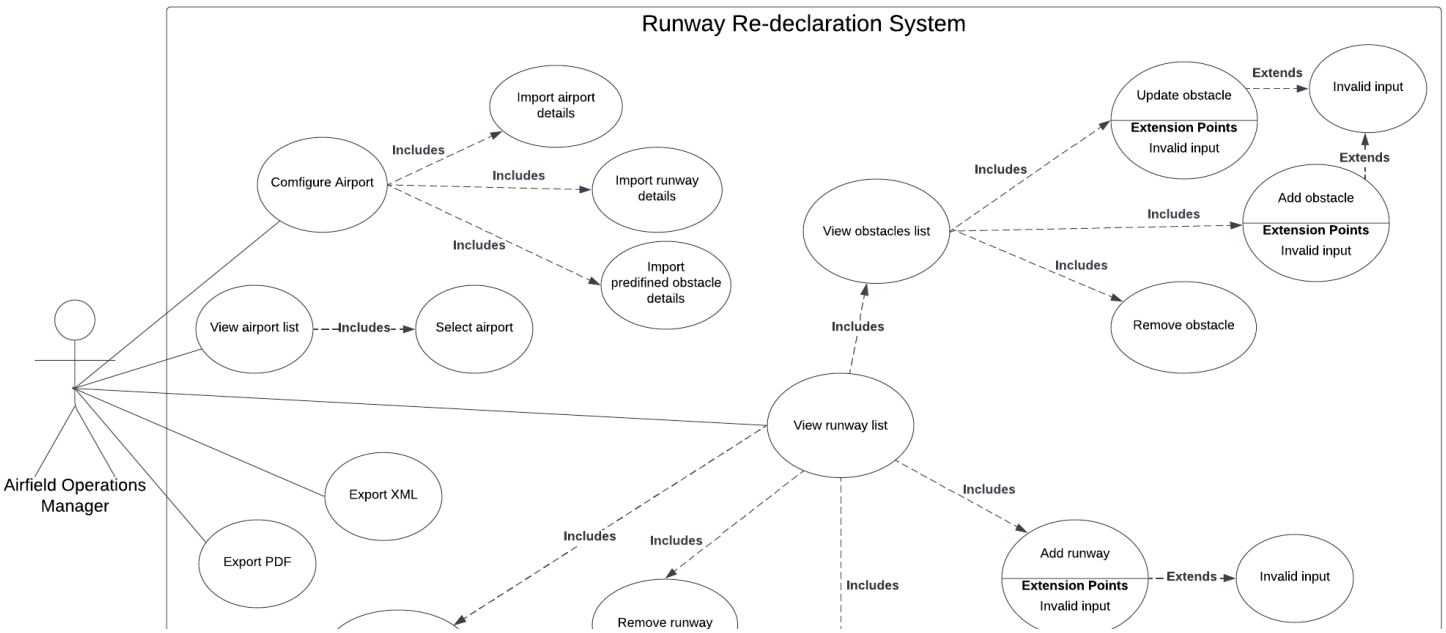


Figure 3: Use Case Diagram Part 1

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.

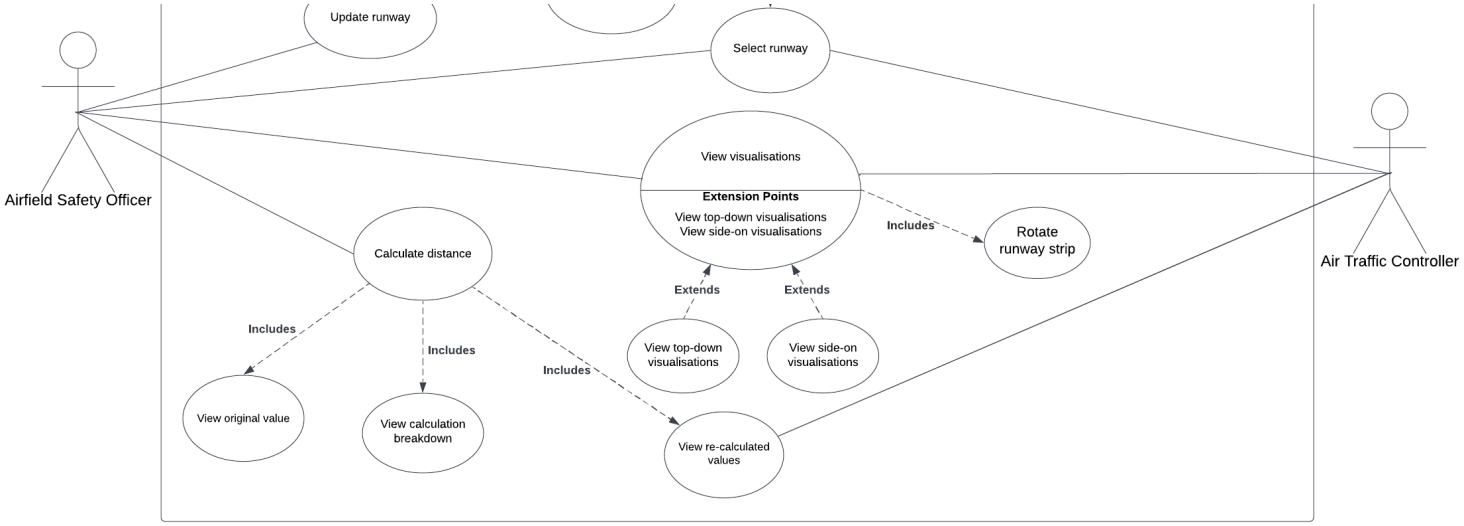


Figure 4: Use Case Diagram Part 2

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.

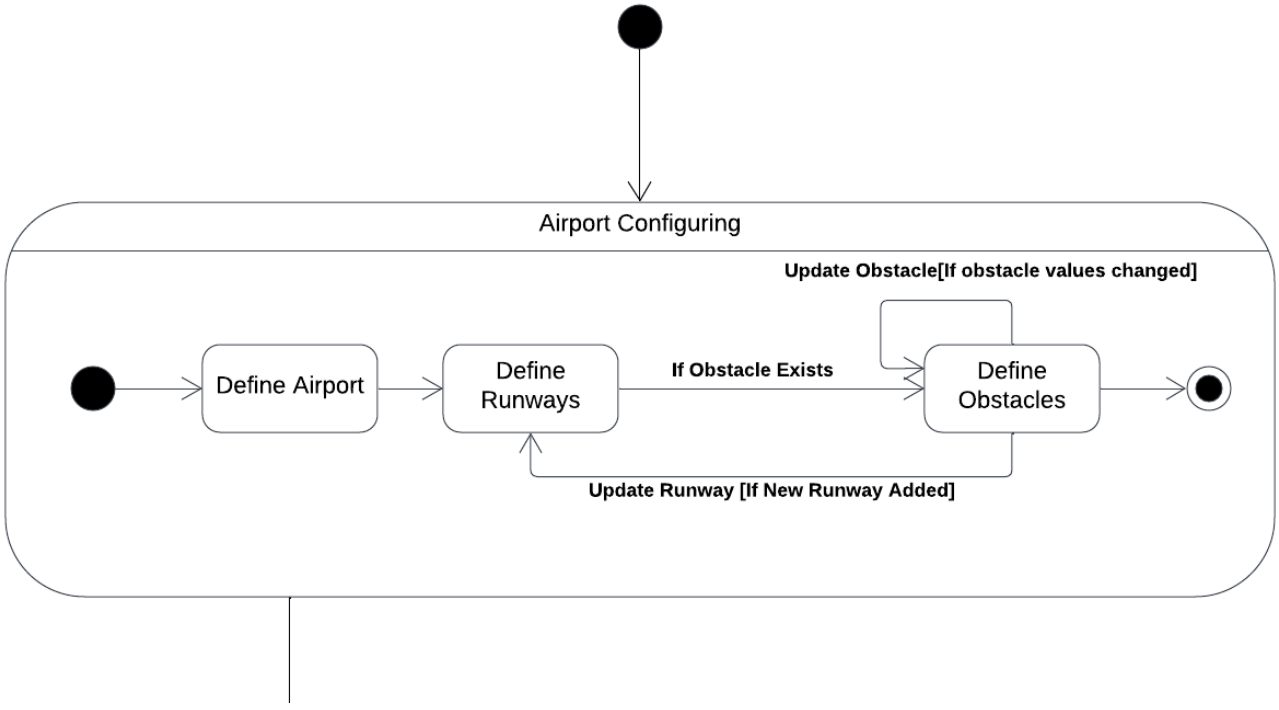


Figure 5: State Machine Diagram Part 1

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

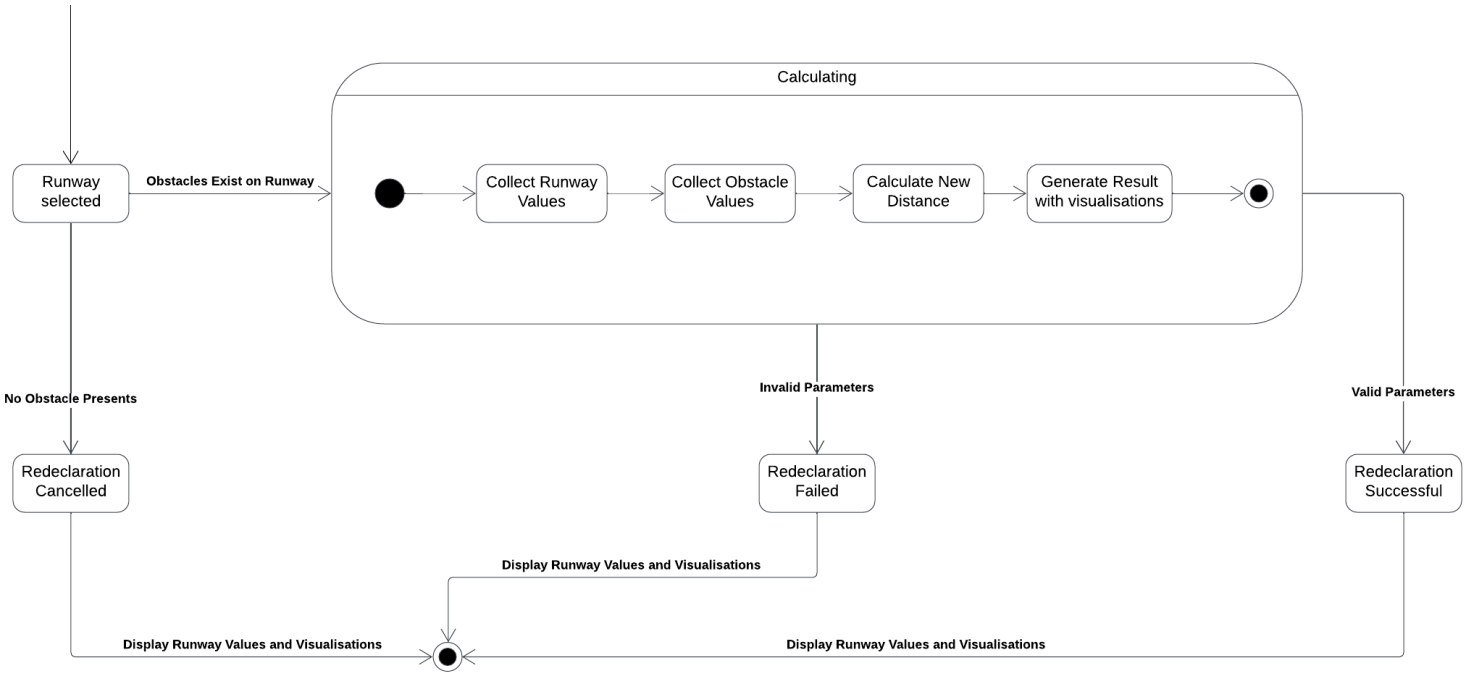


Figure 6: State Machine Diagram Part 2

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.

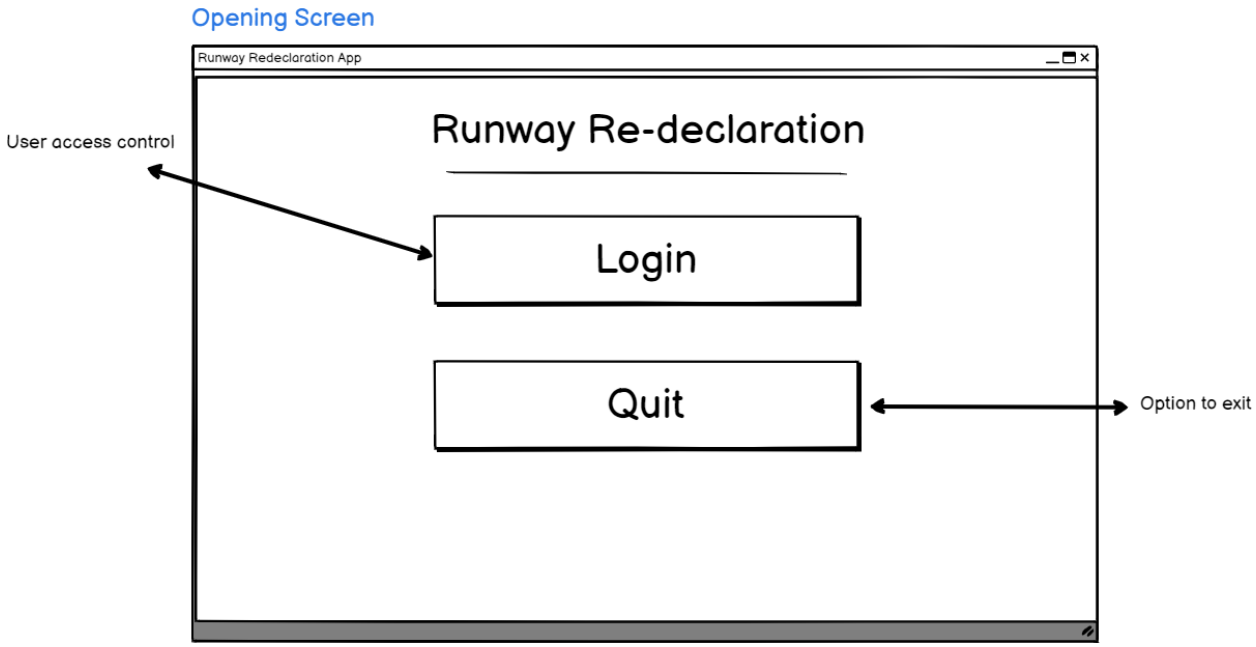


Figure 7: Login page 1 - User can log in to the system by clicking on the 'Login' button

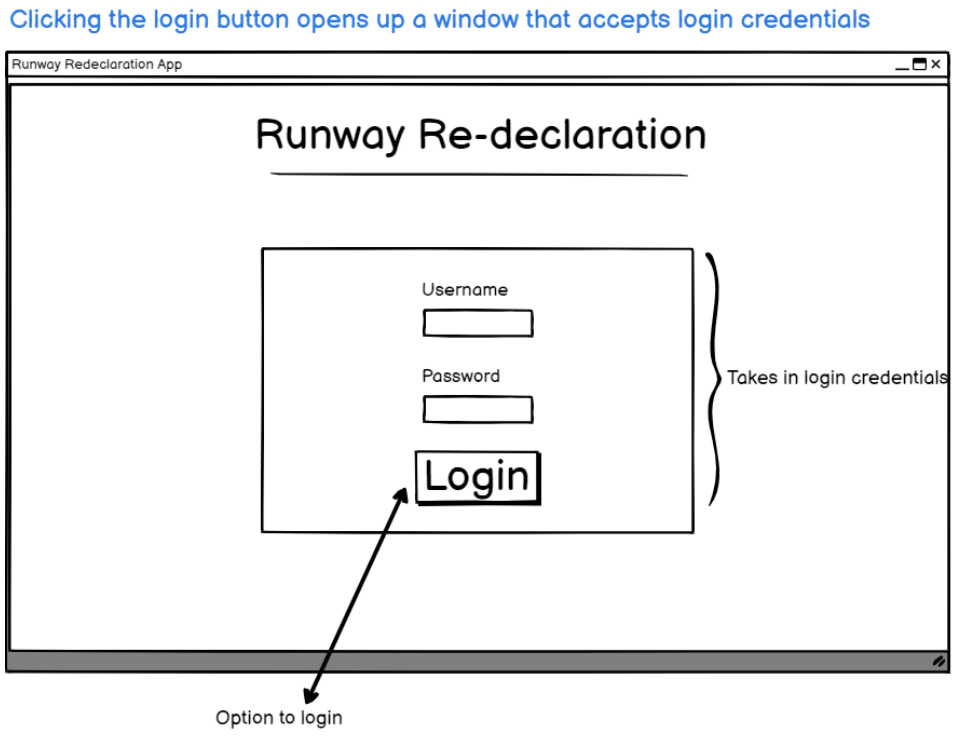


Figure 8: Login page 2 - Only correct credentials will be accepted

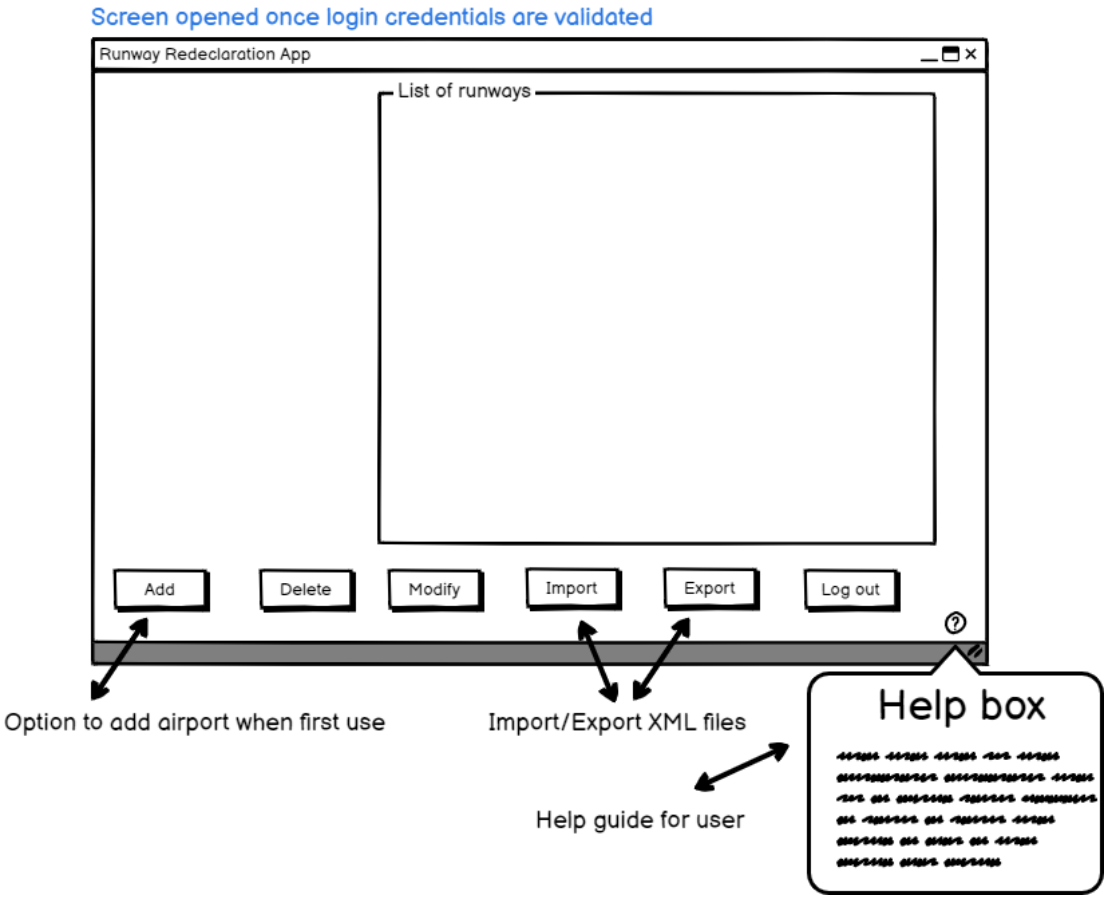


Figure 9: Empty Airport page - user needs to configure airport first

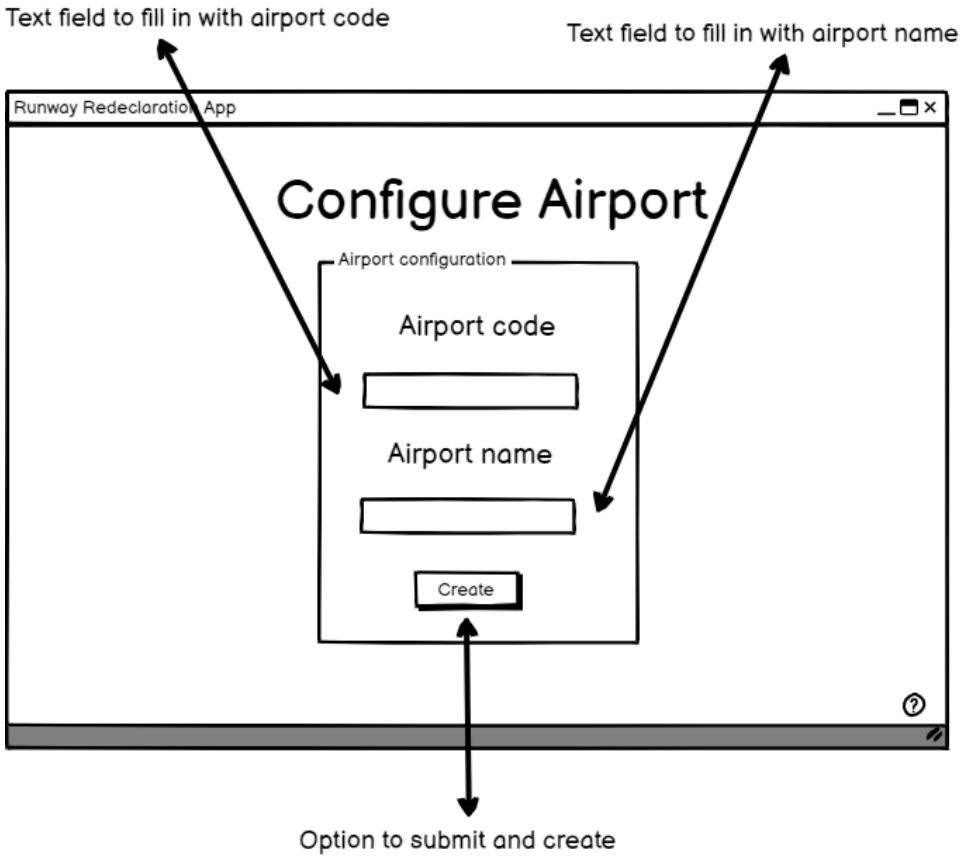


Figure 10: This page will pop up when user clicks on add airport for configuring airport

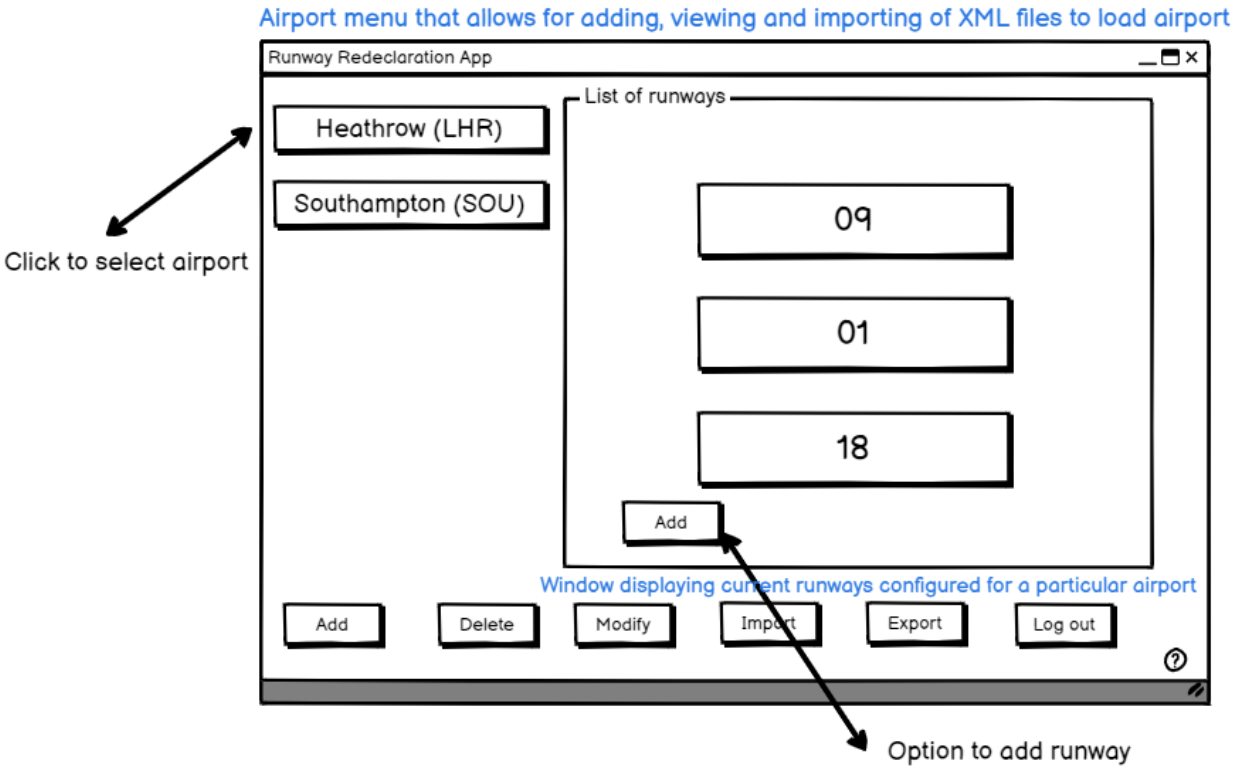


Figure 11: After selecting airport, user can now select runways to view specific runway

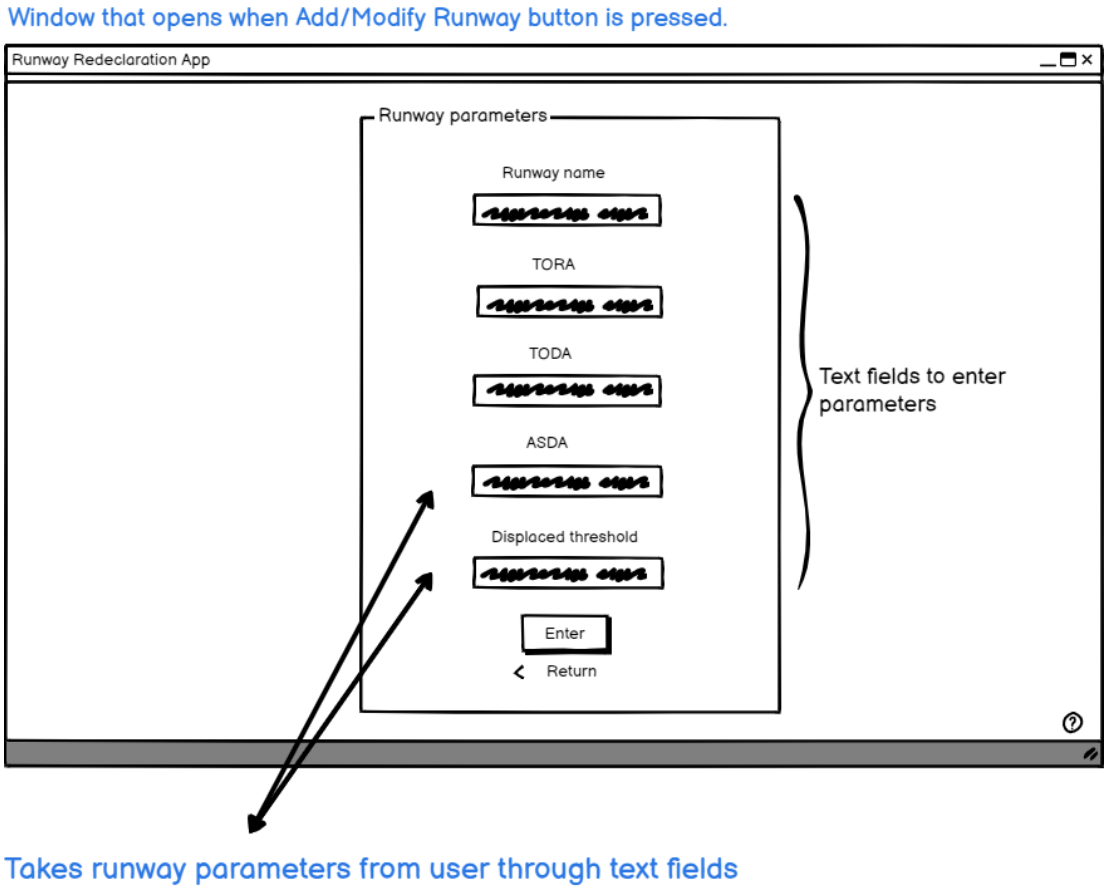


Figure 12: This will pop up when user clicks on add runway

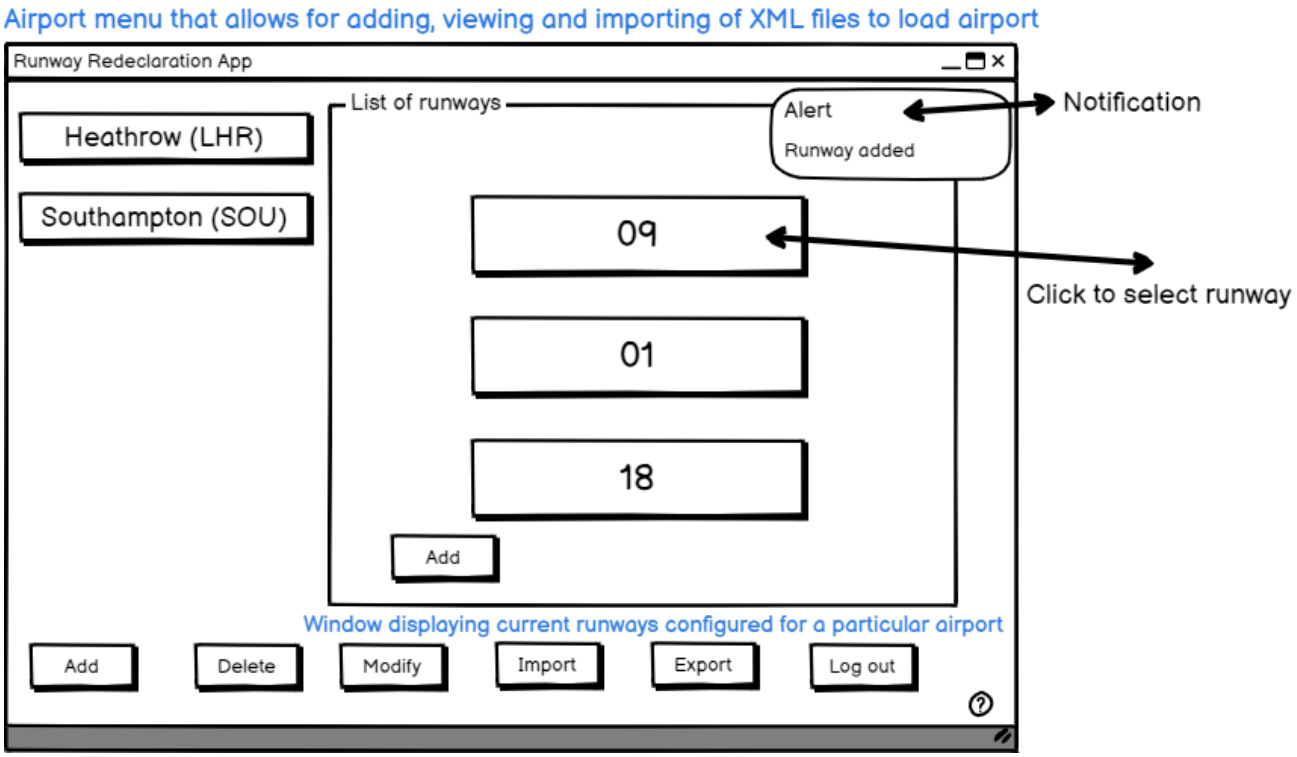


Figure 13: The list of runways will update according to user's input and a notification message will show indicating the operation.

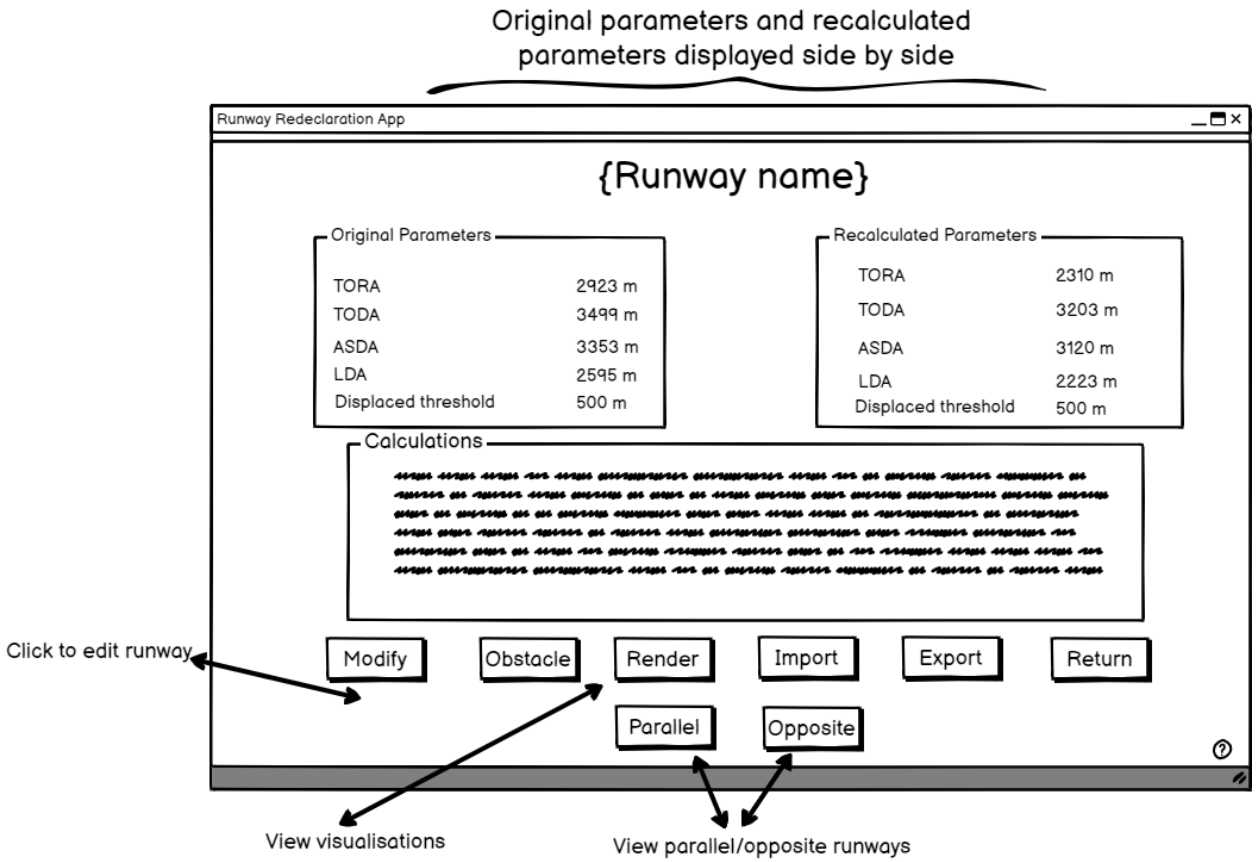


Figure 14: After selecting a runway, user can view details about the selected runway

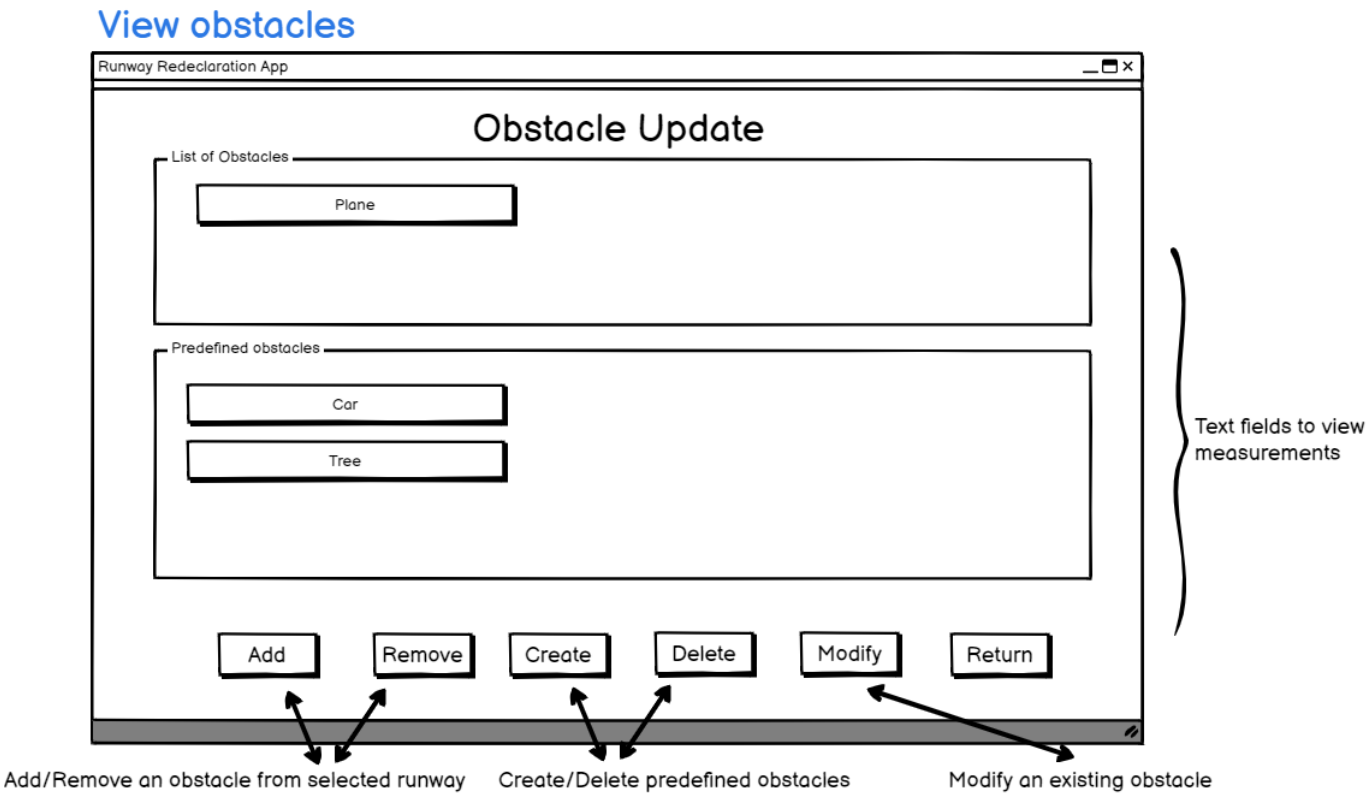


Figure 15: User have the option to view the obstacle on selected runway and any predefined obstacles

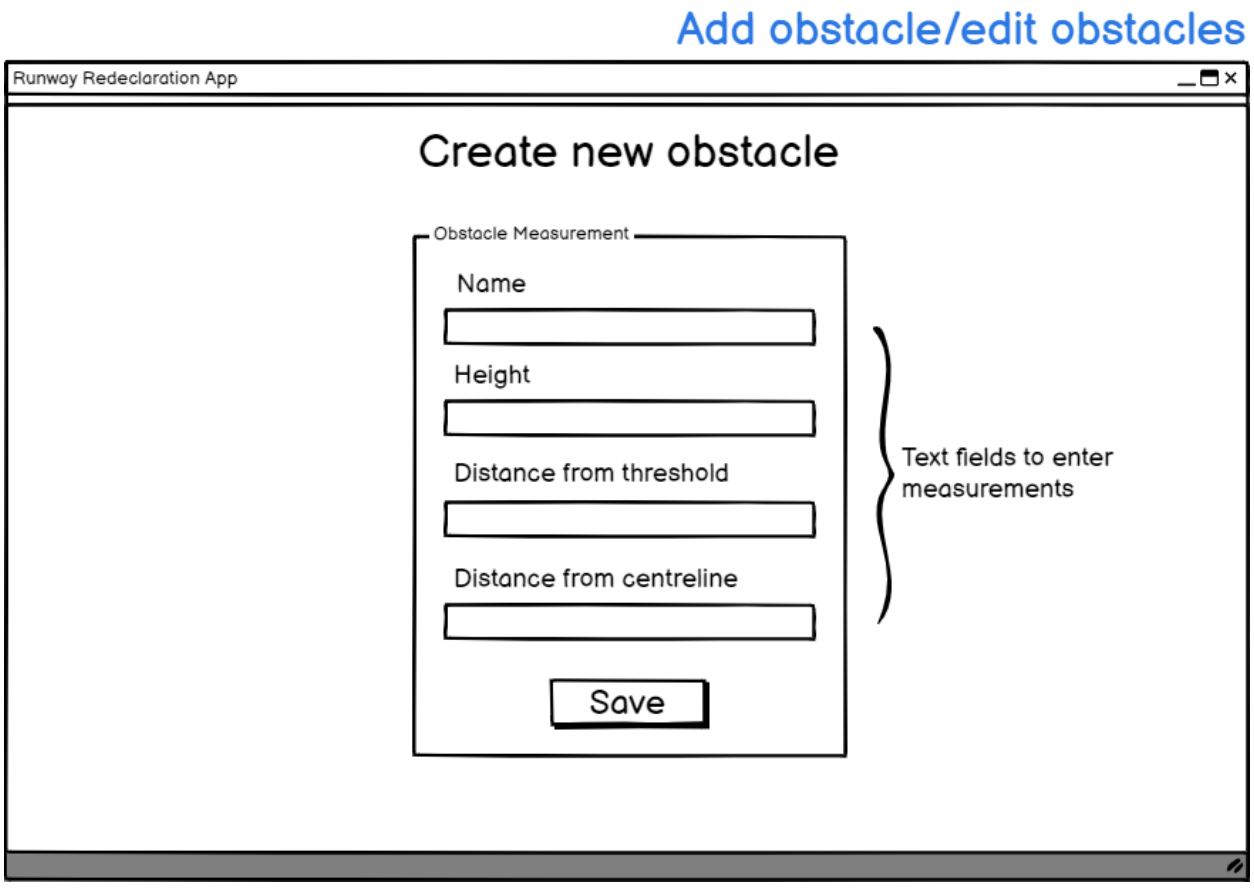


Figure 16: This page will pop up when user selects to create an obstacle

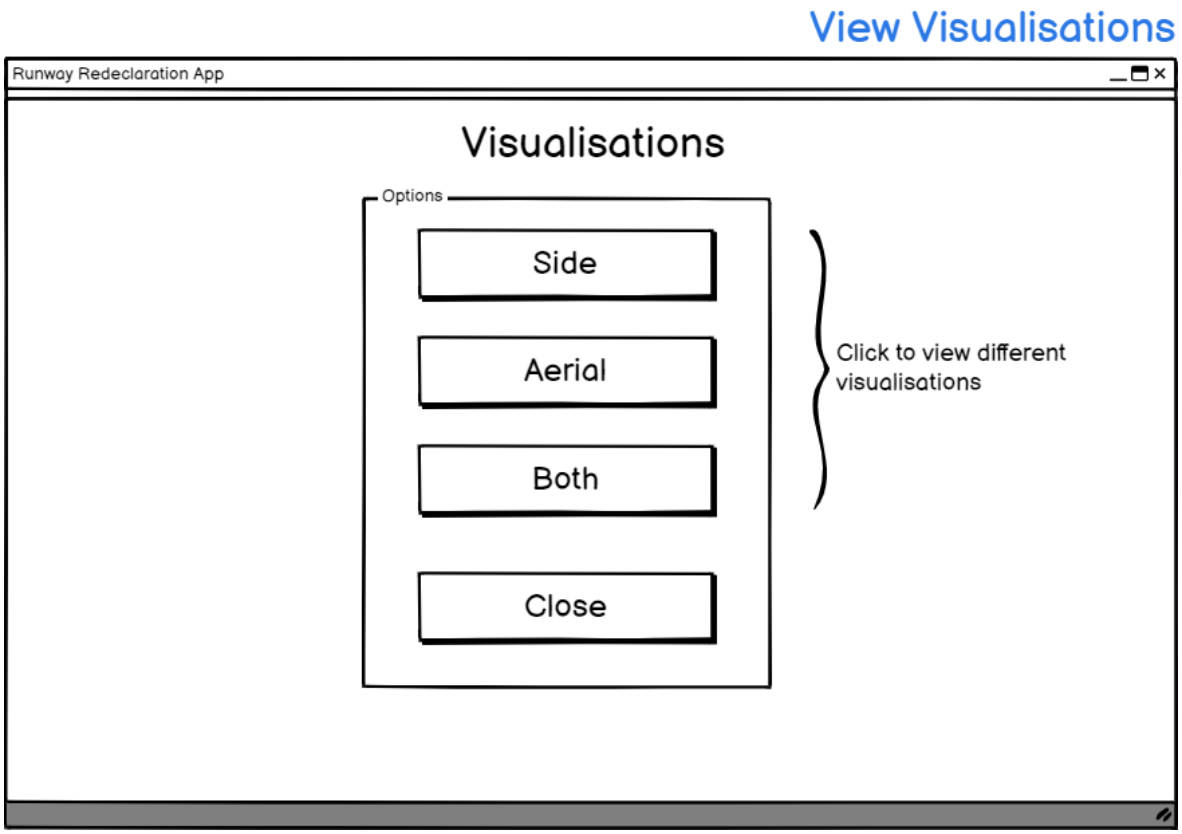


Figure 17: User can click on Render to view the options for visualisations

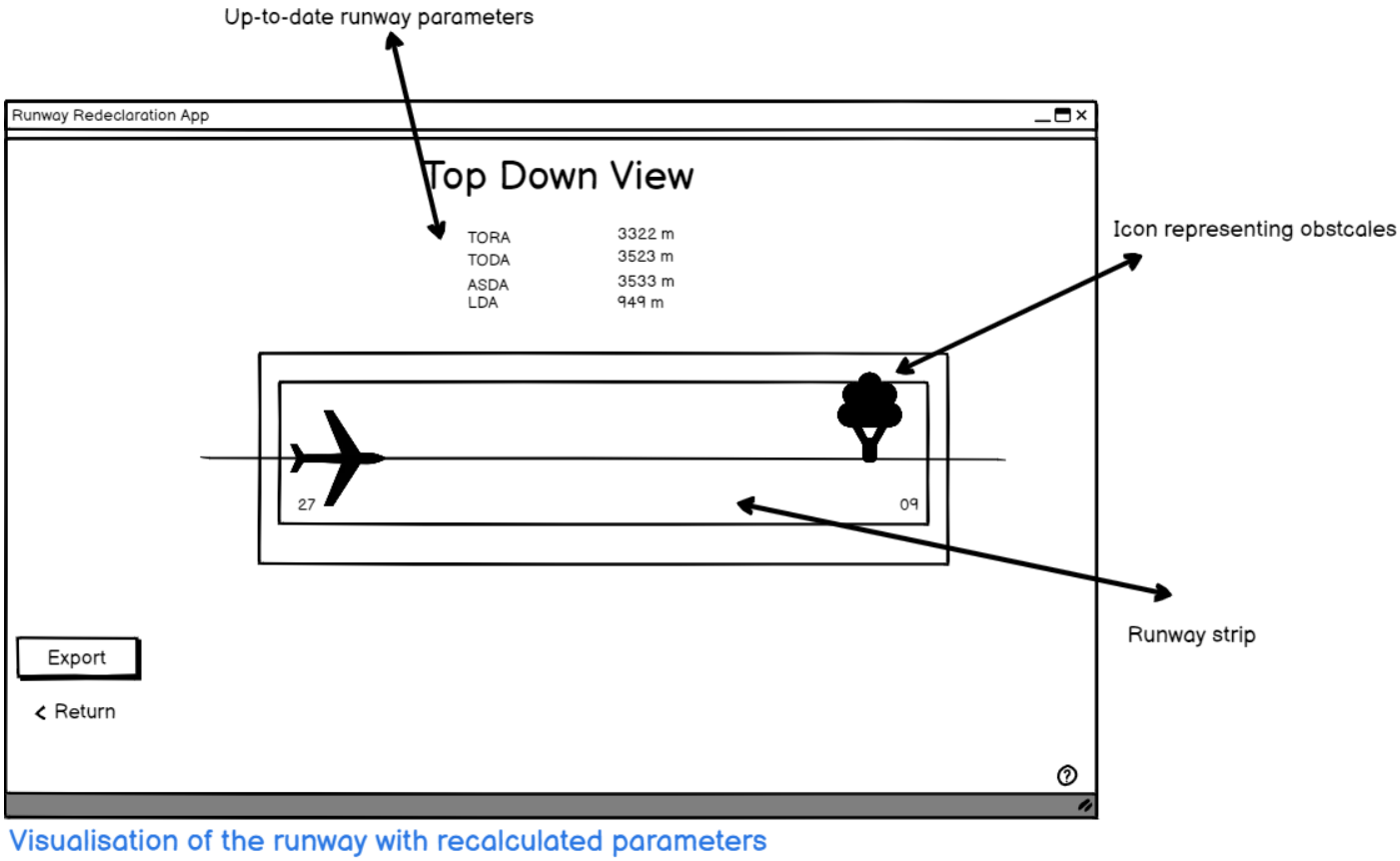


Figure 18: This displays the top down view of the selected runway

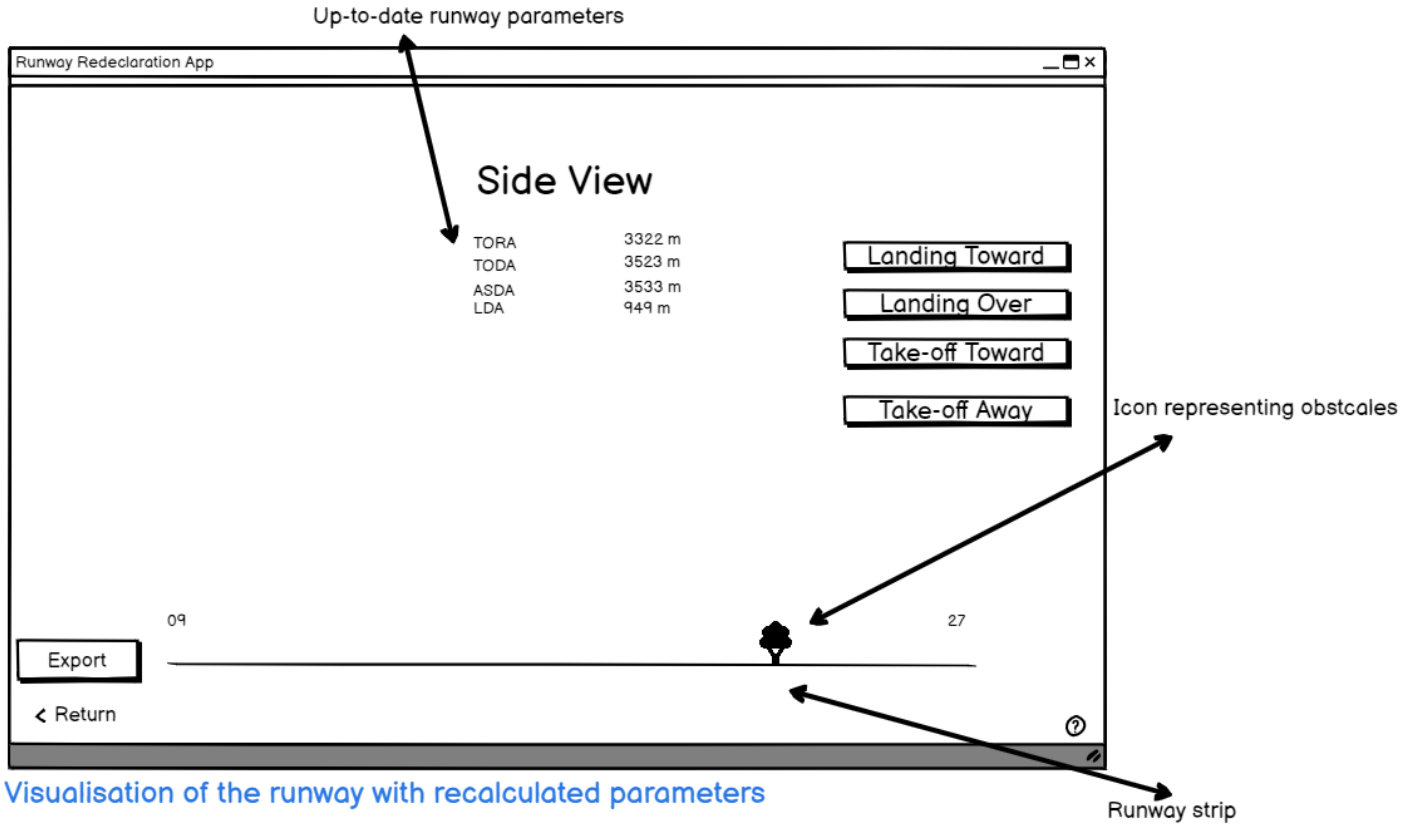


Figure 19: This displays the side on view of the selected runway

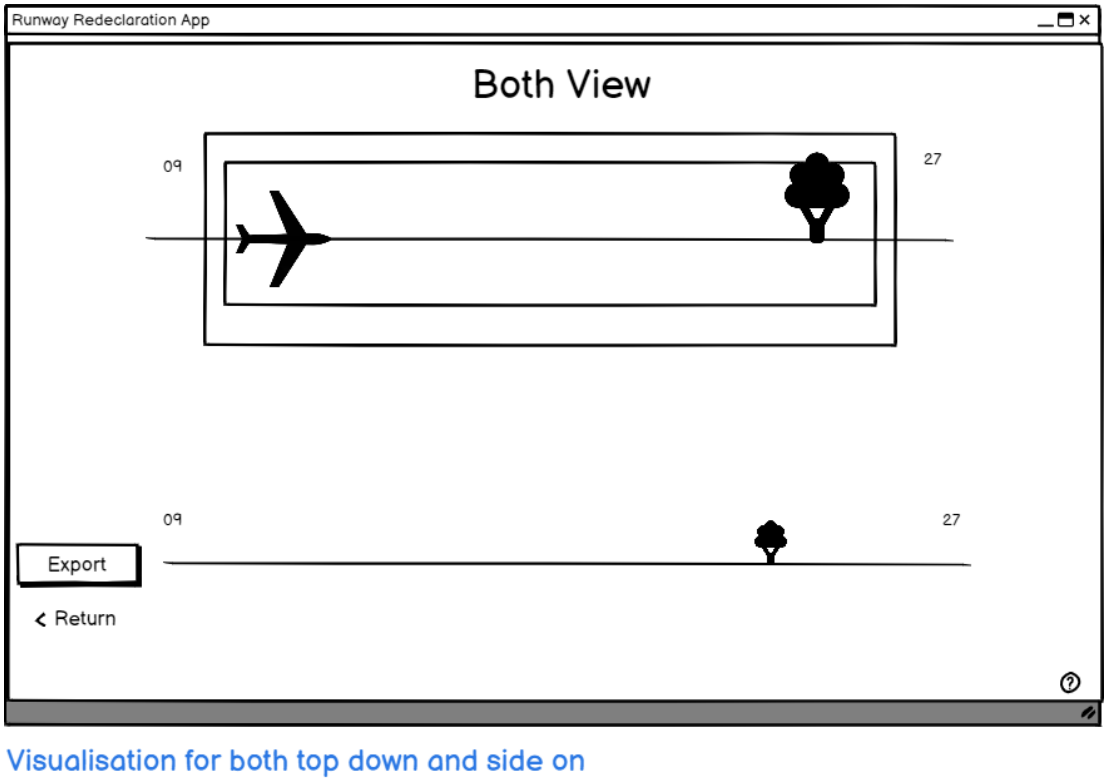


Figure 20: This displays both the side on and top down views of the selected runway

2 Testing

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

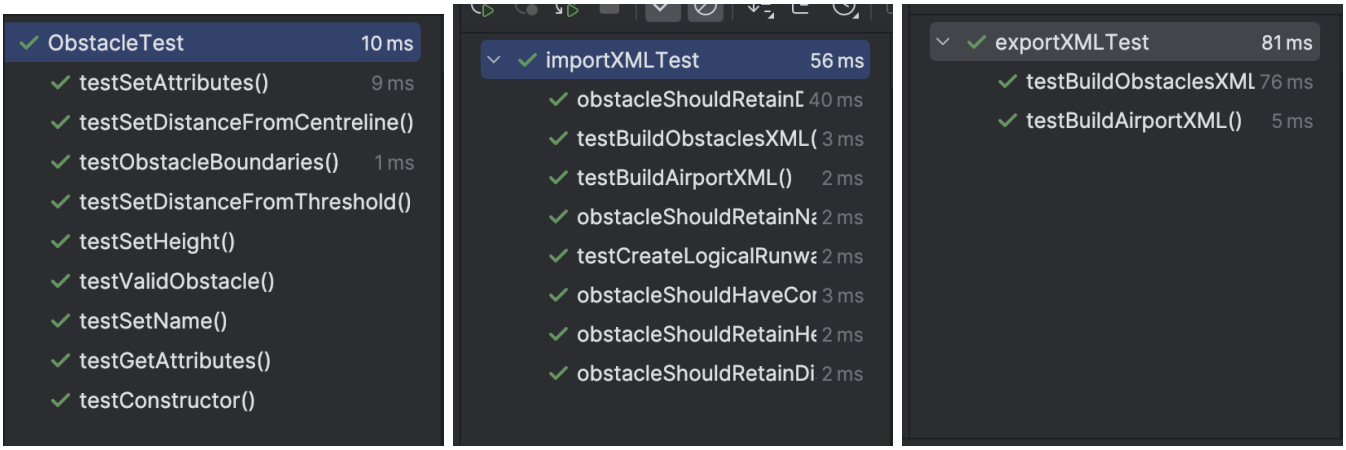
To test our application, we used a range of techniques including defect and validation testing, also boundary, partitioning, and regression testing. These various testing techniques will help us to thoroughly test our program.

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.



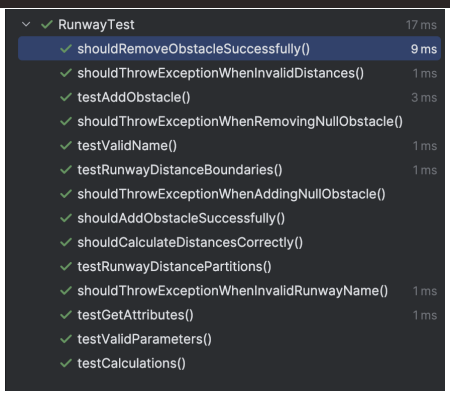
Test Result: ALL PASSED

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

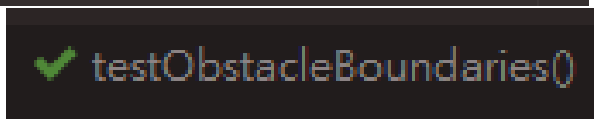
Test Definition: Boundary testing is a method where the focus is on the boundary or limit conditions of the software being tested. It involves testing at the extreme ends or boundaries of input values. The goal is to identify any errors that occur at the edges of the input domain, such as maximum, minimum,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

```
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(
}
```

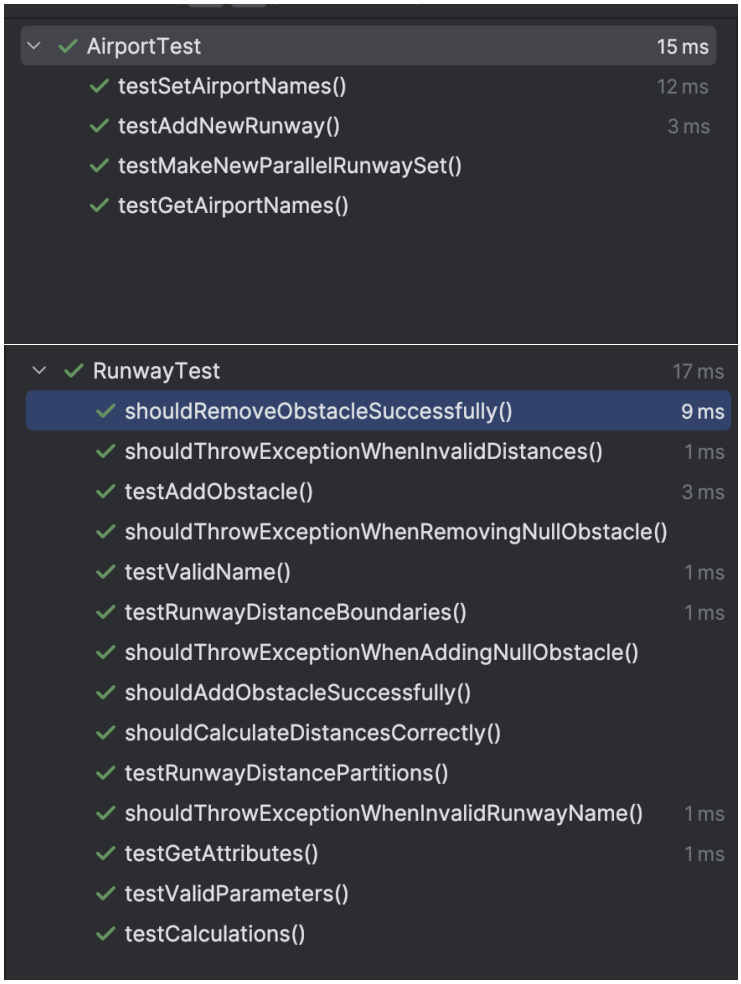
Refer section on Validation Testing and Defect testing to see passed tests. Test Result: ALL PASSED

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 tested acceptance criteria created for tested user story. As we still in the middle of our application development process, more testing will be done and the table will be updated with more user stories and their acceptance criteria in the next increment.

ID	Nickname	Acceptance Criteria
1	Existing Runway	1. User can configure existing runways specific to the UK commercial airport. 2. Enable easy adjustments for different layouts and operations.
2	New Runway	1. User can add new runways 2. User can view added runways and it's parameters 3. Values displays according to user's input
3	2D Top-down	1. Provide a clear and detailed 2D top-down view of the runway 2. Enable easy viewing and declaration of areas on the runway. 3. Displays all key elements
4	2D Side-View	1. Provide a clear and detailed 2D side view of the runway 2. Enable easy viewing and declaration of areas on the runway. 3. Displays all key elements
7	View Calc	1. User can view recalculated runway parameters alongside the original parameters in a side-by-side format for easy comparison. 2. Both original and recalculated runway parameters are accurate. 3. The recalculated parameters can be refreshed or recalculated without affecting the original parameters. 4. User can clearly distinguish between the original and recalculated parameters. 5. User are able to identify when no obstacle presents on a runway as no calculated parameters are displayed.
14	Centreline	1. Centreline must be clearly visible in the top-down view. 2. Centreline should accurately reflect the real runway centreline. 3. Display aids in guiding ATC for takeoff and landing planning.
15	Lower Threshold	1. Lower threshold displayed in a fixed position on screen. 2. Display aligns with standard air traffic control procedures. 3. Lower threshold display is clear and legible.
16	Auto Rotate	1. User can rotates Runway strip to match compass heading. 2. User can switch between rotated and not-rotated runway strip
20	Select Runway	1. User can select different runways and thresholds. 2. Views update automatically upon selection.
21	Notify Action	1. User receive notifications for significant actions. 2. Notifications are real-time. 3. Notifications are clear and informative.

Table 1: User Stories Acceptance Criteria

Test Examples:

Example 1: Testing User Story 7 (View Calc)

As an *airfield safety staff,* ***I want*** *automatic calculation of the new available runway distances when one obstacle is present with its specific dimensions and location,* ***so that*** *I can quickly make arrangements and communications to ensure the safe operations.*

Acceptance Criteria:

1. User can view recalculated runway parameters alongside the original parameters in a side-by-side format for easy comparison.
2. Both original and recalculated runway parameters are accurate.
3. The recalculated parameters can be refreshed or recalculated without affecting the original parameters.
4. User can clearly distinguish between the original and recalculated parameters.
5. User are able to identify when no obstacle presents on a runway as no calculated parameters are displayed.

Test Cases:

1. **Test Case 1: Display Validation**
   Objective: To verify that original and recalculated runway parameters are displayed side-by-side.
   Steps:
   (a) Log in as airport safety staff.
   (b) Navigate to the runway parameters section.
   (c) Update obstacle on runway to initiate a parameter recalculation.
   (d) Observe the display of parameters.

   Expected Outcome: Original and recalculated parameters are displayed side-by-side, clearly labeled and distinguishable.
   Actual Outcome: Both original and recalculated parameters are displayed side-by-side, clearly labeled and distinguishable.

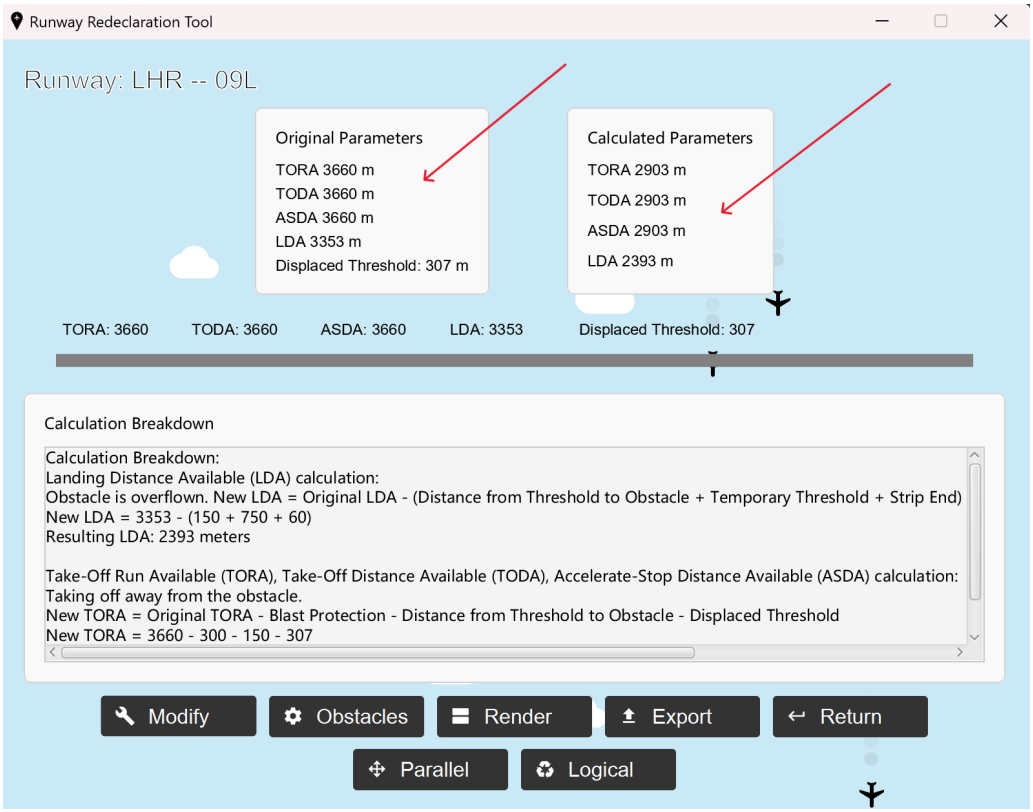


Figure 21: Evidence for test case 1

Test Result: PASSED

2. **Test Case 2: Data Accuracy Check**
   Objective: To ensure the accuracy of both original and recalculated parameters.
   Steps:
   (a) Compare the displayed original parameters with recorded data.
   (b) Verify the recalculated parameters against manual calculations.

   Expected Outcome: Both sets of parameters match their respective sources accurately.
   Actual Outcome: Both sets of parameters matched their respective sources accurately.

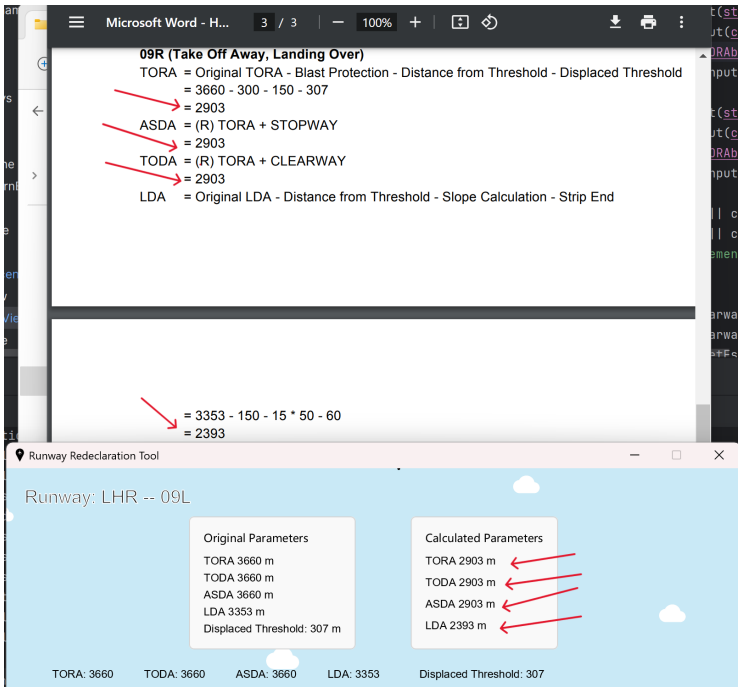


Figure 22: Evidence for test case 2: Result matches with given example calculations

Test Result: PASSED

3. **Test Case 3: Recalculation Functionality**
Objective: To test the recalculation functionality and its impact on display.
Steps:
   (a) Change the obstacle on selected runway
   (b) Observe changes in recalculated parameters while original remain unchanged.

Expected Outcome: Recalculated parameters update accordingly, while original parameters stay the same.
Actual Outcome: Recalculated parameters updated accordingly, while original parameters stayed the same.

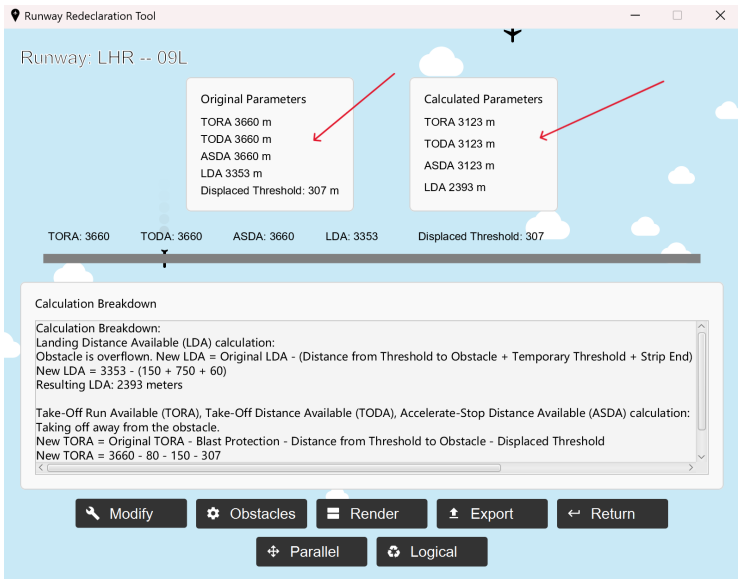


Figure 23: Evidence for test case 3: Recalculated parameters updated

Test Result: PASSED

4. **Test Case 4: No Obstacle**
   Objective: The system indicates no obstacle presents and displays no calculated parameters.
   Steps:

   (a) Remove obstacle on selected runway

   (b) Observe the system's response.

   Expected Outcome: System only displays the original parameters.
   Actual Outcome: System only displayed the original parameters.

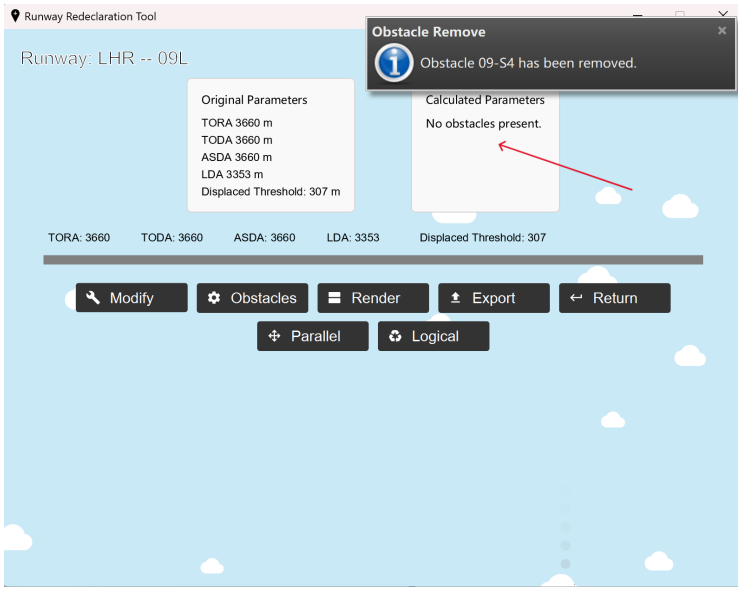


Figure 24: Evidence for test case 4: No Recalculated parameters

Recalculated parameters updated

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 1 - Melisa (Airfield Operations Manager)

Objective: To test the functionality of the 'Add Airport' process as experienced by Melisa, the Airfield Operations Manager.

Test Steps:

1. Start the runway re-declaration software.
2. Log in with Melisa's username and password.
3. Expect the Airport Database page to load with correct options visible.
4. Click 'List' to view configured airports.
5. Verify that an appropriate warning message is displayed if the list is empty.
6. Click 'Add' to trigger the 'Add Airport' dialog.
7. Enter only the airport name and click 'Add'; expect an alert about missing fields.
8. Acknowledge the alert, complete the required fields, and click 'Add'.
9. Verify that the new airport appears in the airport list.
10. Select the newly added airport and click 'Select'.
11. Confirm that the list of runways is empty and then choose to 'Import' runway data.
12. Ensure all runways for the airport are added to the list.
13. Click on a runway where an obstacle has been reported.
14. Navigate to 'Obstacles - Create' and verify the 'Create Obstacle' dialog appears.
15. Fill in all fields in the obstacle creation form and press 'Create'.
16. Check that the new obstacle is correctly listed in the predefined obstacles.

Expected Outcome: Each step should occur without errors, and the system should react as described in the steps. Obstacle creation should be successful and accurately reflected in the system with appropriate notification and update.

Actual Outcome: The test carried out without errors, and the system reacted as described in the steps. The obstacle was created successful and accurately in the system with appropriate notification and update.

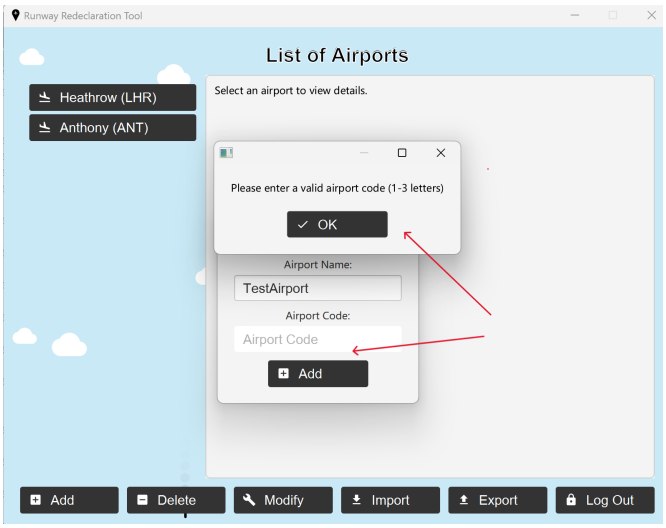


Figure 25: User enters only the airport name and receives an error message indicating needs to enter airport code

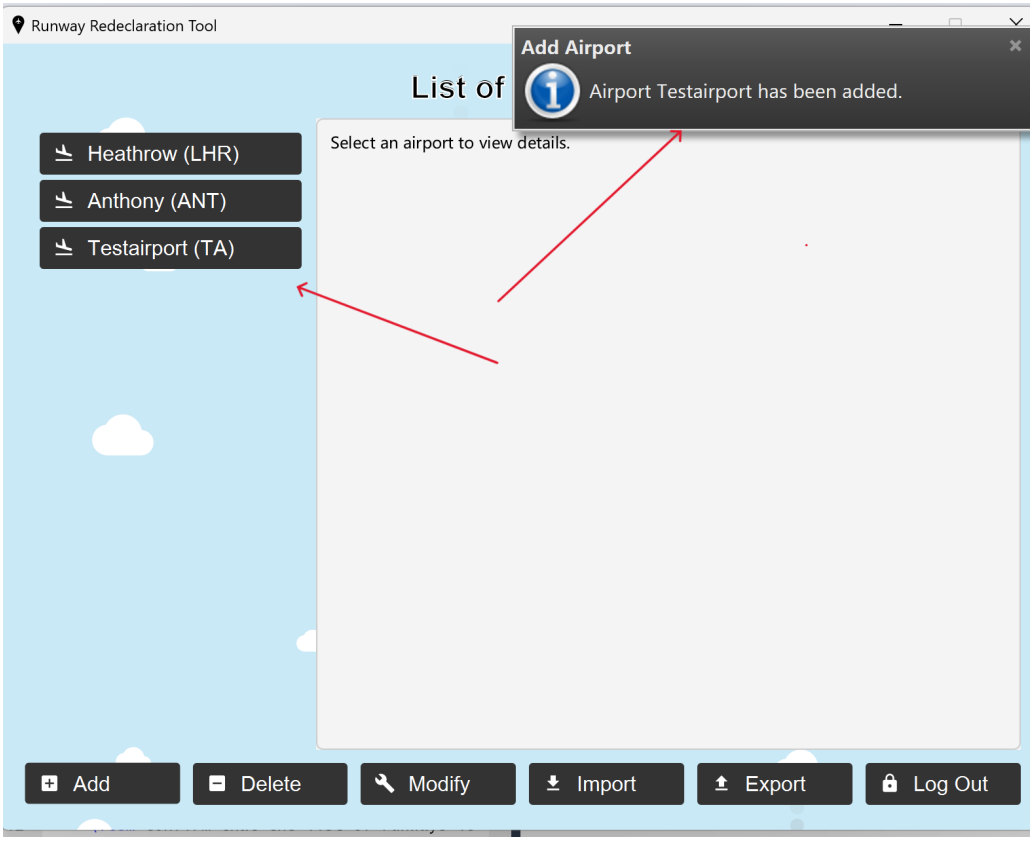


Figure 26: Airport successfully added

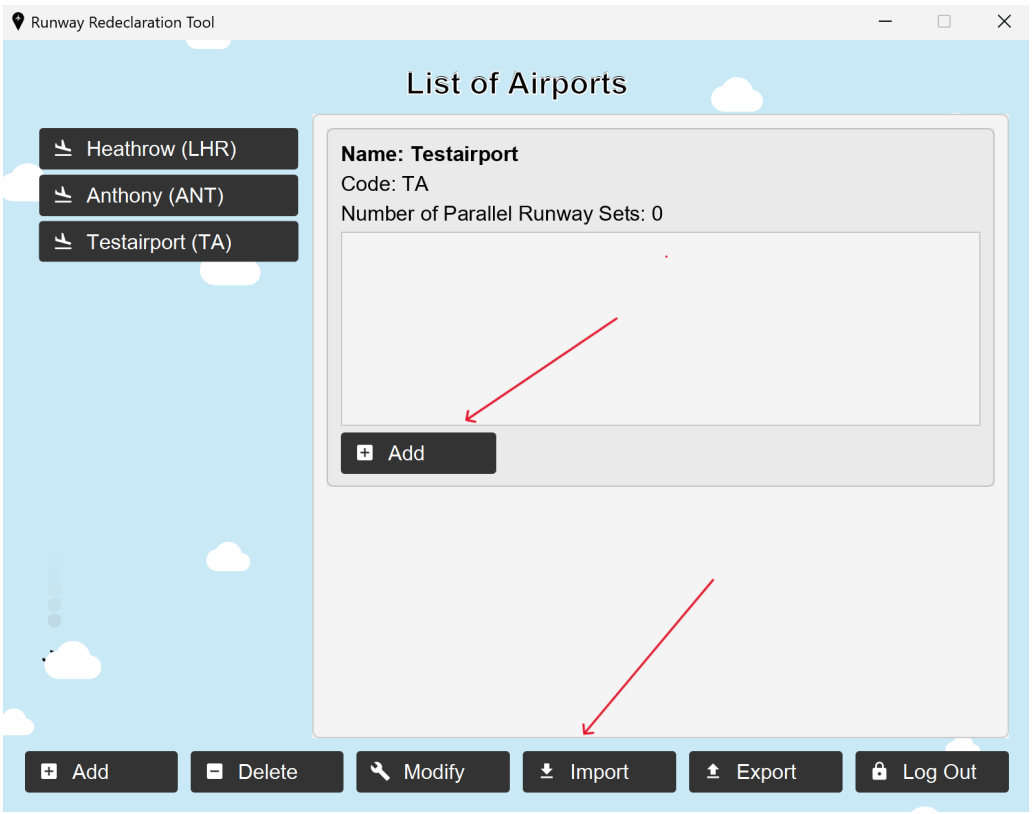


Figure 27: User can now add/import runways

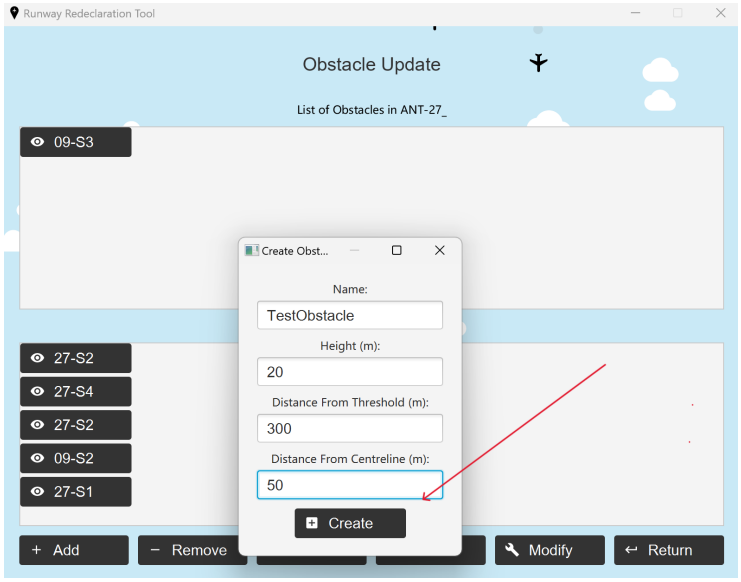


Figure 28: User clicks on create and enters obstacle details to create an obstacle

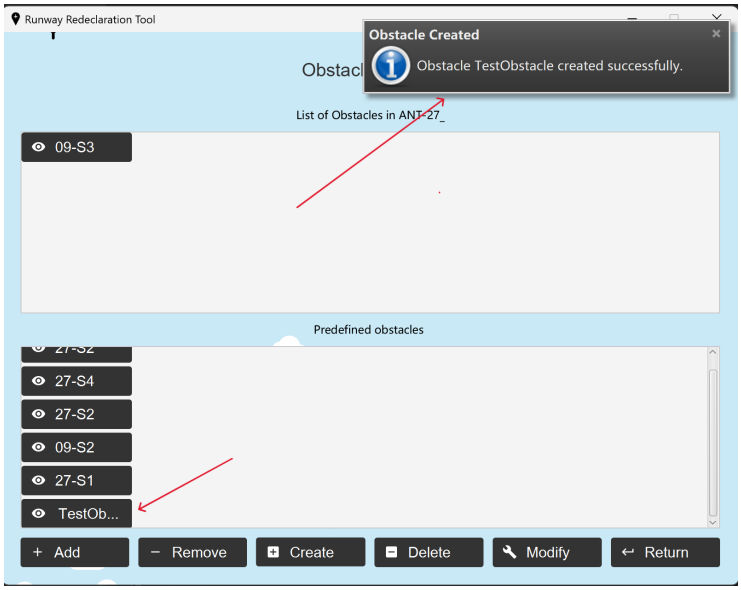


Figure 29: Obstacle successfully created and added to the predefined list

Test Result: PASSED

Example 2: Testing Scenario 2 - Captain Maxine (Airline Pilot)

Objective: To test the airport addition and deletion process as performed by Captain Maxine, the Airline Pilot.

Test Steps:

1. Launch the runway re-declaration software.
2. Click on the "login" button.
3. Enter the valid username and password for Maxine and click the "login" button to submit the credentials.
4. Verify that the airport database screen is displayed after a successful login.

5. Click the "Add" button to open the prompt for entering airport details.
6. Enter the details of the airport and click on "Add".
7. Verify that the prompt closes and the airport database screen is shown.
8. Click the "List" button to view the list of airports.
9. Verify that the newly added airport is displayed on the Airport List page.
10. Click on the newly added airport, which needs to be deleted.
11. Click on the "Delete" button.
12. Confirm that the selected airport is removed from the airport list.
13. Verify that the airport's runway list no longer displays the deleted airport and is updated correctly.

Expected Outcome:

- At each step, the system behaves as described, including providing error feedback if login details are incorrect.
- The airport details are added and displayed correctly.
- The airport is deleted successfully when the "Delete" button is clicked, and the airport list is updated accordingly.

Actual Outcome: The system reacted as described in the steps. Appropriate error message was provided if login details are incorrect. The airport details are added and displayed correctly. And deleting airport was successful with list of airports updated.

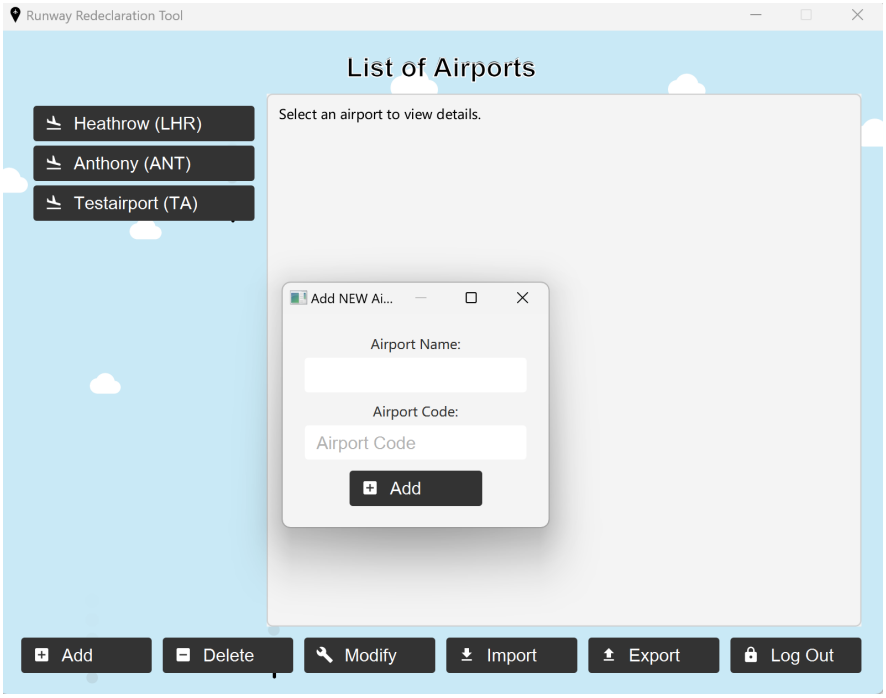


Figure 30: Add airport dialog was prompted

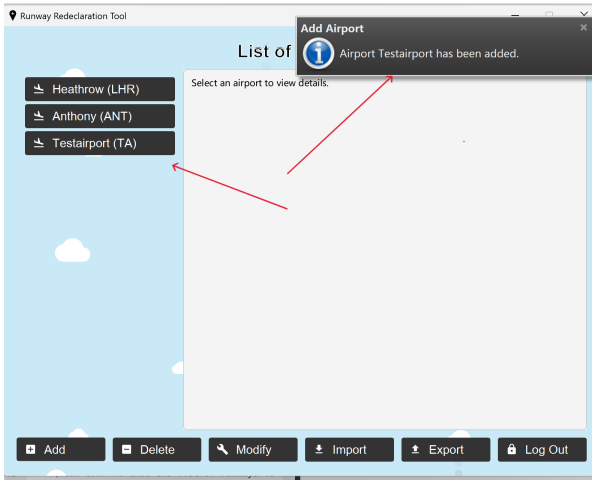


Figure 31: Add airport dialog closes when user click add, and list of airport updated with notification

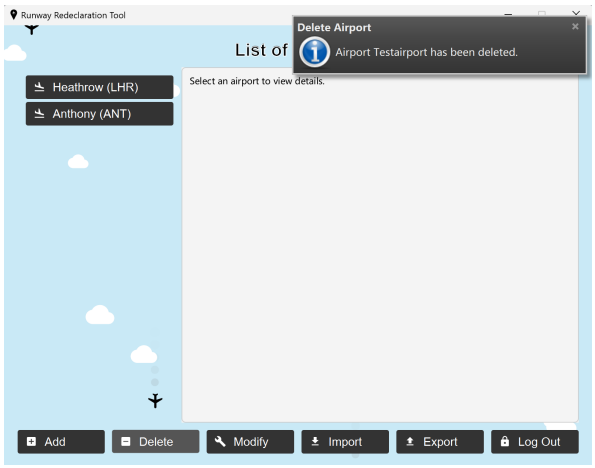


Figure 32: Newly added airport was successfully deleted, list of airport updated with notification

Test Result: PASSED

3 Planning

3.1 Completed Sprint Plan for Deliverable 3

The following sprint plan outlines the tasks, owners, estimated hours, actual hours and priorities that are completed for Deliverable 3.

We added 3 more user stories to our project backlog, which are **Animations, Background music and Dark mode**. These user stories are labeled as '**could have**' as they serves as additional functions within our application that can enhance user experience and accessibility.

Please see **appendix** for our updated product backlog and sprint plan diagrams.

We also decided to change the task for the **Auto Rotate** user story, as rotating the runway strip make the display of the views less user friendly by making it difficult for users to identify the direction properly. Thus we came up with an alternative implementation of this user story, which is adding a compass to the top-down view with it's pointer pointing at the direction of the runway. This makes it easier for the users to identify the correct direction that the runway is pointing at.

User Story	Sprint Backlog (Tasks)	Member	Est. Hours	Actual Hours
3: 2D Top-down	1. Create option to display a 2d top down view of selected runway only.	Bav	2	2
4: 2D Side-View	1. Create option to display a 2d side view of selected runway only.	Bav	2	1
5: Both Views	1. Create option to view both top down and side on views at the same time.	Yash	1	1
12: Side-View Display	1. display the runway strip 2. display Threshold indicators. 3. display Threshold designators e.g. 27Ror 09L, with the letter below the number. 4. display Any displaced thresholds that are present. 5. display Stopway/Clearway for both ends of the runway. 6. have Indication of the take-off / landing direction. 7. display All re-declared distances, with indicators showing where they start and end relative to the runway strip. 8. The distances should be broken down into their respective parts, including RESA/Blast Allowance. 9. display The obstacle, if one is present upon the runway. 10. display The offset caused by the RESA and slope angles relative to the obstacle on the runway.	Manlin	2	4
13: Top-View Display	1. display the runway strip 2. display Threshold indicators. 3. display Threshold designators e.g. 27Ror 09L, with the letter below the number. 4. display Any displaced thresholds that are present. 5. display Stopway/Clearway for both ends of the runway. 6. have Indication of the take-off / landing direction. 7. display All re-declared distances, with indicators showing where they start and end relative to the runway strip. 8. The distances should be broken down into their respective parts, including RESA/Blast Allowance. 9. display The obstacle, if one is present upon the runway. 10. display The offset caused by the RESA and slope angles relative to the obstacle on the runway. 11. display runway centreline.	Zagrosi, Manlin	4	5
14: Centreline	1. display runway centreline for Top-View only	Yash	2	3
15: Lower Threshold	1. have a fixed place on screen to display the Lower Threshold	Yash	2	2
16: Auto Rotate	1. display a compass with it's pointer pointing toward the direction of the runway, this helps to indicate the real direction of the runway.	Anthony	5	5
21: Notify Action	1. have appropriate pop-up message when action has taken place (e.g. obstacle added, runways re—declared, values changed)	Bav	1	3
27: Animations	1. create animations to stimulate landing/take-off movement of the aircrafts.	Zagrosi	1	1
28: Background music	1. have appropriate background music for the application	Zagrosi	1	1
29: Dark mode	1. create option to switch to dark mode	Zagrosi	1	1

Table 2: Completed Sprint Plan for Deliverable 3

3.2 Completed Burn-Down Chart for Deliverable 3

The burn-down chart is modified to include additional user stories we have decided to add and implement during increment 2 (deliverable 3).

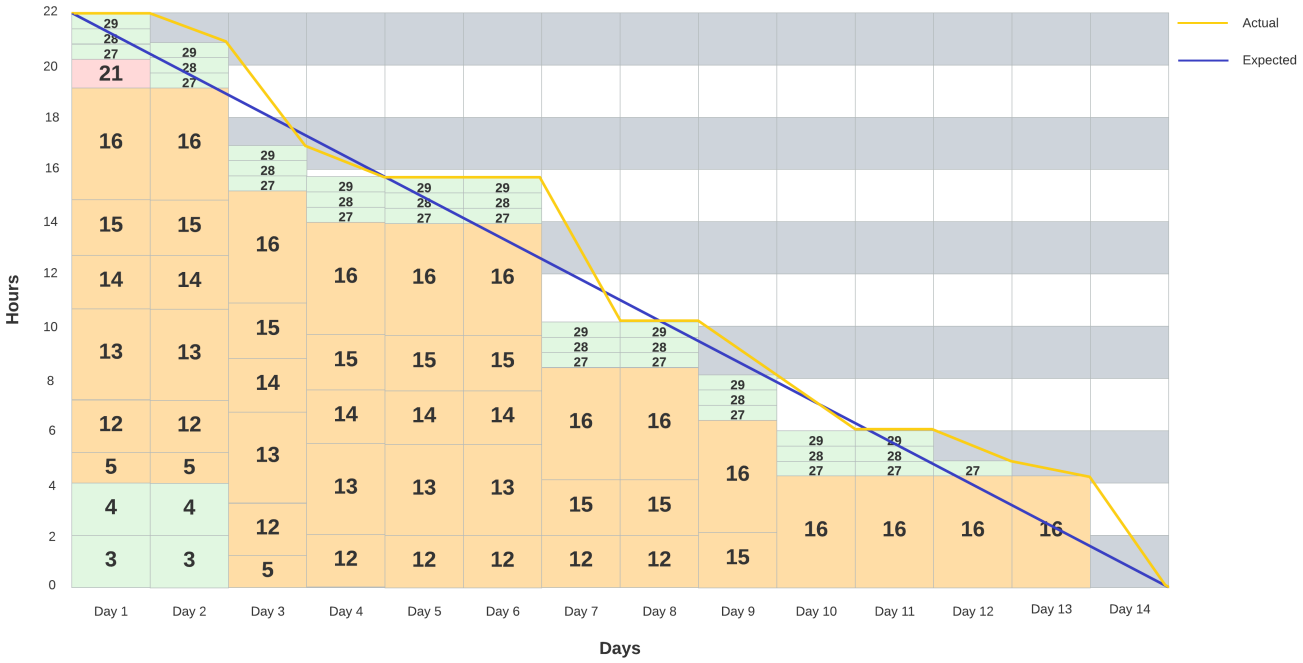


Figure 33: Completed Burn-down Chart for Increment 2 (Deliverable 3)

3.3 Day Zero Burn-down Chart Increment 3 (Deliverable 4)

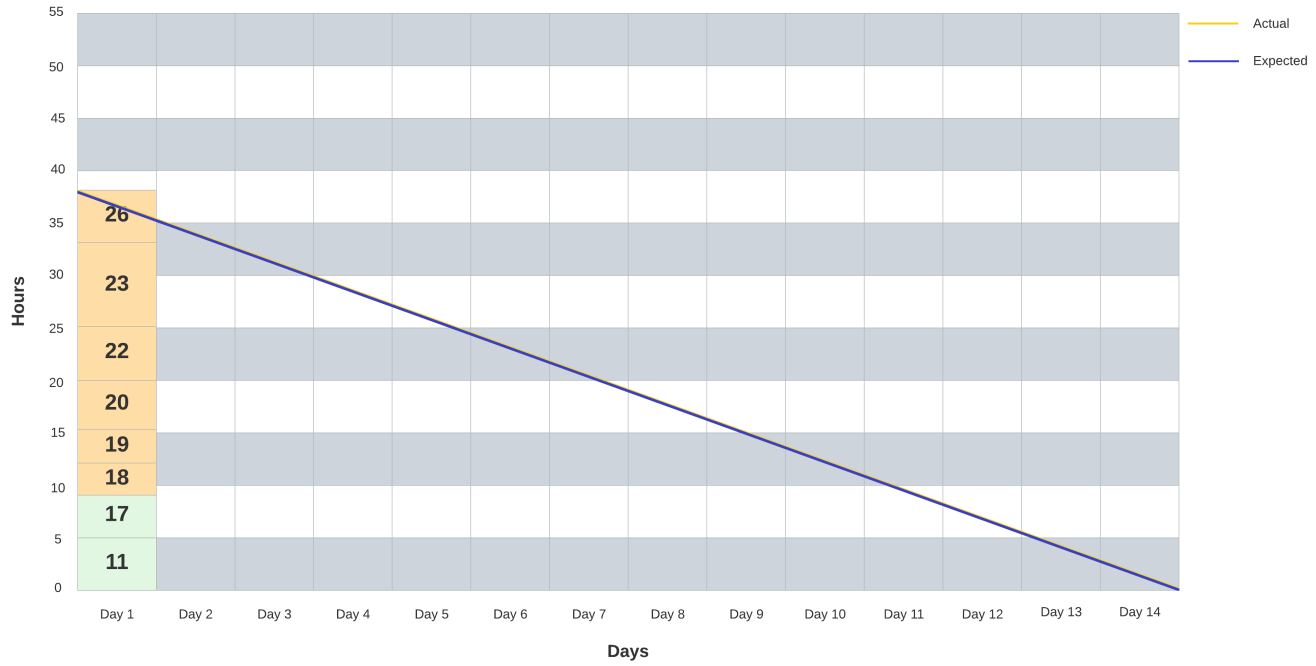


Figure 34: Burn-down Chart for Increment 3 (Deliverable 4)

4 Response to feedback in Deliverable 2

In the supervisor meeting after the marking session, our supervisor gave us many valuable feedback. Below are the feedback we received:

- Add background music to make the app more engaging.
- Add dark mode to increase accessibility.
- Fix the icon issue from last handin.
- Make diagrams more readable by splitting into smaller segments.
- For JUnit testing, show that you passed with screenshot evidence.
- When there is an update in storyboards, add to this increment report.

Based on the feedback we received, we have implemented changes to our program and design artifacts. Please see below for evidences.

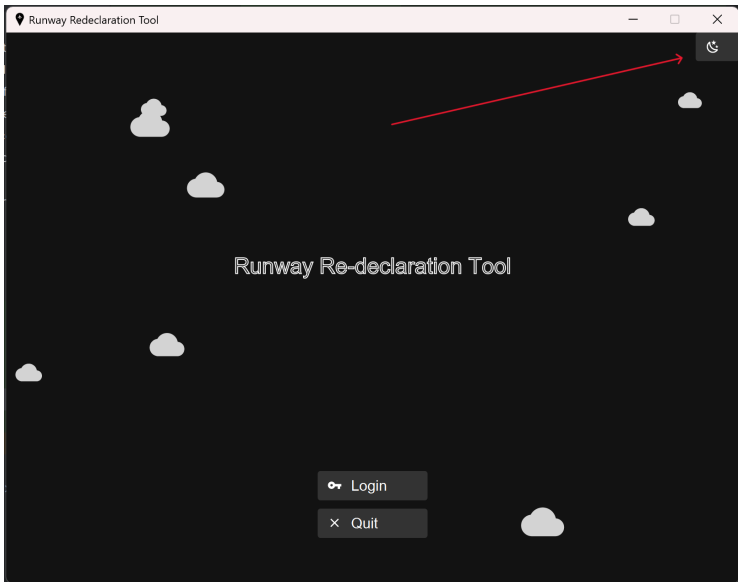


Figure 35: The dark mode functionality was implemented with a button to switch

The icon issue was fix, and the background music is implemented. These will be demonstrated in the marking session.

We modified the format of our design artifact diagrams to make them more readable, these are included in the **Design** section ablove in this report.

In the **Testing** section, we also added screenshots as evidences for our test result.

Also **Storyboards** was updated with our new UI designs.

5 Appendix

5.1 Product Backlog

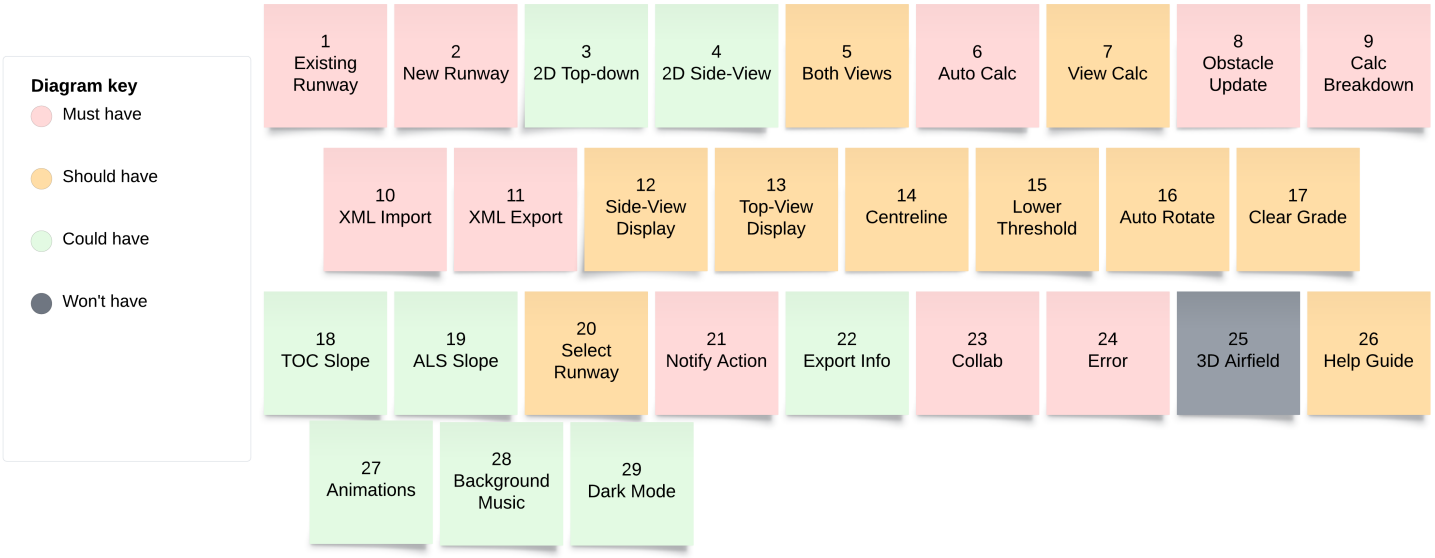


Figure 36: Product Backlog

5.2 Sprint Plans

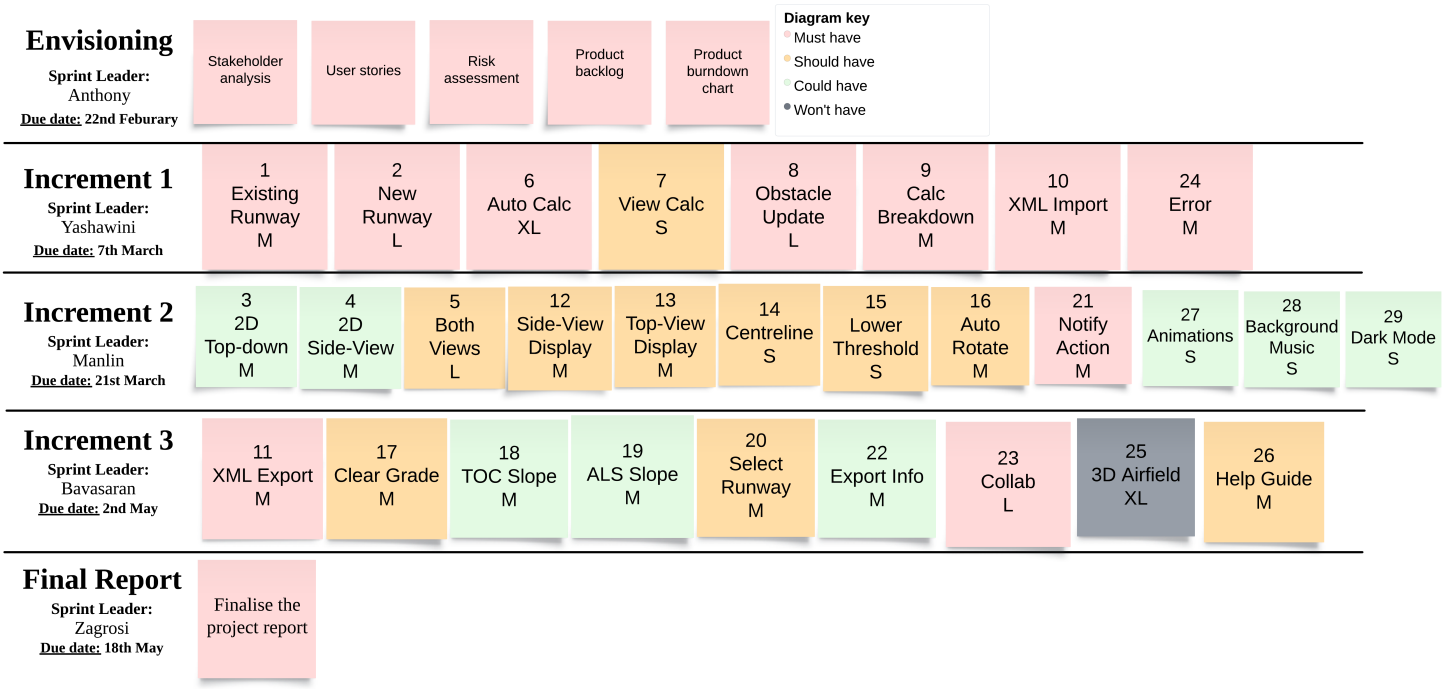


Figure 37: Sprint Plans

5.3 State Machine Diagram

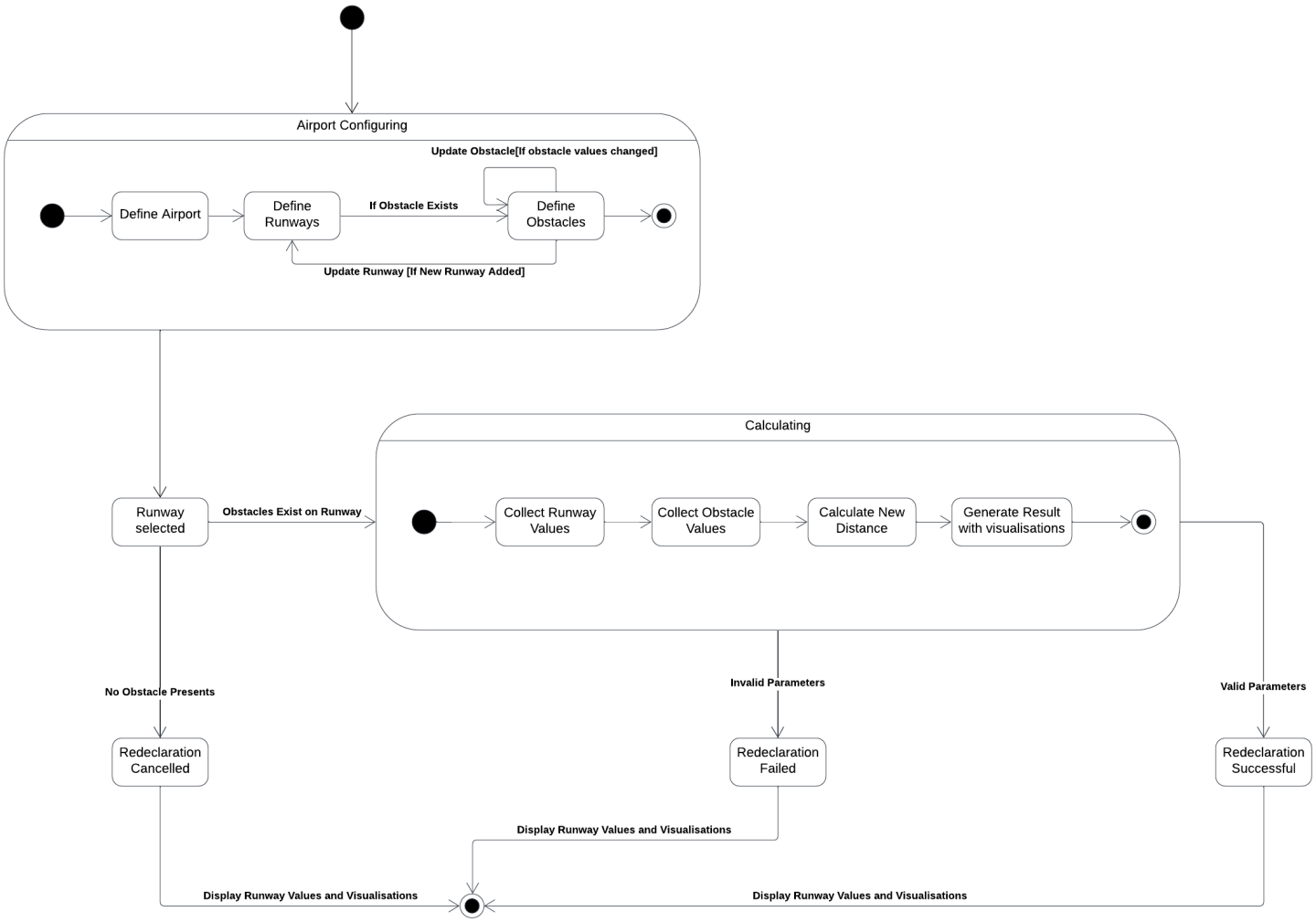


Figure 38: State Machine Diagram

5.4 Use Case Diagram

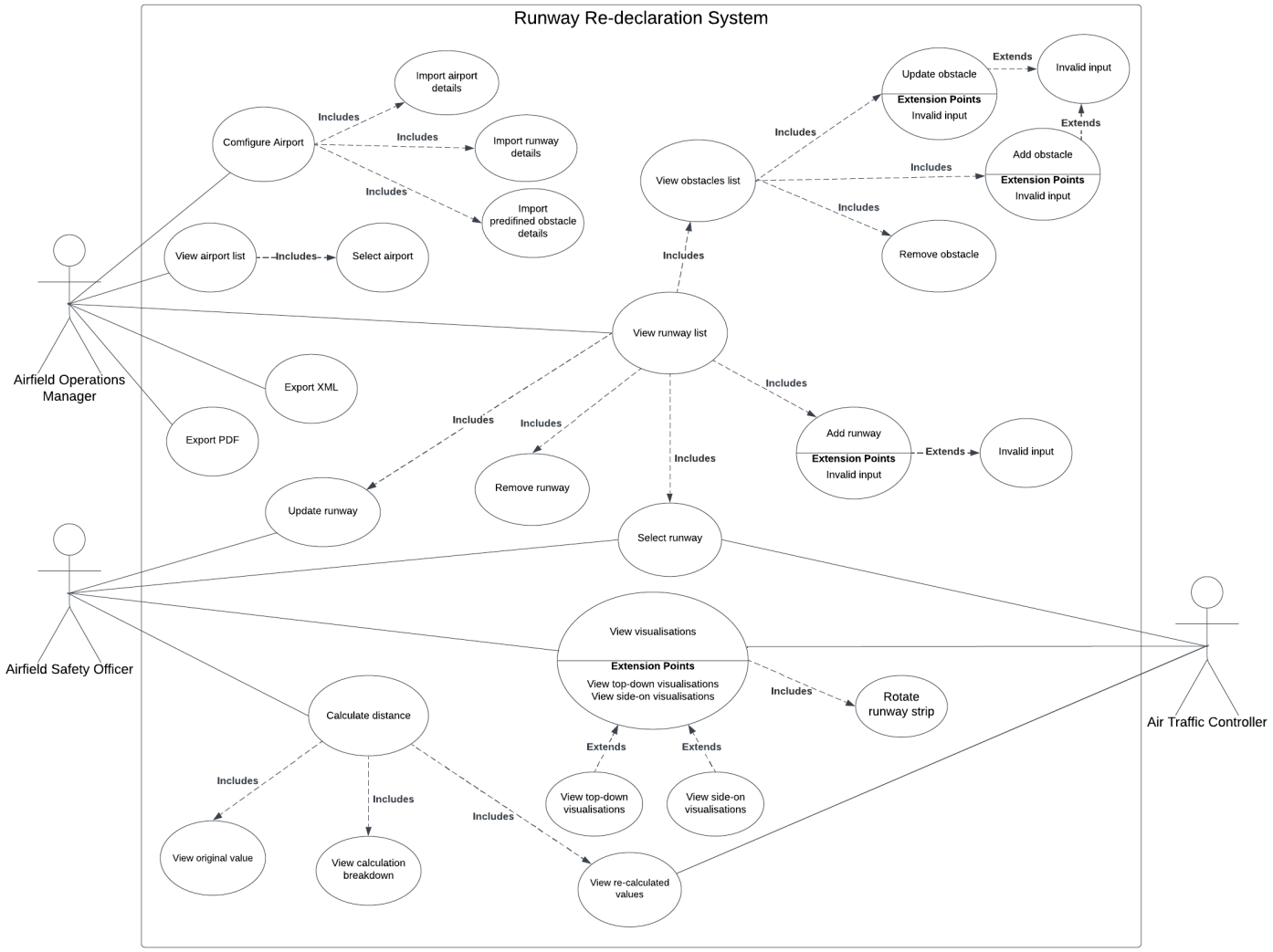


Figure 39: Use Case Diagram

Bibliography

References

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