COMP2211 Runway Redeclaration
Group 2- Deliverable 3, Increment 2

Eren Rafet (er10g23)
Bozhang Wu (bw9n21)
Louis Townsend (lmt1n22)
Abdullah Hariry (ah4u22)
Hossameldin Tammam (htft1e22)

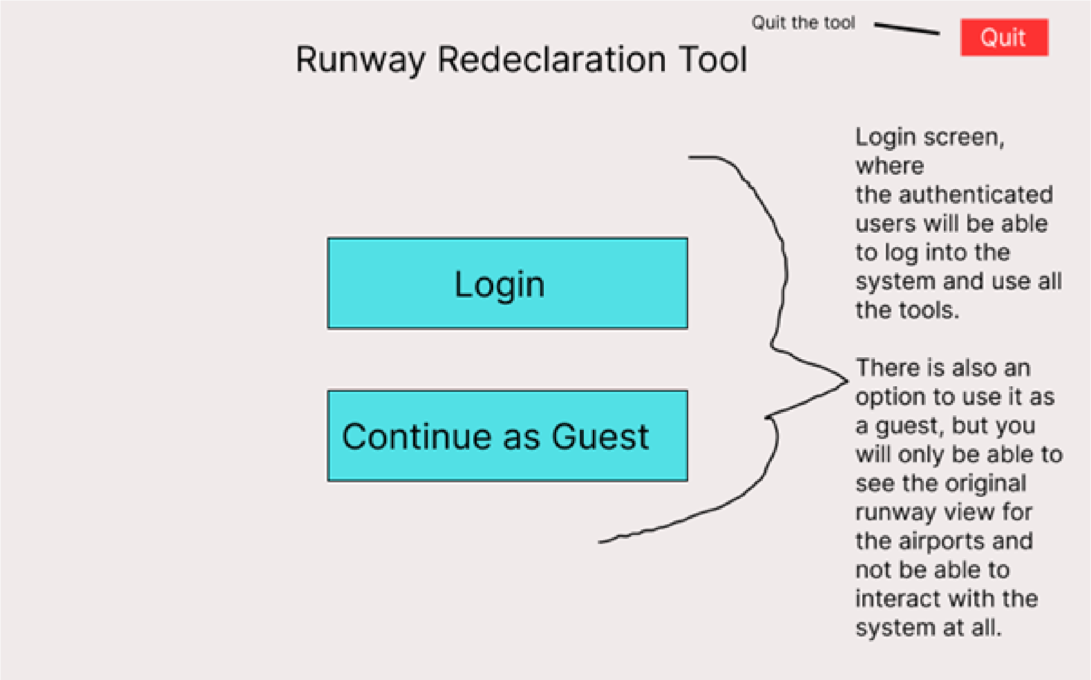
March 2025

Contents

1 Responses to Feedback

1.1 Annotations for Storyboards

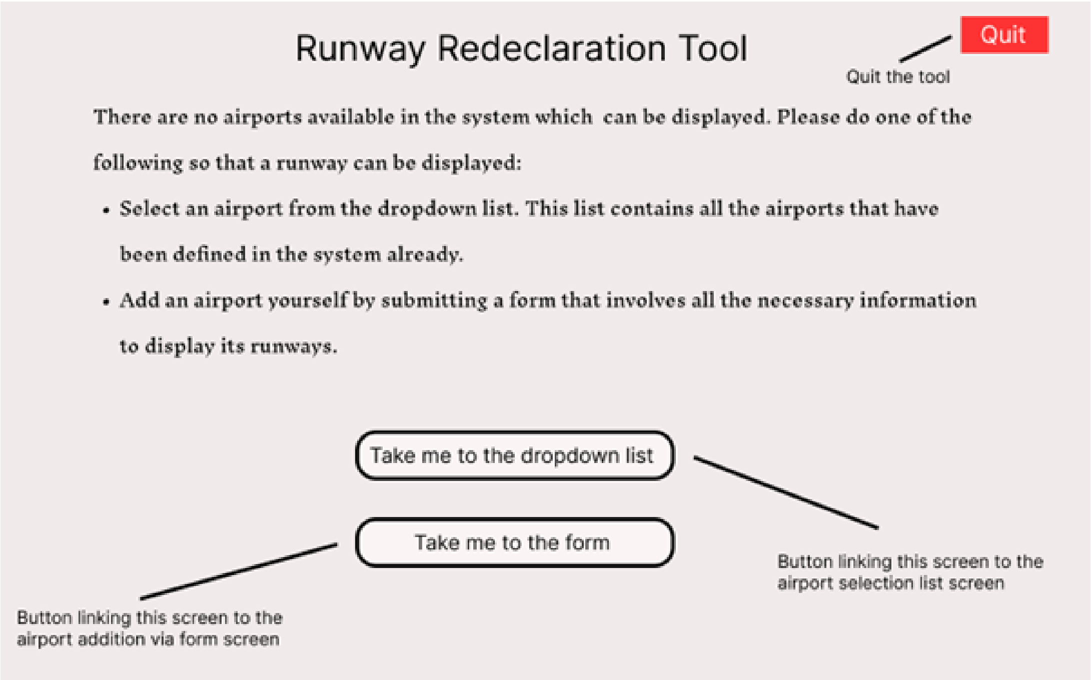
We included short paragraphs summarising each storyboard; however, adding annotations would provide developers and the customer with deeper insight into how the interface would function, making the storyboards easier to follow.



Login screen, where the authenticated users will be able to log into the system and use all the tools.

There is also an option to use it as a guest, but you will only be able to see the original runway view for the airports and not be able to interact with the system at all.

Figure 1: Screen that the users are presented with initially



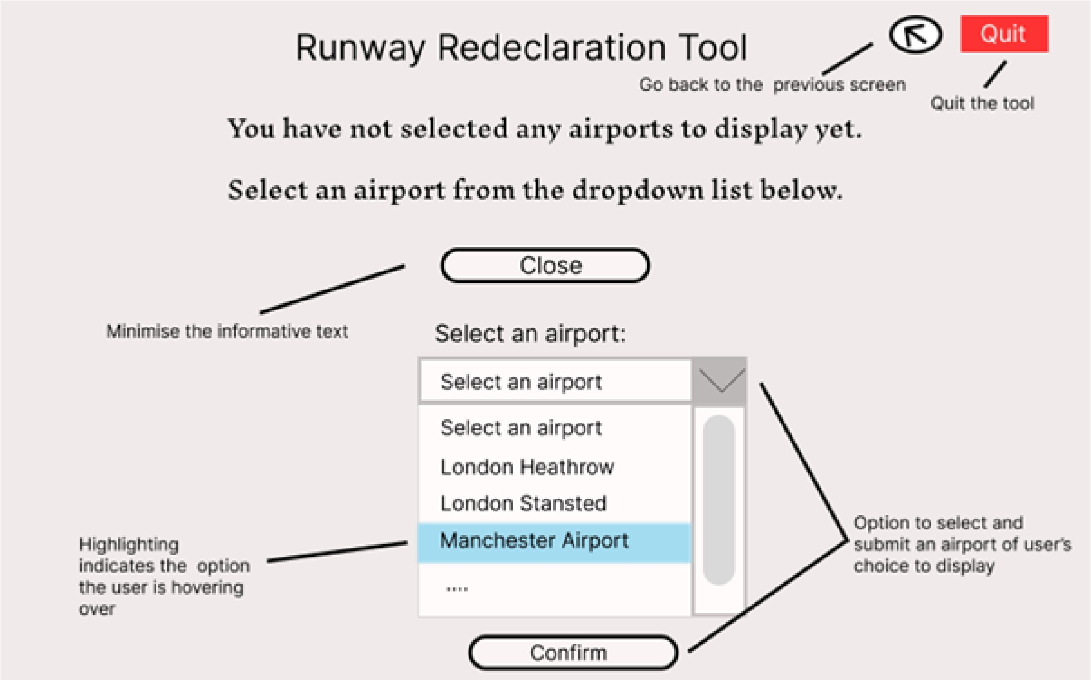
There are no airports available in the system which can be displayed. Please do one of the following so that a runway can be displayed:

- Select an airport from the dropdown list. This list contains all the airports that have been defined in the system already.
- Add an airport yourself by submitting a form that involves all the necessary information to display its runways.

Button linking this screen to the airport selection list screen

Button linking this screen to the airport addition via form screen

Figure 2: Authenticated user picking an airport for the first time



Runway Redeclaration Tool

You have not selected any airports to display yet.

Select an airport from the dropdown list below.

Select an airport:

Go back to the previous screen

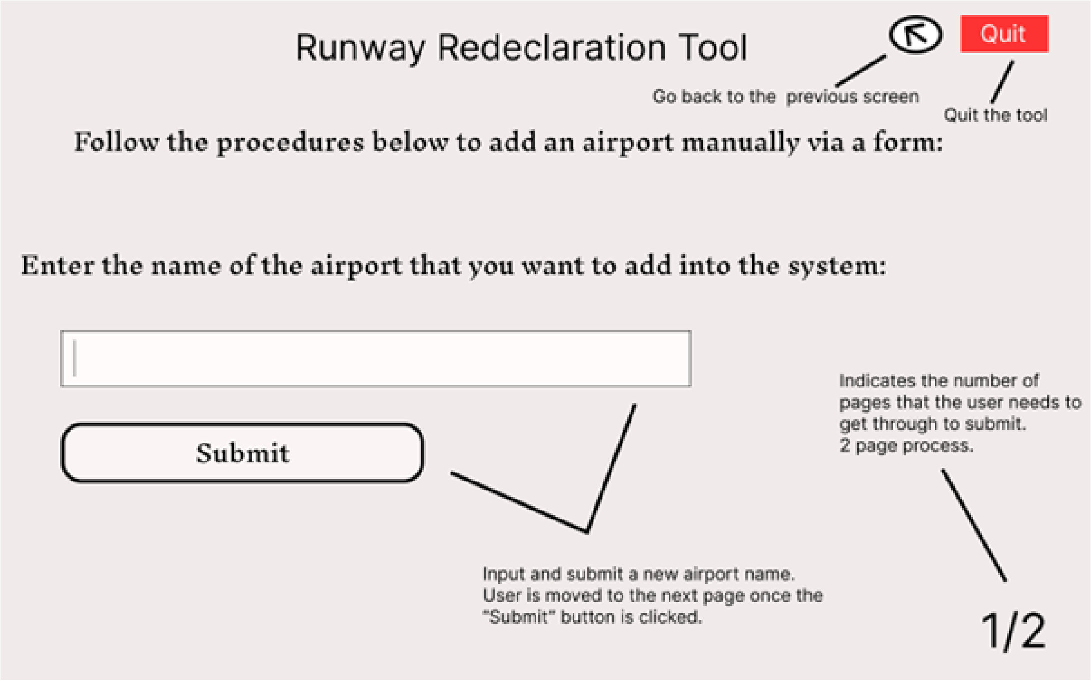
Quit the tool

Minimise the informative text

Highlighting indicates the option the user is hovering over

Option to select and submit an airport of user's choice to display

Figure 3: Picking an airport from the dropdown list



Runway Redeclaration Tool

Follow the procedures below to add an airport manually via a form:

Enter the name of the airport that you want to add into the system:

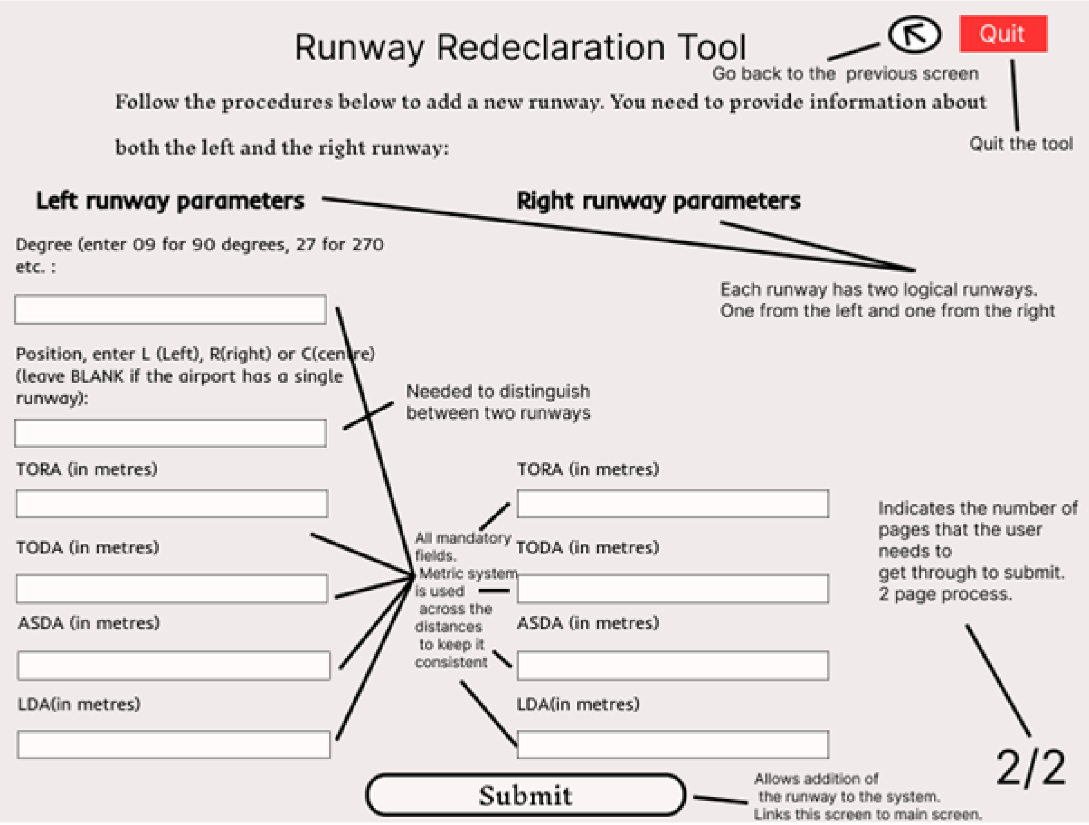
Go back to the previous screen

Quit the tool

Indicates the number of pages that the user needs to get through to submit. 2 page process.

Input and submit a new airport name. User is moved to the next page once the "Submit" button is clicked.

Figure 4: Adding an airport via form 1/2



Runway Redeclaration Tool

Follow the procedures below to add a new runway. You need to provide information about both the left and the right runway:

Each runway has two logical runways. One from the left and one from the right

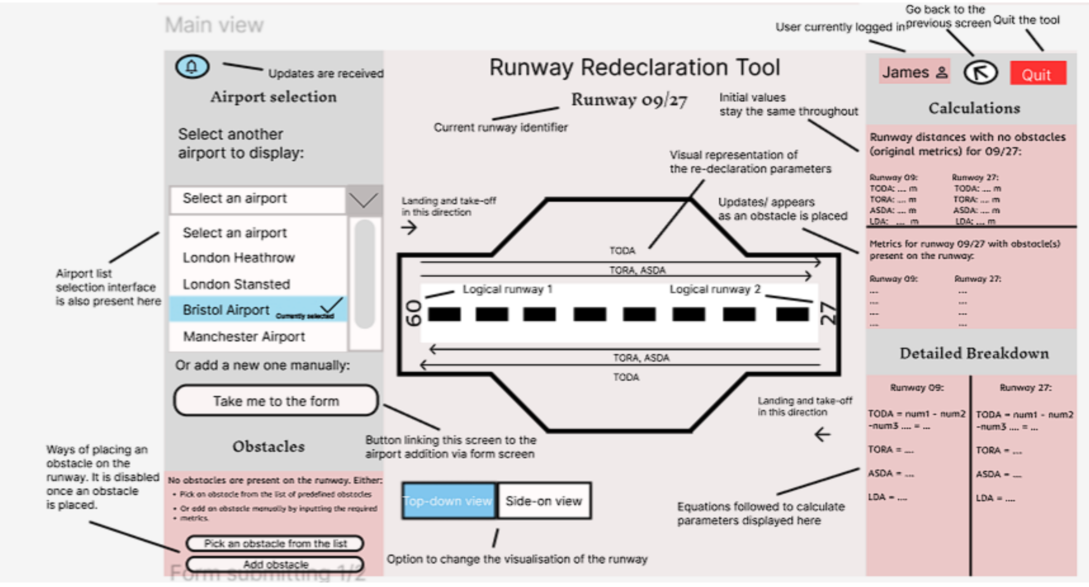
Needed to distinguish between two runways

All mandatory fields. Metric system is used across the distances to keep it consistent

Indicates the number of pages that the user needs to get through to submit. 2 page process.

Allows addition of the runway to the system. Links this screen to main screen.

Figure 5: Inputting the required parameters to add runways 2/2



Updates are received

Select another airport to display:

Or add a new one manually:

Airport list selection interface is also present here

Button linking this screen to the airport addition via form screen

Ways of placing an obstacle on the runway. It is disabled once an obstacle is placed.

No obstacles are present on the runway. Either:

- Pick an obstacle from the list of predefined obstacles
- Or add an obstacle manually by inputting the required metrics.

Current runway identifier

Initial values stay the same throughout

Visual representation of the re-declaration parameters

Updates/ appears as an obstacle is placed

Option to change the visualisation of the runway

Equations followed to calculate parameters displayed here

User currently logged in

Go back to the previous screen

Quit the tool

Runway distances with no obstacles (original metrics) for 09/27:

Runway 09:	Runway 27:
TODA: .... m	TODA: .... m
TORA: .... m	TORA: .... m
ASDA: .... m	ASDA: .... m
LDA: .... m	LDA: .... m

Metrics for runway 09/27 with obstacle(s) present on the runway:

Runway 09:	Runway 27:
...	...
...	...
...	...
...	...

Runway 09:	Runway 27:
TODA = num1 - num2 -num3 .... = ...	TODA = num1 - num2 -num3 .... = ...
TORA = ...	TORA = ...
ASDA = ...	ASDA = ...
LDA = ...	LDA = ...

Figure 6: Inputting the required parameters to add runways

1.2 Increment 1 Sprint Backlog - Missing actual hours and Integration

We had not included an ”actual hours” column in our previous sprint backlog, and adding this will allow us to see how we did over time in terms of successfully estimating the time it would take for each task to be completed. We did not spend that much time on testing because there was minimal implementation that could be tested.

Across the board, we underestimated the hours for each task, as highlighted in the report for the previous deliverable.

Additionally, an additional row, representing the numbers of hours it took us to resolve conflicts and put everything together, is included under the name ”Integration”.

Task	**Estimation**	**Owner**	**Actual Hours**
1: UML Design - MVC Structure	4h	All	5h
2: Import airport runway from XML	**8h**	Louis **(Scrum master)**	**10h**
2.1: Create airport objects in XML	4h		5h
2.2: Parse XML	2h		2h
2.3: Initialise airports	2h		3h
3: Perform calculations on distances	**6h**	Hossam	**7h**
3.1: Automatically trigger calculations	1h		2h
3.2: Give option to expose calculations to user	3h		3h
3.3: Log calculations to txt file	2h		2h
4: Basic GUI	**7h**	Abdullah	**10h**
4.1: Drop down to choose airports	1h		2h
4.2: Form to create airport	1h		3h
4.3: Runway and empty distances rendering	5h		5h
5: Basic obstacle placement	**4h**	Andy	**5h**
5.1: Link obstacle placement to calculations	4h		5h
6: Creating test cases	**5h**	Eren	**3h**
6.1: Testing calculation validity	2h		1h
6.2: Testing initialization	3h		2h
7: Integration	**-**	All	2h

Table 1: Task Ownership Table

1.3 UML Diagram Improvements

Our updated UML diagram, now with additional details and a use-case diagram, is included in the ”Key Design Artifacts” section.

2 Key Design Artifacts

2.1 UML Diagram

2.1.1 Updated UML Diagram

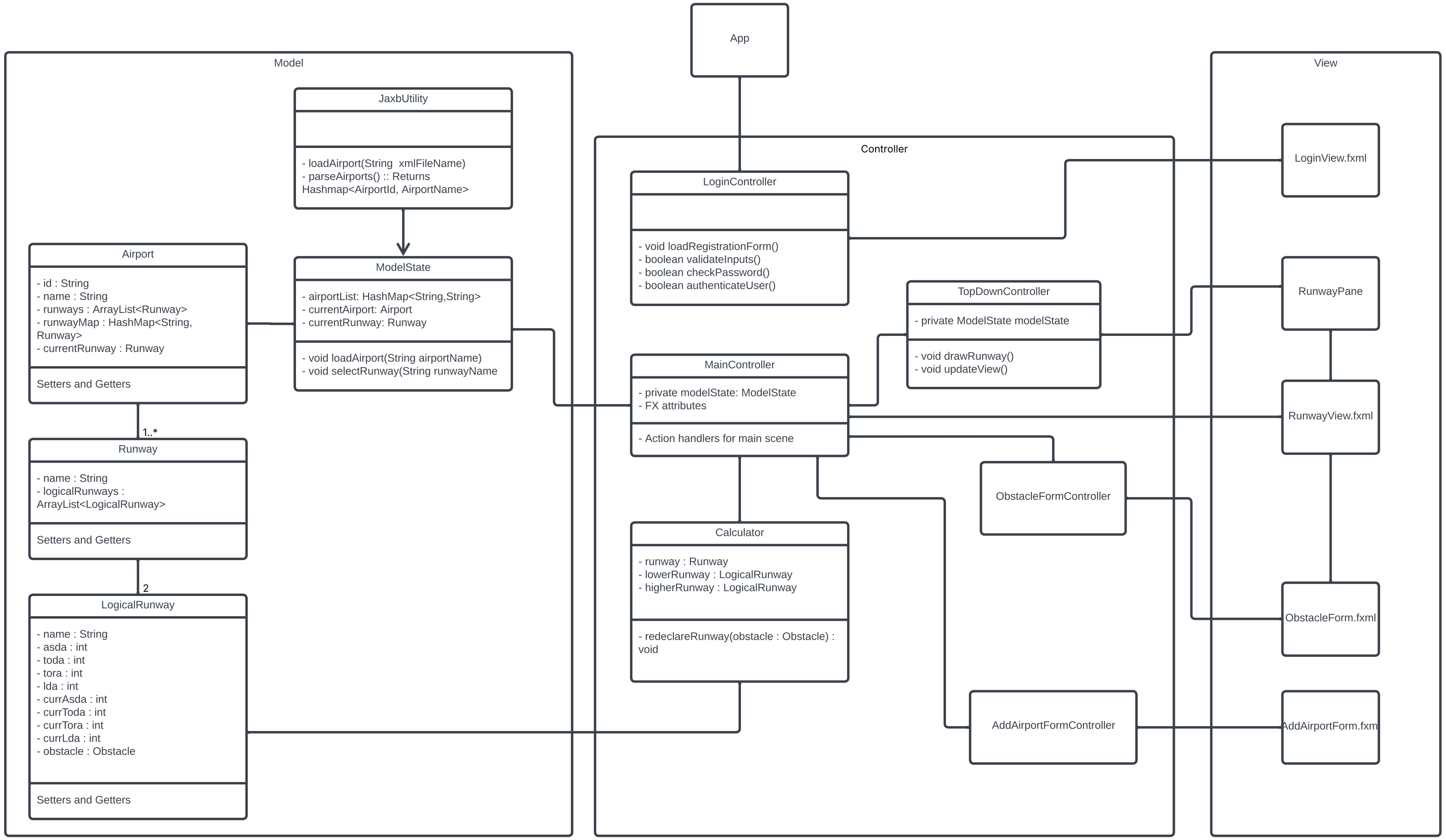
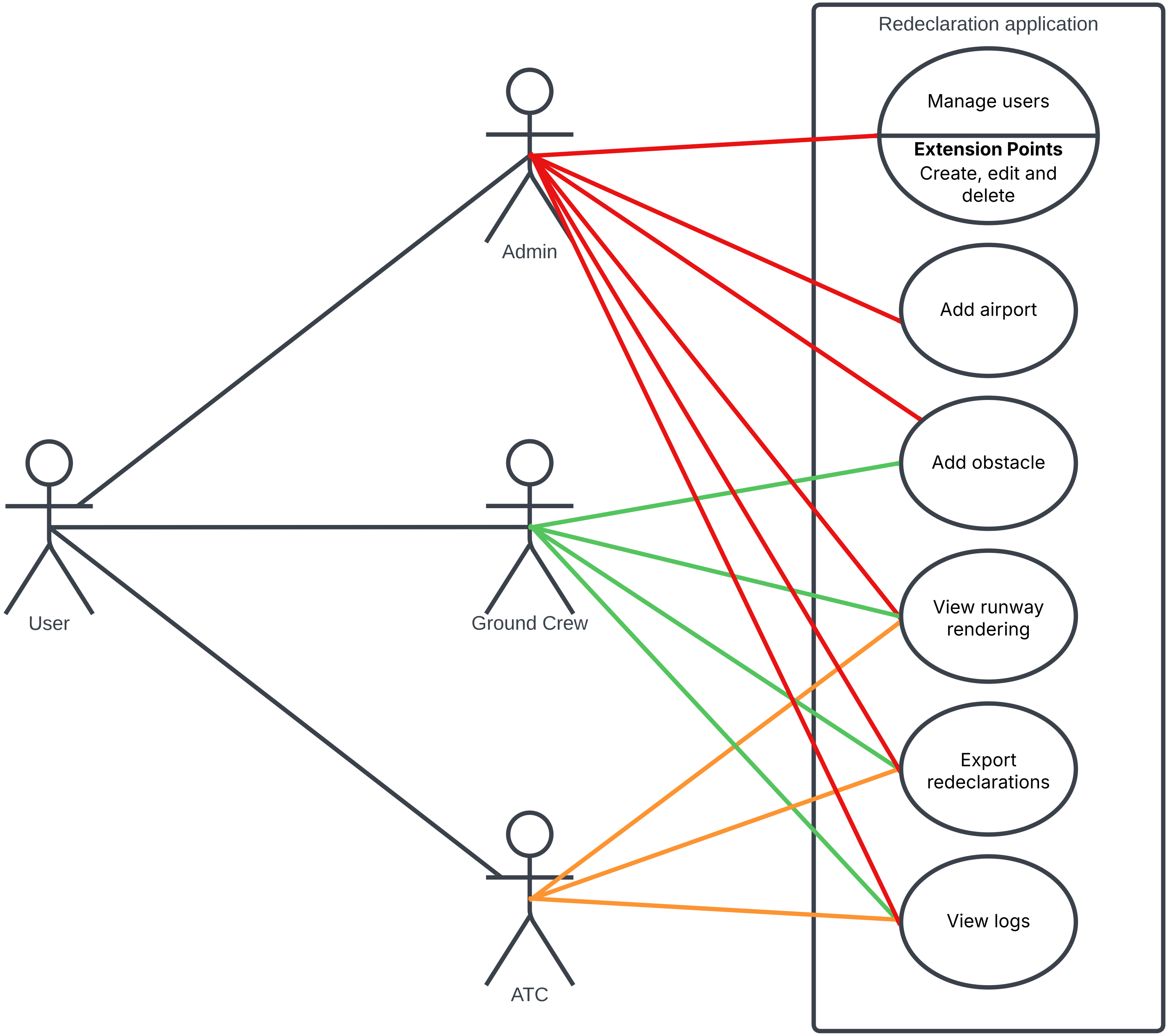

App
Model
JaxbUtility
- loadAirport(String xmlFileName)
- parseAirports() :: Returns Hashmap<AirportId, AirportName>
ModelState
- airportList: HashMap<String,String>
- currentAirport: Airport
- currentRunway: Runway
- void loadAirport(String airportName)
- void selectRunway(String runwayName
Airport
- id : String
- name : String
- runways : ArrayList<Runway>
- runwayMap : HashMap<String, Runway>
- currentRunway : Runway
Setters and Getters
1..*
Runway
- name : String
- logicalRunways : ArrayList<LogicalRunway>
Setters and Getters
2
LogicalRunway
- name : String
- asda : int
- toda : int
- tora : int
- lda : int
- currAsda : int
- currToda : int
- currTora : int
- currLda : int
- obstacle : Obstacle
Setters and Getters
Controller
LoginController
- void loadRegistrationForm()
- boolean validateInputs()
- boolean checkPassword()
- boolean authenticateUser()
TopDownController
- private ModelState modelState
- void drawRunway()
- void updateView()
MainController
- private modelState: ModelState
- FX attributes
- Action handlers for main scene
ObstacleFormController
Calculator
- runway : Runway
- lowerRunway : LogicalRunway
- higherRunway : LogicalRunway
- redeclareRunway(obstacle : Obstacle) : void
AddAirportFormController
View
LoginView.fxml
RunwayPane
RunwayView.fxml
ObstacleForm.fxml
AddAirportForm.fxm

Figure 7: UML Diagram with all the new additions

2.1.2 Use Case Diagram



2.2 Database

2.2.1 Database Structure Thought Process

The database design is centred around two primary entities that need to be stored within the application: Users and Airports. Consequently, the initial database structure consists of two main tables. Users will have attributes for username and password to facilitate authentication, as well as a role to manage access levels. Additionally, since each user must be associated with a specific airport, the Users table will reference the Airports table through a foreign key.

Regarding Airports, each airport can accommodate multiple runways, and each runway is further divided into more than one logical runways. To accurately represent this structure, two additional tables are derived from the Airports table, resulting in a total of four tables in the database. Both the Runways and LogicalRunways tables reference the Airports table.

This design removes redundancy and the foreign key constraints ensure referential integrity. E.g., if an airport goes defunct, all associated users (workers) will be automatically removed from the database, along with any related runways, maintaining data consistency and preventing orphaned records.

2.2.2 ER Diagram for the database

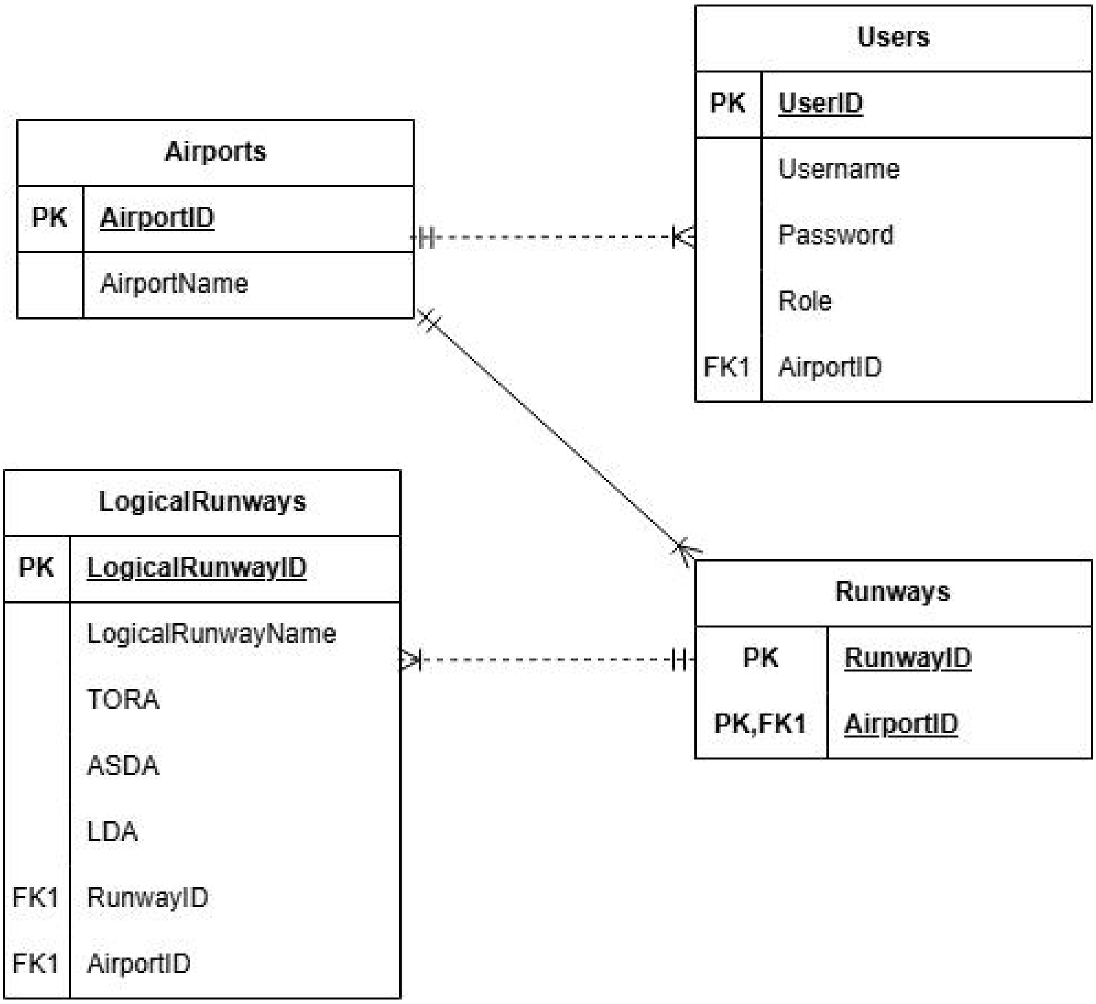


Figure 9: ER Diagram for the database of our tool

2.3 Storyboards, Scenarios

Our storyboards and scenarios were designed with the entire app structure in mind, envisioning the final product. As a result, they align well with both our current implementation and future plans, requiring no major changes or additions. They are available in the previous documentations.

3 Testing

3.1 Unit testing

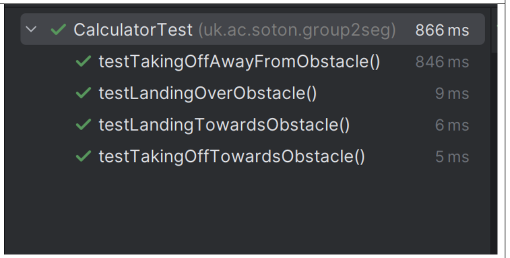
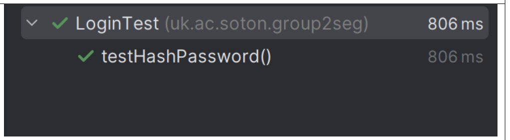
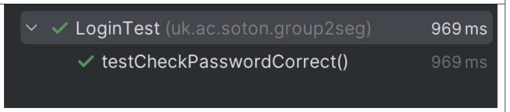
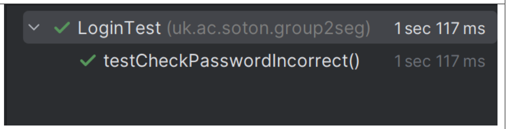
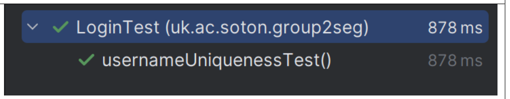
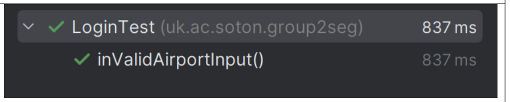
What we were testing	Method of Test	Expected Result	Results of Test	Evidence
Testing Calculations According to Project Specs	JUnit tests for LDA, TORA, TODA, ASDA under different obstacle placements	Calculated values should match expected values from project specifications	As expected	 ✓ CalculatorTest (uk.ac.soton.group2seg) 866 ms ✓ testTakingOffAwayFromObstacle() 846 ms ✓ testLandingOverObstacle() 9 ms ✓ testLandingTowardsObstacle() 6 ms ✓ testTakingOffTowardsObstacle() 5 ms
Password hashing	JUnit test for password hashing method	Hashed password should not be null and should differ from plain text	As expected	 ✓ LoginTest (uk.ac.soton.group2seg) 806 ms ✓ testHashPassword() 806 ms
Correct password verification	JUnit test for password checking method	Correct password should match stored hash	As expected	 ✓ LoginTest (uk.ac.soton.group2seg) 969 ms ✓ testCheckPasswordCorrect() 969 ms
Incorrect password rejection	JUnit test for password checking with wrong input	Incorrect password should not match stored hash	As expected	 ✓ LoginTest (uk.ac.soton.group2seg) 1 sec 117 ms ✓ testCheckPasswordIncorrect() 1 sec 117 ms
Username uniqueness detection	JUnit test querying database for existing username	System should detect non-unique username	As expected	 ✓ LoginTest (uk.ac.soton.group2seg) 878 ms ✓ usernameUniquenessTest() 878 ms
Invalid airport input handling	JUnit test with a non-existent airport code	System should reject invalid input	As expected	 ✓ LoginTest (uk.ac.soton.group2seg) 837 ms ✓ inValidAirportInput() 837 ms

Table 2: JUnit Test Table

3.2 Acceptance testing

User Story	**Test**	**Result**
As ATC I want to be able to check the validity of calculations to maintain safety	Calculation breakdowns can be viewed and are valid	Pass
As a user I want to see obstacles on the runway when they cause a redeclaration	Obstacles are rendered around the runway	Pass
As ATC I want to see available distances on the screen	Distance lines are rendered to scale on the screen	Pass
As ground crew I want to be able to report obstructions on the runway	An obstacle can be added to the runway strip	Pass
As a user I want to be able to add a new airport	Airports can be added from within the application	Pass
As a regulator I want an error log so I can check system integrity	Error logs are generated and accessible	Pass
As a regulator I want calculation logs so I can check system validity	Calculation logs are generated and accessible	Pass
As ground crew I want predefined obstacles list so I can quickly add an obstacle to the system	A predefined list of obstacles is available for selection	Pass
As ATC I want to see distances for each runway separately to ensure I only see relevant information	Distances are displayed separately for each runway	Pass

Table 3: Acceptance testing. See appendices for evidence

4 Planning

4.1 Progress for Increment 2

4.1.1 Sprint Backlog with actual hours and integration

Task	Estimation	Owner	Actual Hours
1: Predefined obstacle list	**6h**	Abdullah	**3h**
1.1: Implement obstacle dropdown	2h		1h
1.2: Implement obstacle placement using dropdown	4h		2h
2: Runway visualisation	**14h**	Louis	**11h**
2.1: Render cleared and graded area	4h		2h
2.2: Add lines to scale for declared distances	4h		6h
2.3: Lines update on obstacle placement	2h		1h
2.4: Rendering obstacles	4h		2h
3: Implementing multi-user login	**15h**	Eren & Hossam	**11h**
3.1: Design user database	5h		4h
3.2: Store users with roles (e.g. admin, viewer)	2h		1h
3.3: Design login page	4h		3h
3.4: Implement login function	4h		3h
4: Logging	**4h**	Hossam (**Scrum Master**)	**4h**
4.1: Logging calculations to a txt file	2h		2h
4.2: Debug logging	2h		2h
5: Notifications	**6h**	Andy	**6h**
5.1: Incorrect input popup box	2h		2h
5.2: Invalid password popup	2h		2h
5.3: Invalid obstacle popup	2h		2h
6: Integration	**2h**	Everyone	1h

Table 4: Task Ownership Table

4.1.2 Burndown criteria for tasks

We ensured that the following criteria were met before removing a task from the burndown chart:

- Comprehensive test coverage that take into account many scenarios, with all the tests passing.
- The development team agrees that any code written for related tasks (if applicable) can be easily integrated.
- The customer is satisfied with the implementation(product) and believes it delivers value.

4.1.3 Complete Burndown Chart for Increment 2

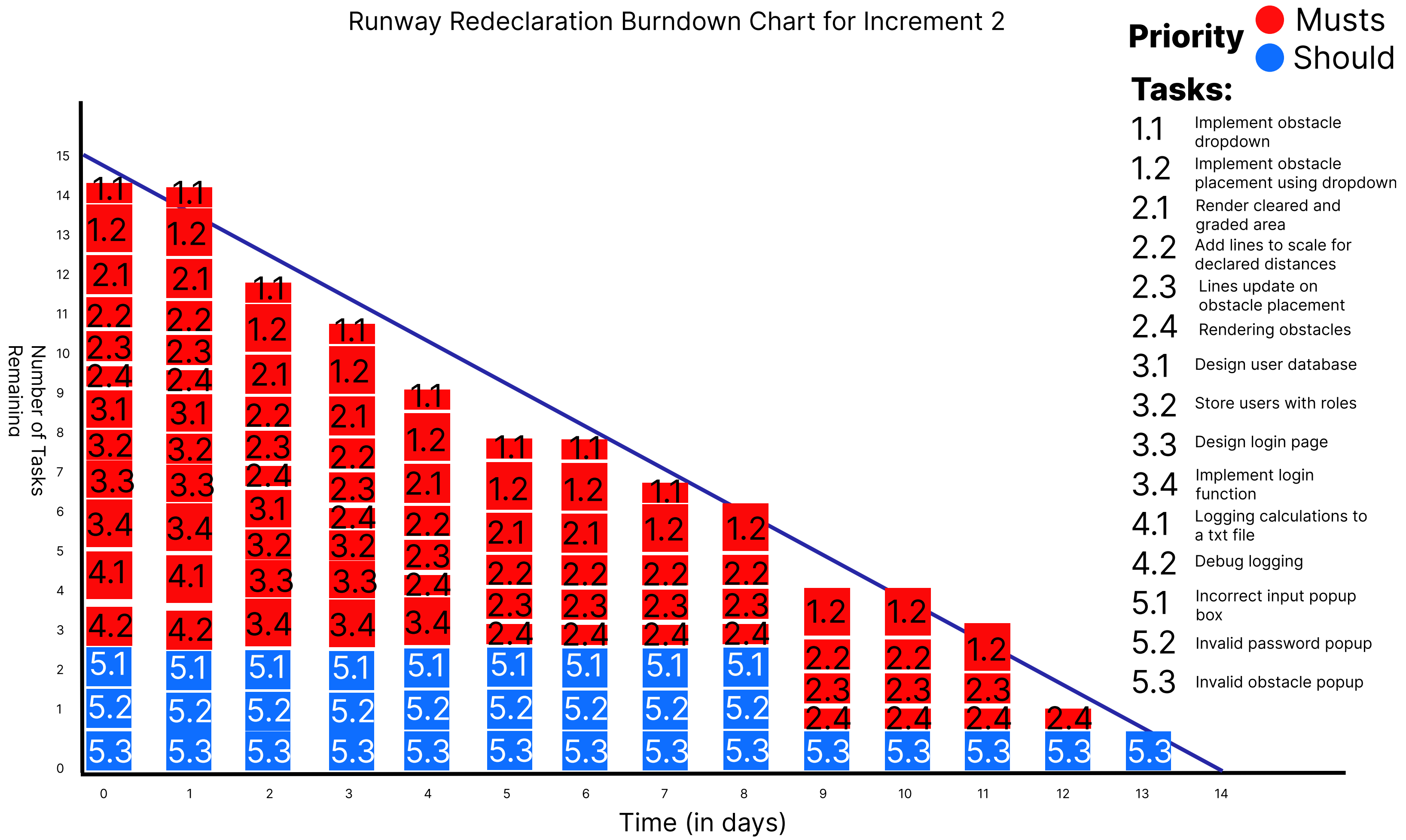


Figure 10: Chart for tracking task completion

4.2 Sprint 3 Plan

4.2.1 Retrospective on Sprint 2:

In Sprint 2, we finally moved past the design phase, which meant everyone could focus purely on coding without having to revisit old decisions. Since the core features were already in place, building on top of them felt much easier and more natural. We did encounter some challenges when integrating different parts of the system, but team discussions and debugging sessions helped us to work through them. Testing became more important at this stage, as we started catching edge cases that needed fixing. On the bright side, communication and collaboration improved greatly and task distribution was much smoother, making the whole sprint run more efficiently.

4.2.2 Goal

The main target for sprint 3 is to implement the final bits of functionality that we are yet to implement from our product backlog e.g., side-on visualisation. Once all core functionality is in place, we will develop a comprehensive user guide detailing existing issues (if any) and providing solutions to potential problems users may encounter while using the product.

Additionally, since no tasks have been carried over to this sprint and we have more time than usual, we will take the opportunity to refine and enhance the user interface. We will also explore the possibility of implementing extensions to further improve the user experience.

4.2.3 Sprint Backlog

Task	Owner	Estimation	Actual Hours
1: Admin dashboard back-end	Eren (**Scrum master**)	**10h**	
1.1: SQL queries to edit and add users		5h	
1.2: Modifying access levels based on roles		5h	
2: Finalise views	Louis	**10h - 14h**	
2.1: Debug top down view		2h	
2.2: Implement side on view		8h	
2.3: (EXTENSION) Pan and zoom views		4h	
3: Help documentation	Andy	**10h**	
3.1: Writing documentation		5h	
3.2: Implementing documentation within application		5h	
4: Information exporting	Abdullah	**14h**	
4.1: Exporting airport and obstacle details as XML		7h	
4.2: Exporting visualisations and calculations in PDF		7h	
5: General graphics and UI improvements	Hossam	**9h-11h**	
5.1: Improve usability		2h	
5.2: User(role)-specific application views		7h	
5.3: (EXTENSION) Colour blindness views		3h	
6: Integration	All	4h	

Table 5: Task Ownership Table for Increment 3

4.2.4 Day 0 Burndown Chart for Increment 3

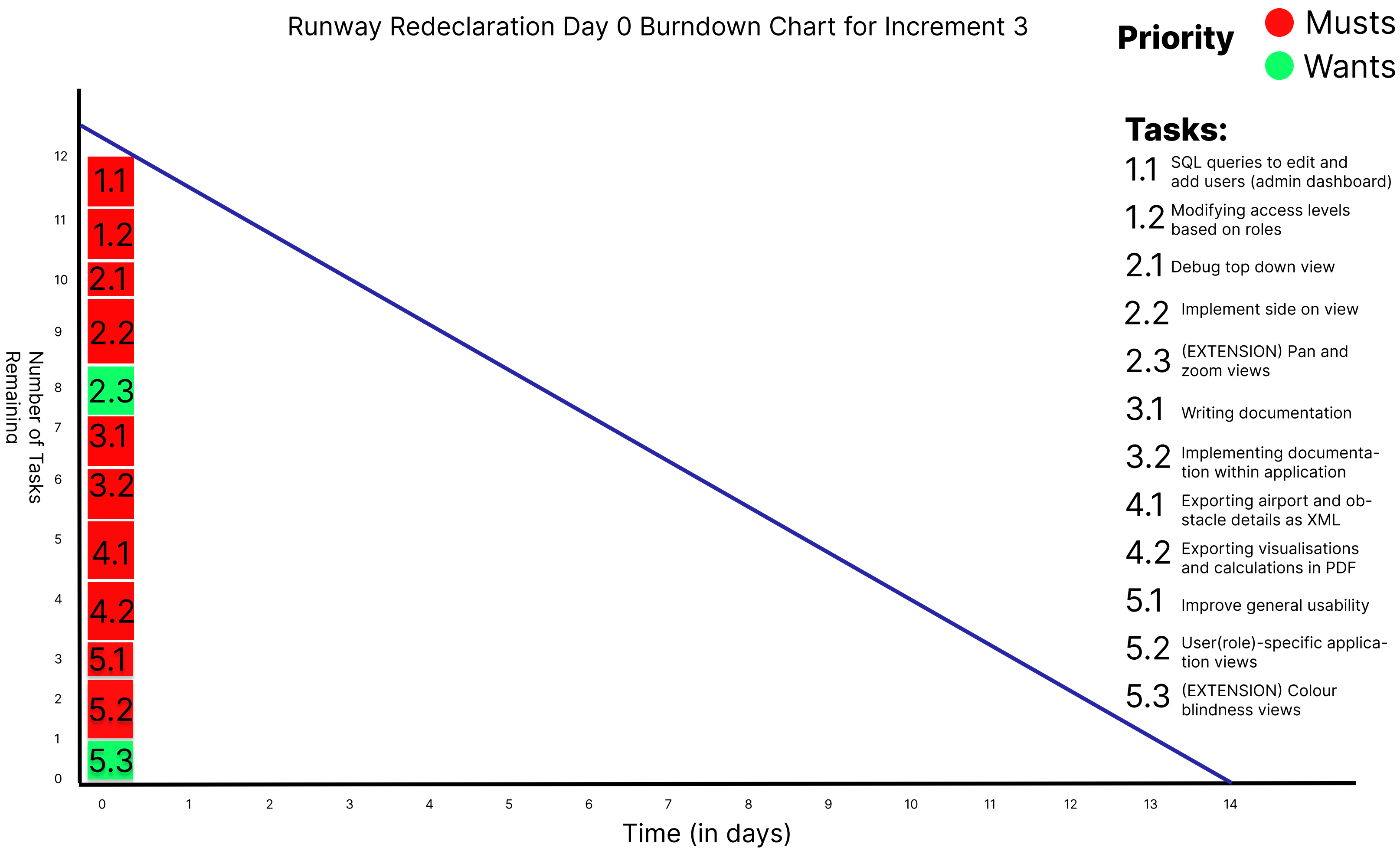
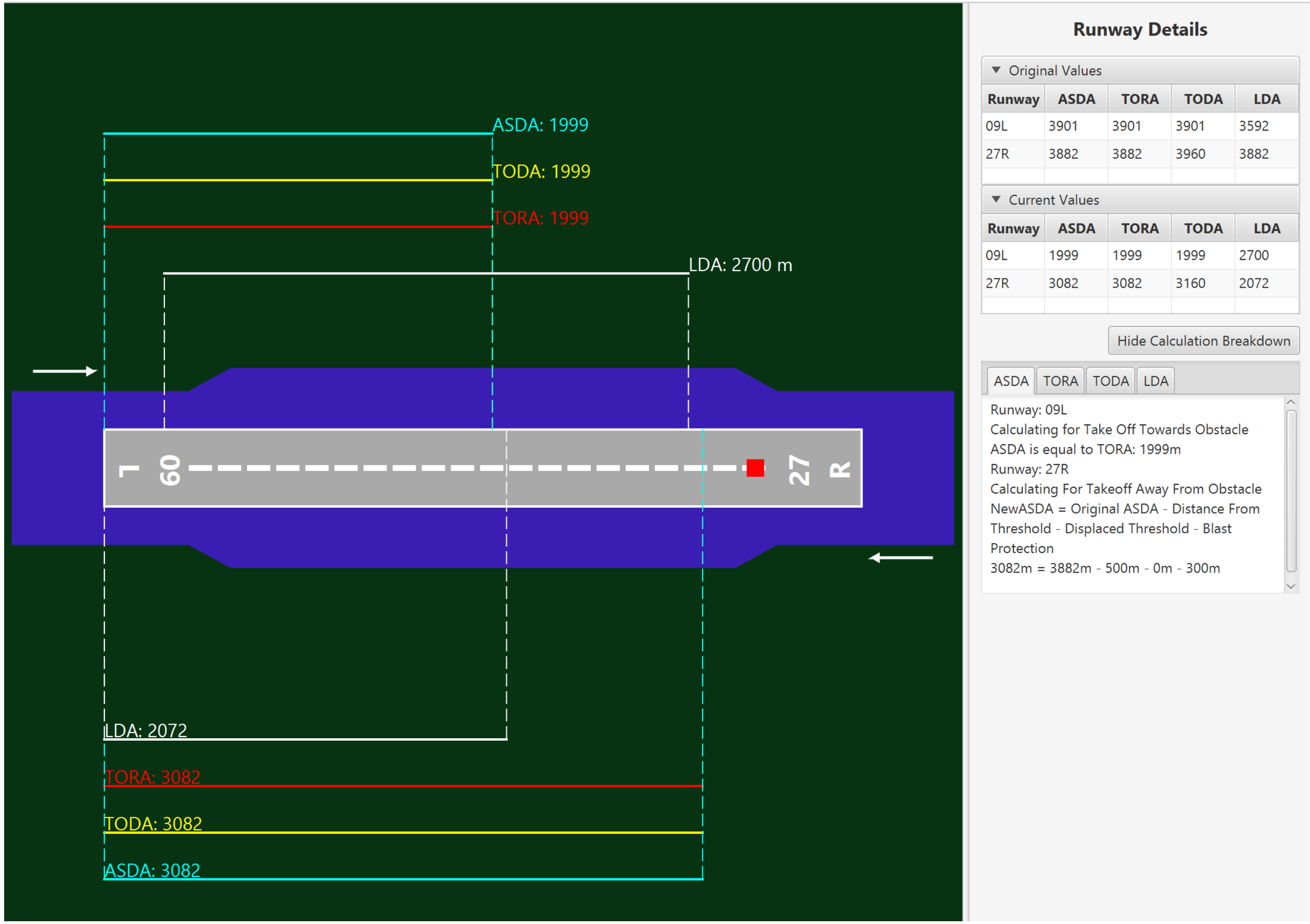


Figure 11: Increment 3 Day 0 Chart

A Appendix A: Acceptance testing evidence



Runway Details

Original Values

Runway	ASDA	TORA	TODA	LDA
09L	3901	3901	3901	3592
27R	3882	3882	3960	3882

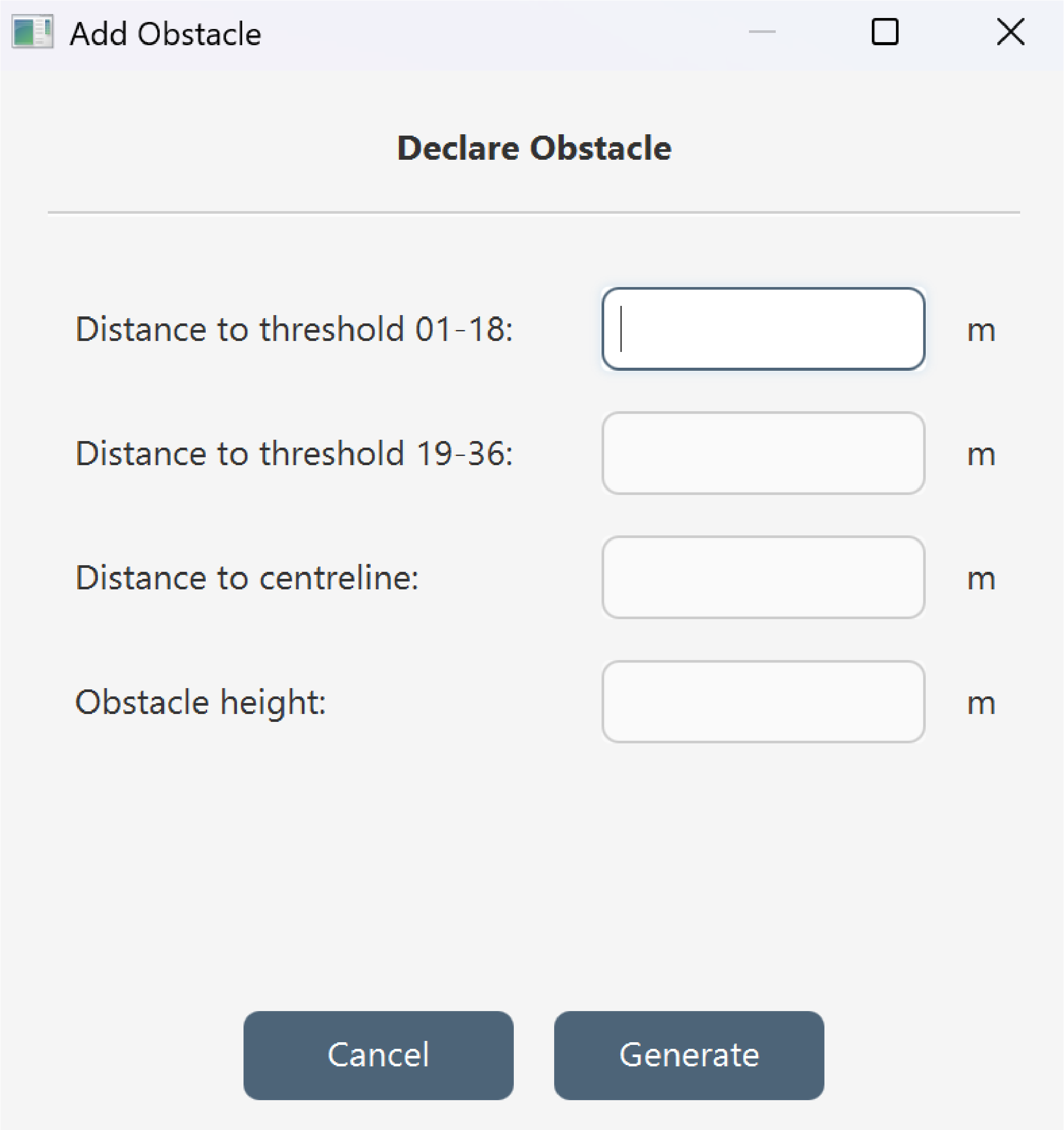
Current Values

Runway	ASDA	TORA	TODA	LDA
09L	1999	1999	1999	2700
27R	3082	3082	3160	2072

Runway: 09L
Calculating for Take Off Towards Obstacle
ASDA is equal to TORA: 1999m
Runway: 27R
Calculating For Takeoff Away From Obstacle
NewASDA = Original ASDA - Distance From Threshold - Displaced Threshold - Blast Protection
3082m = 3882m - 500m - 0m - 300m

Figure 12: Runway view after adding an obstacle

This view shows how distance lines are shown graphically, as well as recalculated. Furthermore, the calculation breakdown is visible on the right side of the screen



Add Obstacle

Declare Obstacle

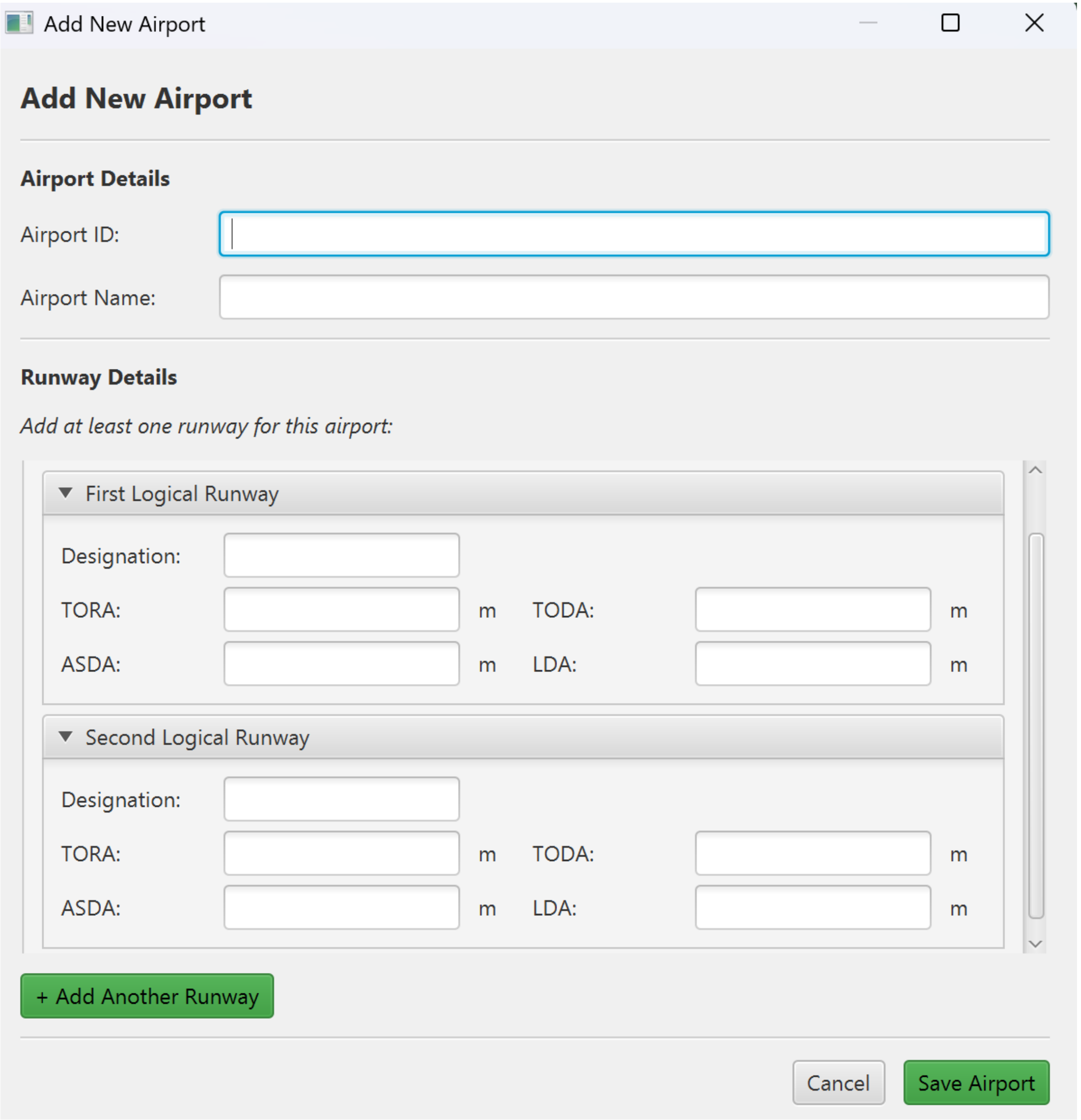
Distance to threshold 01-18: m

Distance to threshold 19-36: m

Distance to centreline: m

Obstacle height: m

Figure 13: Obstacle adding form



Add New Airport

Add New Airport

Airport Details

Airport ID:

Airport Name:

Runway Details

Add at least one runway for this airport:

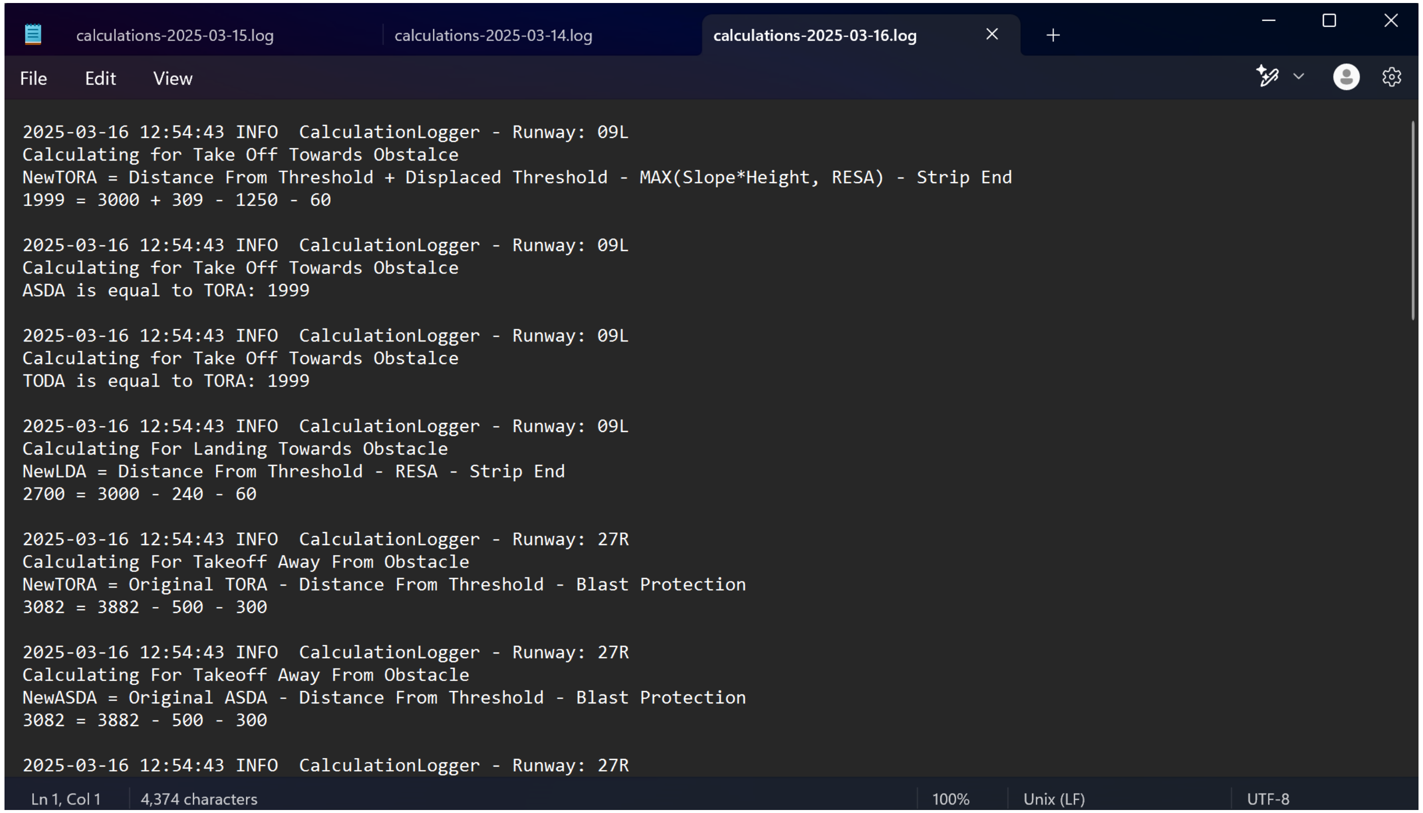
▼ First Logical Runway

Designation:					
TORA:		m	TODA:		m
ASDA:		m	LDA:		m

▼ Second Logical Runway

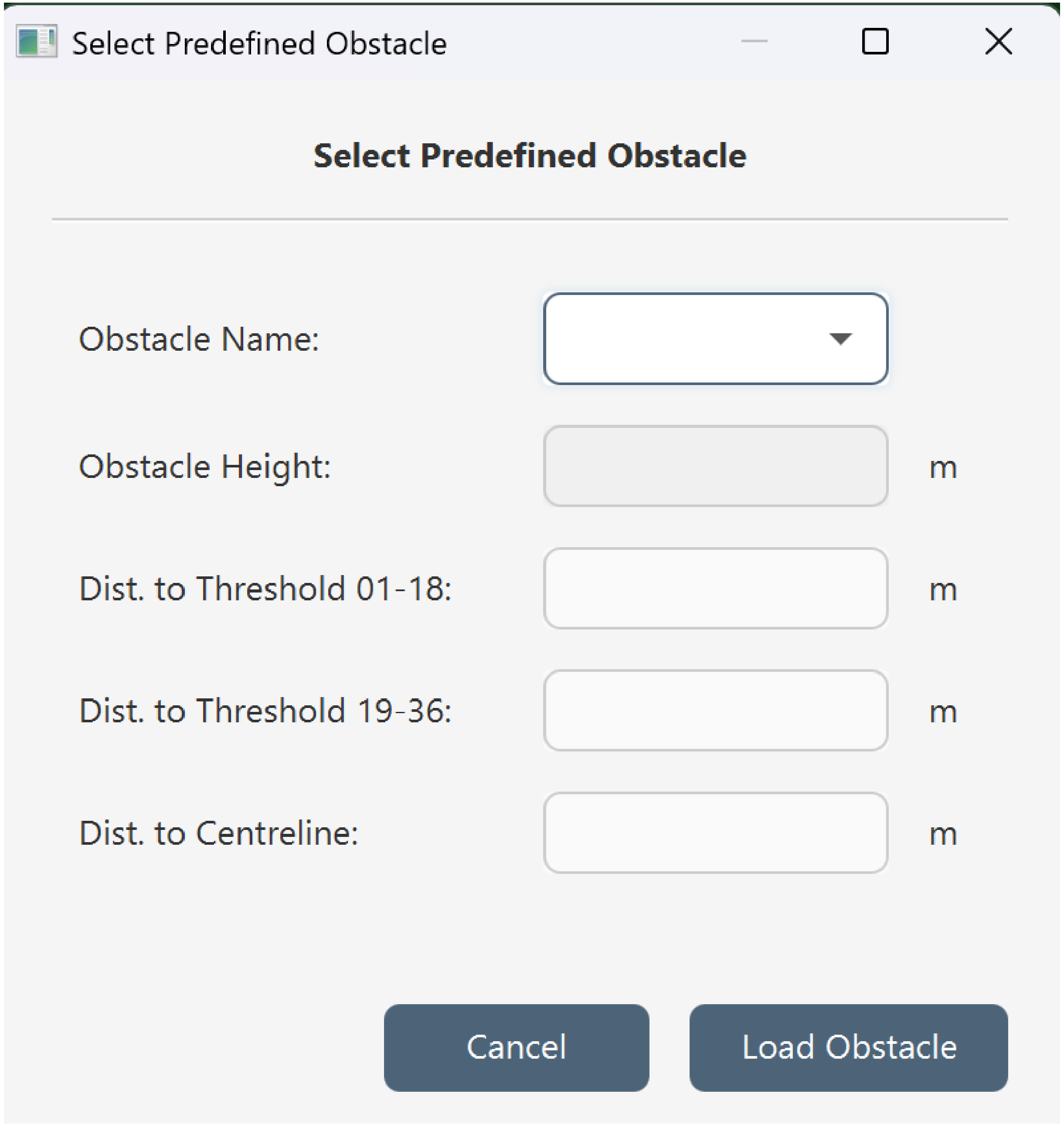
Designation:					
TORA:		m	TODA:		m
ASDA:		m	LDA:		m

Figure 14: Airport adding form


calculations-2025-03-15.log
calculations-2025-03-14.log
calculations-2025-03-16.log
File
Edit
View
Ln 1, Col 1
4,374 characters
100%
Unix (LF)
UTF-8

```
2025-03-16 12:54:43 INFO  CalculationLogger - Runway: 09L
Calculating for Take Off Towards Obstalce
NewTORA = Distance From Threshold + Displaced Threshold - MAX(Slope*Height, RESA) - Strip End
1999 = 3000 + 309 - 1250 - 60

2025-03-16 12:54:43 INFO  CalculationLogger - Runway: 09L
Calculating for Take Off Towards Obstalce
ASDA is equal to TORA: 1999

2025-03-16 12:54:43 INFO  CalculationLogger - Runway: 09L
Calculating for Take Off Towards Obstalce
TODA is equal to TORA: 1999

2025-03-16 12:54:43 INFO  CalculationLogger - Runway: 09L
Calculating For Landing Towards Obstacle
NewLDA = Distance From Threshold - RESA - Strip End
2700 = 3000 - 240 - 60

2025-03-16 12:54:43 INFO  CalculationLogger - Runway: 27R
Calculating For Takeoff Away From Obstacle
NewTORA = Original TORA - Distance From Threshold - Blast Protection
3082 = 3882 - 500 - 300

2025-03-16 12:54:43 INFO  CalculationLogger - Runway: 27R
Calculating For Takeoff Away From Obstacle
NewASDA = Original ASDA - Distance From Threshold - Blast Protection
3082 = 3882 - 500 - 300

2025-03-16 12:54:43 INFO  CalculationLogger - Runway: 27R
```

Figure 15: Calculation Logs



Select Predefined Obstacle

Select Predefined Obstacle

Obstacle Name:

Obstacle Height: m

Dist. to Threshold 01-18: m

Dist. to Threshold 19-36: m

Dist. to Centreline: m

Figure 16: Predefined Obstacle Form