COMP2211
Software Engineering Group Project

Runway Re-declaration
Deliverable 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

University of Southampton

Electronics & Computer Science
University of Southampton
United Kingdom

Contents

Footnote: Cover page airplane graphic [1]

Deliverable 2: Increment 1

Introduction

Commercial airports are bustling places seeing the arrival and departure of numerous airplanes, making safe and efficient operation of runways the primary focus of airport operations. In certain situations, runways may be obstructed by objects such as other aircrafts, trees, etc. In such situations, a runway can still operate, albeit with altered parameters [2–4]. This document details the design process for the first deliverable of a Runway Dedeclaration App through scenarios, storyboards as well as UML diagrams. It contains screenshots of the app along with testing methods employed to ensure that the results provided by the application are accurate.

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

In this project we are basing the implementation on the model view controller design. We have implemented a main controller class called MainApplication that controls all the other classes like the models and the view.

For the model component of MVC design we included the classes: Airport, Runway, Obstacle, importXML and exportXML. All the model classes had a one-way association with the controller class as the model classes aren't supposed to be able to impact the controller class like in the MVC design.

The airport class has a one-way association with the runway class as the airport has access to the runway attributes and methods whereas the runway doesn't have access to the contents of the airport class. The same relation for the airport to runway classes is expressed for the relation between the runway class to obstacle class.

The XML parsing is done via the 2 model classes, importXML and exportXML. Both of these XML parsing classes have one way association to not only the controller, but all other model classes in the implementation. This is as the object creation is done via the controller as well as the XML parsing classes when the file is being read or written to.

For the view component of the MVC design we are implementing the following classes: MenuScene, LoginScene, AirportScene, AirportListScene, AnimatedPatternBackground, ObstacleListScene, RunwayConfigViewSceneand RunwayListScene. All the view classes have a one-way association with the controller class simulating the MVC design.

The controller class has an association relationship with the MenuScene class as both classes access the other class' fields and methods. The MenuScene class has a composition relationship with the LoginScene class as the MenuScene class has an instance of the LoginScene class.

The relationship between the LoginSCene class and the MenuScene class is also 1-to-1 as there only needs to be one LoginScene for a MenuScene and vice versa. The AirportScene has a one way association with the AirportListScene due to the varying access of the class' attributes and methods.

In this project we also decided to include classes that inherit from a parent class. For the display aspect as we realised that there would be repeating methods so we decided to implement inheritance here. The parent class is RunwayTopDown and the child class is the RunwayView class. We have also implemented the RunwaySideView class but are still deciding on whether the class should inherit from the RunwayTopDown class.

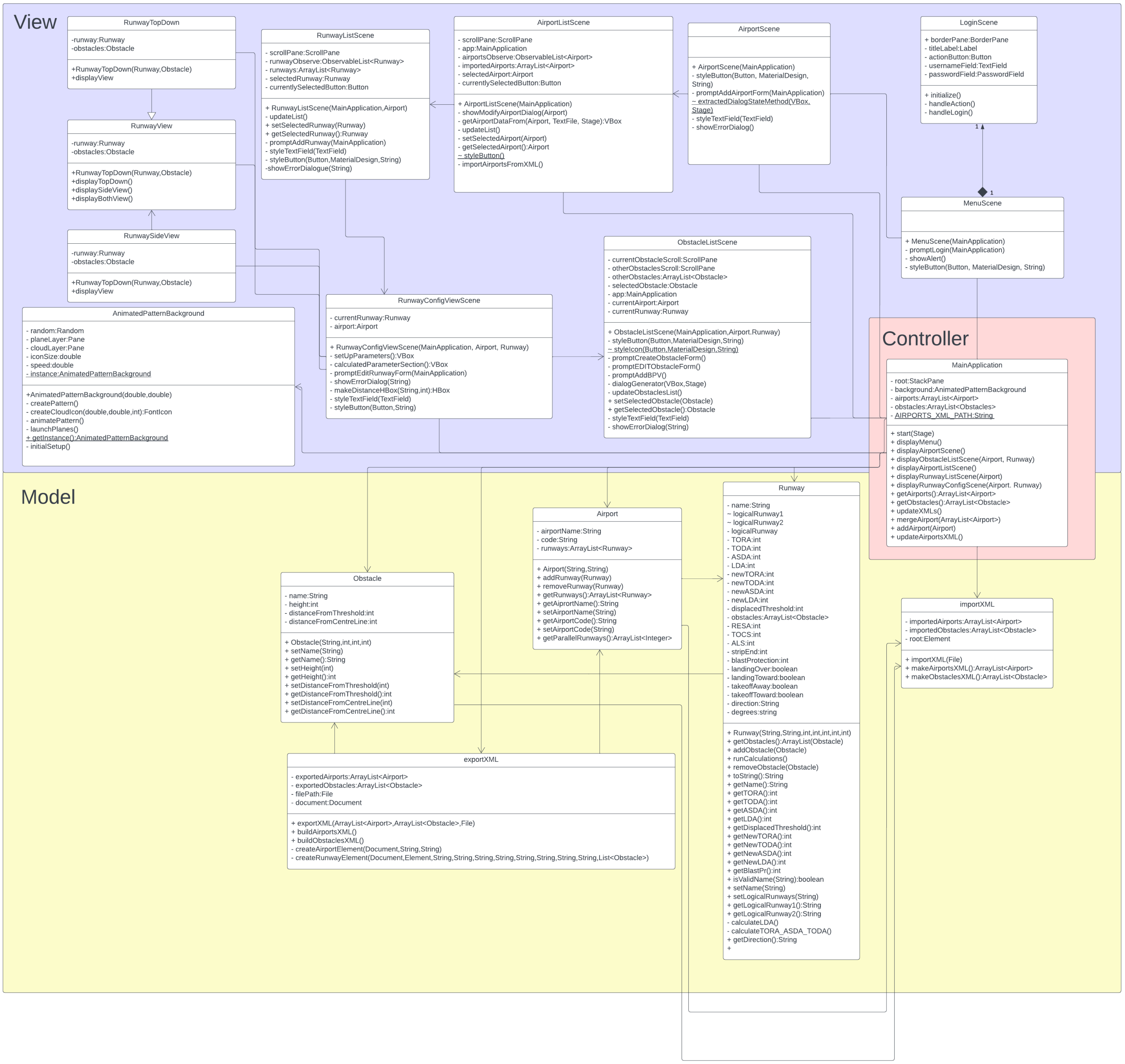

View
RunwayTopDown
-runway:Runway
-obstacles:Obstacle
+RunwayTopDown(Runway,Obstacle)
+displayView
RunwayListScene
- scrollPane:ScrollPane
- runwayObserve:ObservableList<Runway>
- runways:ArrayList<Runway>
- selectedRunway:Runway
- currentlySelectedButton:Button
+ RunwayListScene(MainApplication,Airport)
- updateList()
+ setSelectedRunway(Runway)
+ getSelectedRunway():Runway
- promptAddRunway(MainApplication)
- styleTextField(TextField)
- styleButton(Button,MaterialDesign,String)
-showErrorDialogue(String)
AirportListScene
- scrollPane:ScrollPane
- app:MainApplication
- airportsObserve:ObservableList<Airport>
- importedAirports:ArrayList<Airport>
- selectedAirport:Airport
- currentlySelectedButton:Button
+ AirportListScene(MainApplication)
- showModifyAirportDialog(Airport)
- getAirportDataFrom(Airport, TextFile, Stage):VBox
- updateList()
- setSelectedAirport(Airport)
- getSelectedAirport():Airport
~ styleButton()
- importAirportsFromXML()
AirportScene
+ AirportScene(MainApplication)
- styleButton(Button, MaterialDesign, String)
- promptAddAirportForm(MainApplication)
~ extractedDialogStateMethod(VBox, Stage)
- styleTextField(TextField)
- showErrorDialog()
LoginScene
+ borderPane:BorderPane
- titleLabel:Label
- actionButton:Button
- usernameField:TextField
- passwordField:PasswordField
+ initialize()
- handleAction()
- handleLogin()
1
1
RunwayView
-runway:Runway
-obstacles:Obstacle
+RunwayTopDown(Runway,Obstacle)
+displayTopDown()
+displaySideView()
+displayBothView()
MenuScene
+ MenuScene(MainApplication)
- promptLogin(MainApplication)
- showAlert()
- styleButton(Button, MaterialDesign, String)
RunwaySideView
-runway:Runway
-obstacles:Obstacle
+RunwayTopDown(Runway,Obstacle)
+displayView
ObstacleListScene
- currentObstacleScroll:ScrollPane
- otherObstaclesScroll:ScrollPane
- otherObstacles:ArrayList<Obstacle>
- selectedObstacle:Obstacle
- app:MainApplication
- currentAirport:Airport
- currentRunway:Runway
+ ObstacleListScene(MainApplication,Airport.Runway)
- styleButton(Button,MaterialDesign,String)
~ styleIcon(Button,MaterialDesign,String)
- promptCreateObstacleForm()
- promptEDITObstacleForm()
- promptAddBPV()
- dialogGenerator(VBox,Stage)
- updateObstaclesList()
+ setSelectedObstacle(Obstacle)
+ getSelectedObstacle():Obstacle
- styleTextField(TextField)
- showErrorDialog(String)
AnimatedPatternBackground
- random:Random
- planeLayer:Pane
- cloudLayer:Pane
- iconSize:double
- speed:double
- instance:AnimatedPatternBackground
+AnimatedPatternBackground(double,double)
- createPattern()
- createCloudIcon(double,double,int):FontIcon
- animatePattern()
- launchPlanes()
+ getInstance():AnimatedPatternBackground
- initialSetup()
RunwayConfigViewScene
- currentRunway:Runway
- airport:Airport
+ RunwayConfigViewScene(MainApplication, Airport, Runway)
- setUpParameters():VBox
- calculatedParameterSection():VBox
- promptEditRunwayForm(MainApplication)
- showErrorDialog(String)
- makeDistanceHBox(String,int):HBox
- styleTextField(TextField)
- styleButton(Button,String)
Controller
MainApplication
- root:StackPane
- background:AnimatedPatternBackground
- airports:ArrayList<Airport>
- obstacles:ArrayList<Obstacles>
- AIRPORTS_XML_PATH:String
+ start(Stage)
+ displayMenu()
+ displayAirportScene()
+ displayObstacleListScene(Airport, Runway)
+ displayAirportListScene()
+ displayRunwayListScene(Airport)
+ displayRunwayConfigScene(Airport. Runway)
+ getAirports():ArrayList<Airport>
+ getObstacles():ArrayList<Obstacle>
+ updateXMLs()
+ mergeAirport(ArrayList<Airport>)
+ addAirport(Airport)
+ updateAirportsXML()
Model
Runway
- name:String
~ logicalRunway1
~ logicalRunway2
- logicalRunway
- TORA:int
- TODA:int
- ASDA:int
- LDA:int
- newTORA:int
- newTODA:int
- newASDA:int
- newLDA:int
- displacedThreshold:int
- obstacles:ArrayList<Obstacle>
- RESA:int
- TOCS:int
- ALS:int
- stripEnd:int
- blastProtection:int
- landingOver:boolean
- landingToward:boolean
- takeoffAway:boolean
- takeoffToward:boolean
- direction:String
- degrees:string
+ Runway(String,String,int,int,int,int,int)
+ getObstacles():ArrayList(Obstacle)
+ addObstacle(Obstacle)
+ runCalculations()
+ removeObstacle(Obstacle)
+ toString():String
+ getName():String
+ getTORA():int
+ getTODA():int
+ getASDA():int
+ getLDA():int
+ getDisplacedThreshold():int
+ getNewTORA():int
+ getNewTODA():int
+ getNewASDA():int
+ getNewLDA():int
+ getBlastPr():int
+ isValidName(String):boolean
+ setName(String)
+ setLogicalRunways(String)
+ getLogicalRunway1():String
+ getLogicalRunway2():String
- calculateLDA()
- calculateTORA_ASDA_TODA()
+ getDirection():String
+
Airport
- airportName:String
- code:String
- runways:ArrayList<Runway>
+ Airport(String,String)
+ addRunway(Runway)
+ removeRunway(Runway)
+ getRunways():ArrayList<Runway>
+ getAiprortName():String
+ setAirportName(String)
+ getAirportCode():String
+ setAirportCode(String)
+ getParallelRunways():ArrayList<Integer>
Obstacle
- name:String
- height:int
- distanceFromThreshold:int
- distanceFromCentreLine:int
+ Obstacle(String,int,int,int)
+ setName(String)
+ getName():String
+ setHeight(int)
+ getHeight():int
+ setDistanceFromThreshold(int)
+ getDistanceFromThreshold():int
+ setDistanceFromCentreLine(int)
+ getDistanceFromCentreLine():int
importXML
- importedAirports:ArrayList<Airport>
- importedObstacles:ArrayList<Obstacle>
- root:Element
+ importXML(File)
+ makeAirportsXML():ArrayList<Airport>
+ makeObstaclesXML():ArrayList<Obstacle>
exportXML
- exportedAirports:ArrayList<Airport>
- exportedObstacles:ArrayList<Obstacle>
- filePath:File
- document:Document
+ exportXML(ArrayList<Airport>,ArrayList<Obstacle>,File)
+ buildAirportsXML()
+ buildObstaclesXML()
- createAirportElement(Document,String,String)
- createRunwayElement(Document,Element,String,String,String,String,String,String,String,String,List<Obstacle>)

Figure 1: Class diagram Diagram

1.1.2 Use Case Diagram

The use case diagram represents the funcationality of the system from user's point of view. It's a structured texual description.

The use case diagram outlines the interactions between users (actors) and the system itself. It serves as a visual representation of the system's functionalities and the user's ability to interact with these functionalities, known as use cases.

We included all primary and one secondary stakeholders as the actors who directly interacts with the system. These includes:

- Airfield Operations Manager

- Airfield Safety Officer
- Air Traffic Controller

We made initial decision on their level of access to the system. With Airfield Operations Manager is the editor, Airfield Safety Officer is the admin, and Air Traffic Controller is the viewer. This allows us to determine which actions each stakeholders are able to perform and allows us to clearly see the relationships while developin our application.

The diagram illustrates the actions that they can perform within the system. It also shows different functionalities each stakeholders is able to access, helps to clarify the system's requirements and ensure that all potential interactions are accounted for during the design phase.

Also relationships between use cases are demonstrated through the 'includes/extends' labels. This indicates to us whether a use case **always includes** another use case in it's function, or if there is **sometimes** a variation of that use case.

The diagram also addresses error handling through extension points for invalid inputs, ensuring a robust design that anticipates user errors and provides appropriate feedback.

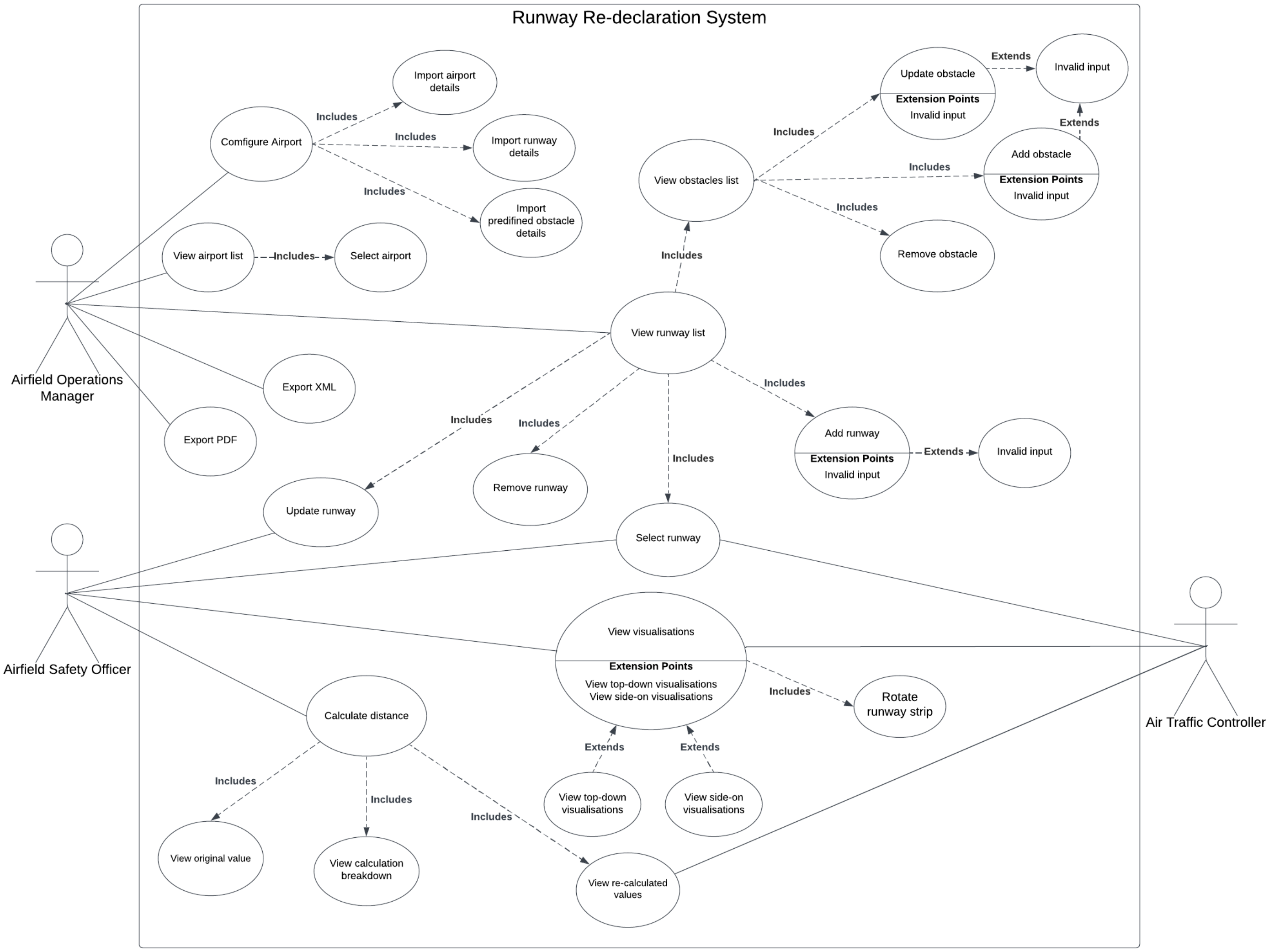


Figure 2: Use Case Diagram

1.1.3 State Machine Diagram

Below is our State Machine Diagram. We uses this diagram to help with visualising the states an entity can be in, as well as the transitions between these states. This is very useful as it can represent the behavior of systems, particularly when describing the behavior of objects in response to a series of events or conditions.

The **Airport Configuring** and **Calculating** states are broken down into smaller sub-states to provide a clearer and more detailed overview of the processes within these larger operational contexts.

The state machine diagram allows us to verify system states when developing the system, making sure that it follows the correct transition and arrives at expected state.

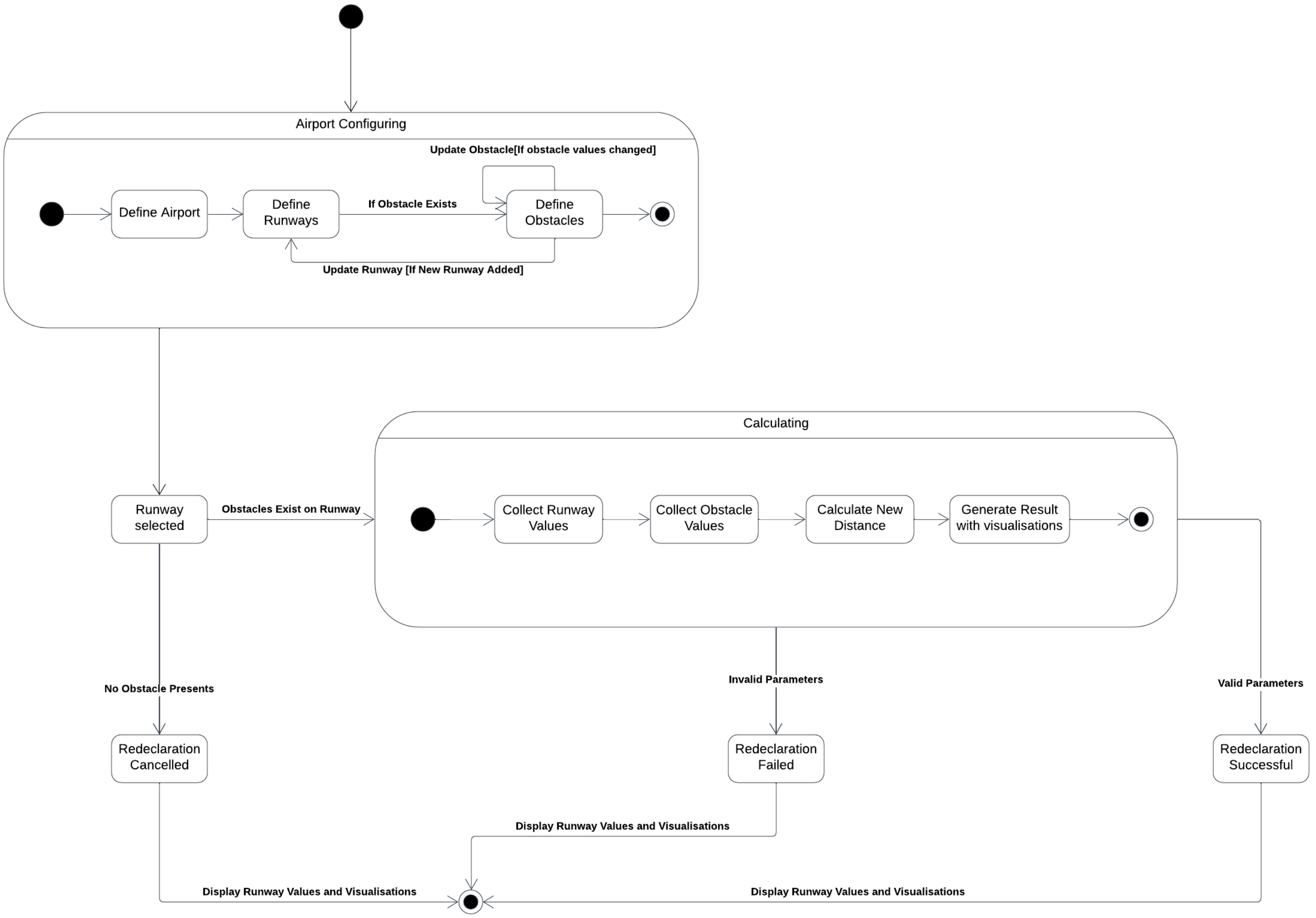


Figure 3: State Machine Diagram

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.

Scenarios help in capturing concrete details about **user requirements and expectations** from the system. They are particularly useful for identifying functional requirements and understanding the **user's workflow**.

For our project, we created **one** scenario for **each** of the personas to ensure a thorough and inclusive understanding of the diverse needs and contexts in which the system may be used.

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.

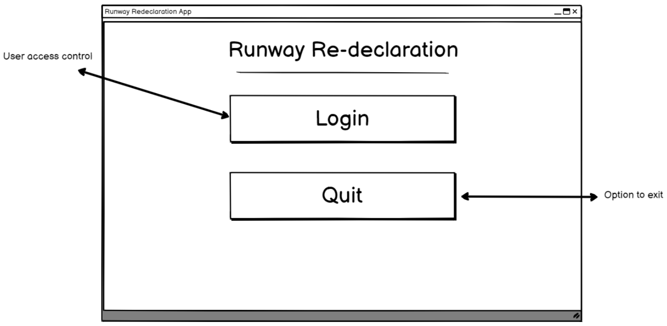
Scenario 5: Andrew (Airfield Safety Officer)

- 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.

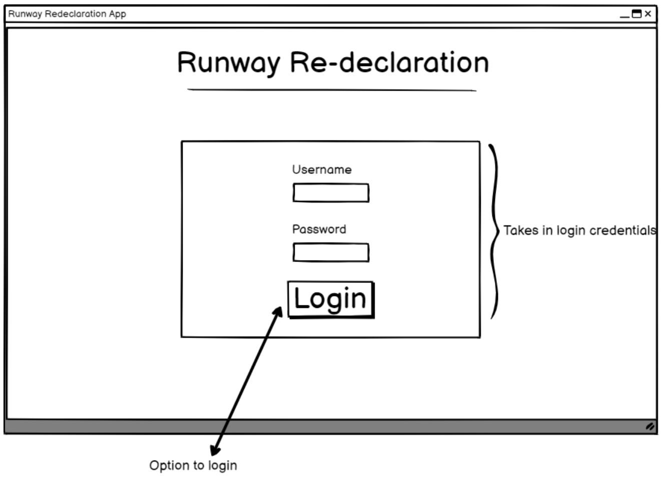
1.3 Storyboards

The storyboards describe the requirements of the product in a visual way, including the operation of the software from the user’s point of view as well as the basic function of the software. They are a plan and a record of the design decisions for the product. The diagrams show the layout of your app with display details (e.g. buttons, text boxes, labels).

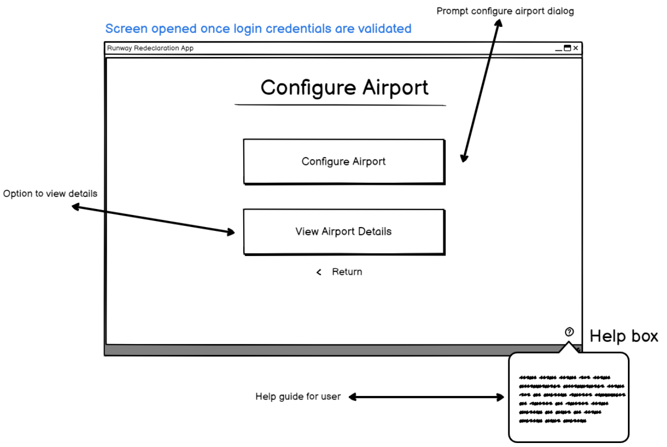
Opening Screen

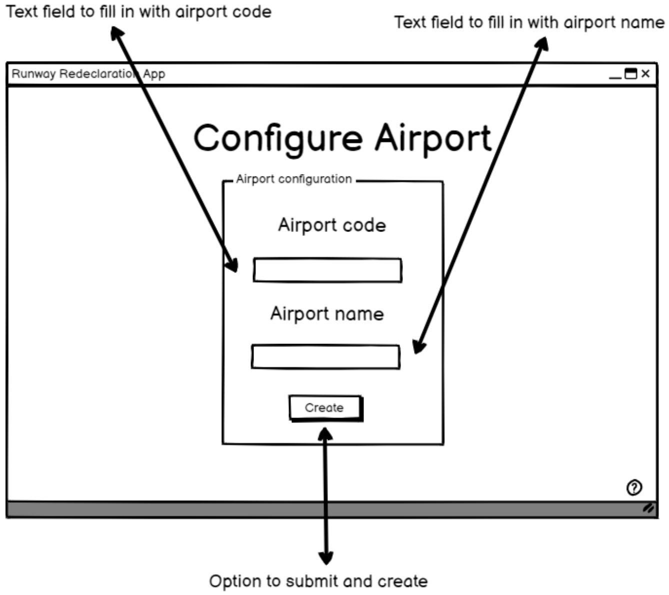


Clicking the login button opens up a window that accepts login credentials

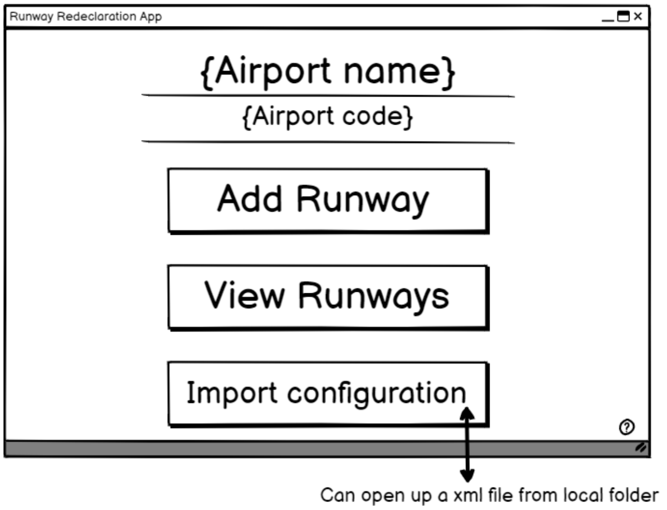


Screen opened once login credentials are validated

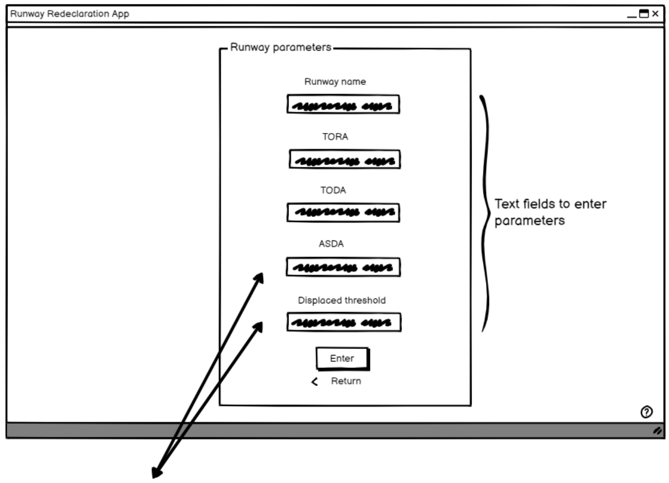



Text field to fill in with airport code
Text field to fill in with airport name
Runway Redeclaration App
Configure Airport
Airport configuration
Airport code
Airport name
Create
Option to submit and create

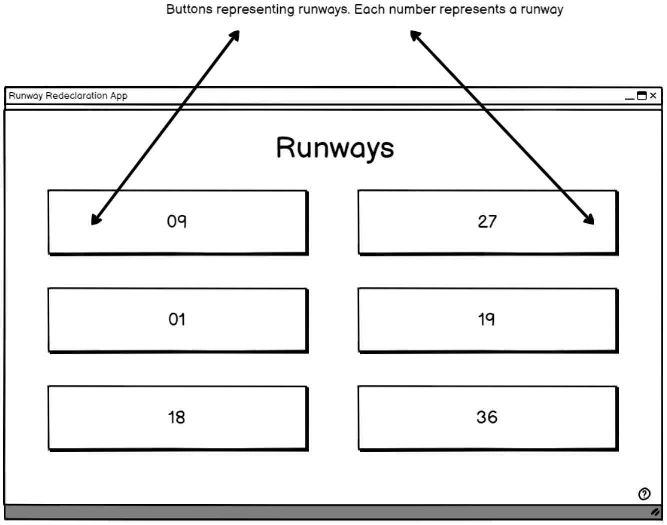
Airport menu that allows for adding, viewing and importing of XML files to load airport



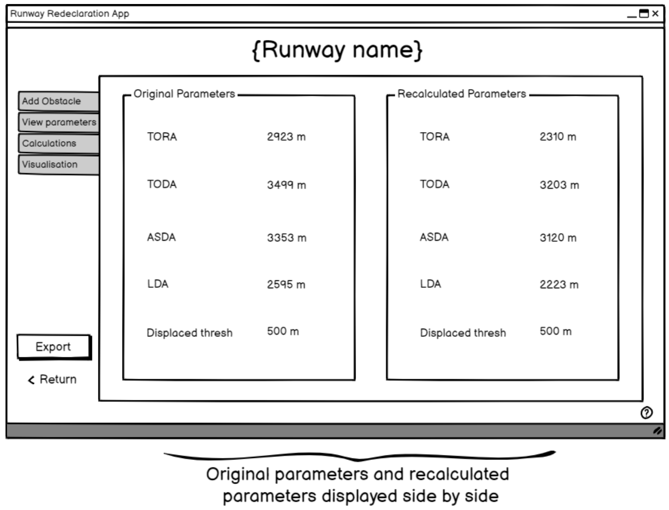
Window that opens when Add Runway button is pressed.



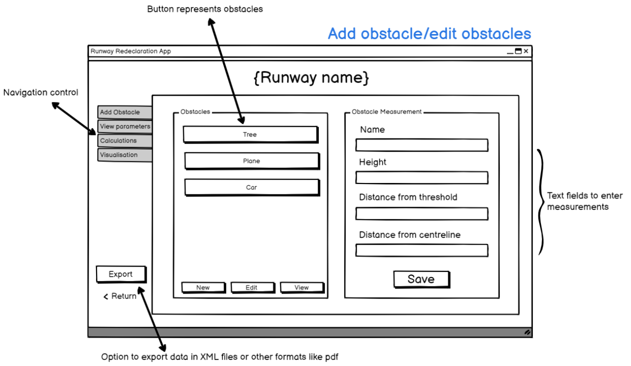
Takes runway parameters from user through text fields

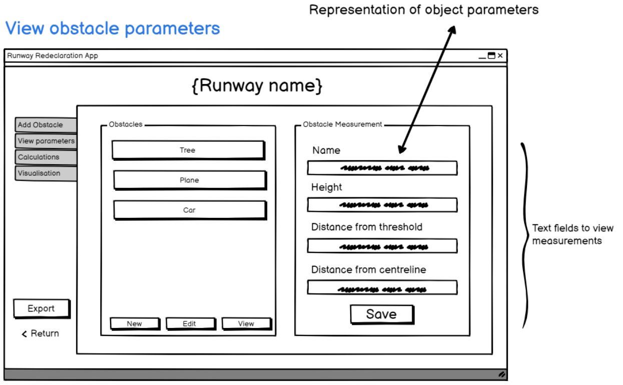

Buttons representing runways. Each number represents a runway
Runway Redeclaration App
Runways
09
27
01
19
18
36

Window displaying current runways configured for a particular airport

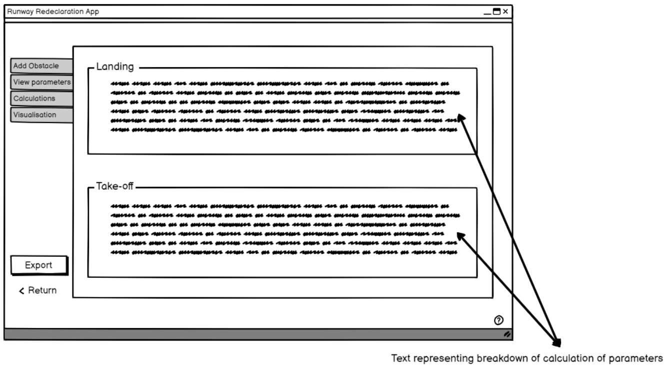


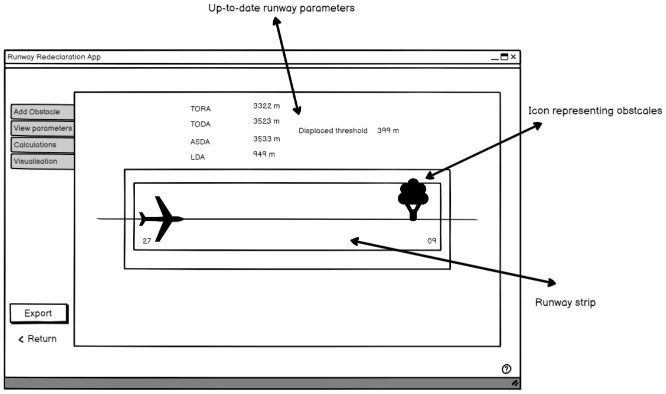
Original Parameters		Recalculated Parameters	
TORA	2923 m	TORA	2310 m
TODA	3499 m	TODA	3203 m
ASDA	3353 m	ASDA	3120 m
LDA	2595 m	LDA	2223 m
Displaced thresh	500 m	Displaced thresh	500 m


Add obstacle/edit obstacles


View obstacle parameters

Step by step calculations of landing and take-off parameters





Visualisation of the runway with recalculated parameters

2 Testing

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

We performed a collection of test cases each corresponds to a sub task within our user stories that we have planned to complete within this increment. We have carefully went through every test cases and evaluated the result.

Overview of the testing phase: ALL TEST PASSED.

2.1 Test Case 1: Existing Runway config.

Verify that the system reads the configuration of existing runways correctly.

Test steps:

1. Access the feature to read runway configurations.
2. Enter an accurate existing runway configuration to read.

Expected: The system should correctly display the configuration details of the selected runway.
Actual: The system displayed the configuration correctly.
Result: PASSED.

2.2 Test Case 2: Runway Config. UI

Verify that the UI displays the current runway configurations accurately.

Test steps:

1. Navigate to the runway configuration page.
2. Check the displayed runway configuration values.

Expected: The UI should display all current runway configurations without errors.
Actual: The configurations displayed on the UI are all correct.
Result: PASSED.

2.3 Test Case 3: Modify Runway

Ensure the modification functionality updates runway's parameters as expected.

Test steps:

1. Select the option to modify a runway.
2. Make changes to the parameters.
3. Save the changes.

Expected: The system updates the runway parameters according to the changes made and display updated values.
Actual: The runway parameters updated according to the changes made.
Result: PASSED.

2.4 Test Case 4: Save Changes

Verify that changes made to the airport configuration can be saved.

Test steps:

1. Make a change to the airport configuration (e.g. add a new runway).
2. Save the configuration.
3. Reopen the system to view saved changes.

Expected: The system should save the specific airport configuration changes and show the updated configuration upon reopening.
Actual: The system saves the changes made and displays updated configuration when reopened the application.
Result: PASSED.

2.5 Test Case 5: Add New Runway

Verify that the system allows the addition of new runway configurations.

Test steps:

1. Navigate to runway list page.
2. Select 'Add' new runway.
3. Input the required details for a new runway configuration.
4. Submit the new runway configuration.

Expected: The system should accept and save the new runway configuration.
Actual: The system saved the added runway when appropriate input provided.
Result: PASSED

2.6 Test Case 6: Define Obstacle

Ensure the system can correctly captures and stores obstacle dimensions and locations.

Test steps:

1. Navigate to obstacle list page.
2. Select the feature to input obstacle data.
3. Enter obstacle dimensions and location.
4. Submit the data.

Expected: The system saves the obstacle information correctly and display it in the UI.
Actual: The correct obstacle information was saved and displayed on the obstacle list page.
Result: PASSED

2.7 Test Case 7: Compare calculation

Verify the comparison tool displays recalculated and original runway parameters correctly.

Test steps:

1. Navigate to the comparison page.
2. View the recalculated runway parameters.
3. Compare them against the original parameters.

Expected: The system correctly displays the recalculated parameters with the originals.
Actual: The recalculated parameters the originals.
Result: PASSED

2.8 Test Case 8: Predefined Obstacle

Verify if user can view predefined obstacles.

Test steps:

1. Navigate to the obstacle page.
2. View the predefined obstacles.

Expected: The system correctly displays the predefined obstacle.
Actual: Predefined obstacles listed correctly on the obstacle page.
Result: PASSED

2.9 Test Case 9: Obstacle Update

Verify the system allows selection and updating of runway obstacles.

Test steps:

1. Navigate to the obstacle page.
2. Select an obstacle.
3. Add to current runway.
4. Remove from current runway.

Expected: The system successfully add and remove selected obstacle from current runway's and updates the list of obstacles.
Actual: The selected obstacle can be added and removed successfully.
Result: PASSED

2.10 Test Case 10: Auto runway update

Ensure the system automatically updates runway data upon obstacle modifications.

Test steps:

1. Add or remove an obstacle from the runway.
2. Check runway configuration.
3. Observe the system for automatic updates.

Expected: The system updates the runway data automatically in response to obstacle changes.
Actual: Runway data automatically updated when a obstacle was added/removed.
Result: PASSED

2.11 Test Case 11: Calc Breeakdown

Verify that the detailed breakdown of runway distance calculations is correctly displayed.

Test steps:

1. Add an selected obstacle to a selected runway.
2. Access the detailed breakdown view of the automatic calculation.

Expected: The system displays a detailed breakdown that matches the manual calculation steps and results.
Actual: The breakdown of the calculation provided by the system matched with manual calculation.
Result: PASSED

2.12 Test Case 12: Auto Calc

Verify the automatic calculation calculates new available runway distances correctly.

Test steps:

1. Provide the necessary input data to the system in order to perform calculation.

2. Initiate the calculation process.

Expected: The system calculates and displays the new available runway distances accurately.
Actual: The accurate result was calculated and displayed for selected runway with an obstacle.
Result: PASSED

2.13 Test Case 13: XML Import

Test the XML import feature for correct data integration.

Test steps:

1. Use the import feature to import an XML file.
2. Verify the data is integrated into the system.

Expected: XML data is correctly integrated into the system.
Actual: The XML data was integrated successfully and UI updates accordingly
Result: PASSED

2.14 Test Case 14: Validate XML

Confirm the accuracy and consistency of the data after being imported via XML.

Test steps:

1. Review the data imported into the system for consistency and accuracy.
2. Perform checks against known datasets for validation.

Expected: The data imported is consistent with the source file and accurate.
Actual: The imported data are both consistent and accurate compare to the XML file data.
Result: PASSED

2.15 Test Case 15: Error Message

The system provides appropriate error messages when invalid inputs/operation occurs.

Test steps:

1. Input invalid data
2. Perform an invalid operation.
3. Check if any appropriate error messages were prompted.

Expected: Appropriate error messages are displayed clearly and are understandable.
Actual: Appropriate error messages are displayed clearly and are understandable.
Result: PASSED

3 Planning

3.1 Sprint Plan for Deliverable 2

The following sprint plan outlines the tasks, owners, estimated hours, actual hours and priorities for Deliverable 2.

User Story	Sprint Backlog (Tasks)	Member	Est. Hours	Actual Hours
1: Existing Runway	1. Develop a feature to read the configuration of existing runways. 2. Create a UI to display current runway configurations. 3. Implement functionality to modify the existing runway layout and operations. 4. Ensure the system allows for saving changes specific to the airport configuration.	Anthony	5	7
2: New Runway	1. Build a feature to add new runway configurations to the system. 2. Design a UI that allows the addition of new runways and editing their layout. 3. Integrate a computation module to assist with runway layout calculations.	Anthony, Zagrosi	6	4
6: Auto Calc	1. Develop an automated calculation tool for new available runway distances. 2. Create a function to input obstacle dimensions and locations into the system. 3. Implement real-time updates and communication protocols for the safe operation of runways.	Bav, Yash	10	7
7: Update Calc	1. Set up a comparison tool to view recalculated runway parameters alongside original parameters. 2. Provide export functionality for the comparison data for reporting and analysis.	Manlin	2	4
8: Obstacle Update	1. Create a database of predefined obstacles with their specifications. 2. Develop a UI to select and update obstacles on the runway. 3. Implement a system to automatically update runway data when obstacles are added or removed.	Manlin, Bav	8	9
9: Compare distances	1. Develop a detailed breakdown view for runway distance calculations. 2. Ensure the system can export calculation data for verification against manual calculations.	Anthony, Zagrosi	3	4
10: XML Import	1. Implement an XML import feature to integrate external data into the system. 2. Ensure the system supports various XML schemas for different types of airport data. 3. Validate the imported data for consistency and accuracy.	Bav, Zagrosi	5	6
24: Error	1. Develop a comprehensive error logging system. 2. Create a UI for operational staff to view and understand error messages. 3. Implement a troubleshooting guide within the system to assist with quick resolution of issues.	Manlin, Yash	5	3

Table 1: Sprint Plan for Deliverable 2

3.2 Sprint Plan for Deliverable 3

The following sprint plan outlines the tasks, owners, estimated effort, task complexity and priorities for Deliverable 3.

User Story	Sprint Backlog (Tasks)	Member	Complexity	Est. Hours
3: 2D Top-down	1. Create option to display a 2d top down view of selected runway only.	Bav	M	2
4: 2D Side-View	1. Create option to display a 2d side view of selected runway only.	Bav	M	2
5: Both Views	1. Create option to view both top down and side on views at the same time.	Yash	S	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	M	2
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	M	4
14: Centreline	1. display runway centreline for Top-View only	Manlin	S	2
15: Lower Threshold	1. have a fixed place on screen to display the Lower Threshold	Yash	S	2
16: Auto Rotate	1. create option to automatically rotate runway strip to match its compass heading	Anthony	M	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	M	1

Table 2: Sprint Plan for Deliverable 3

3.3 Completed Burn-Down Chart for Deliverable 2

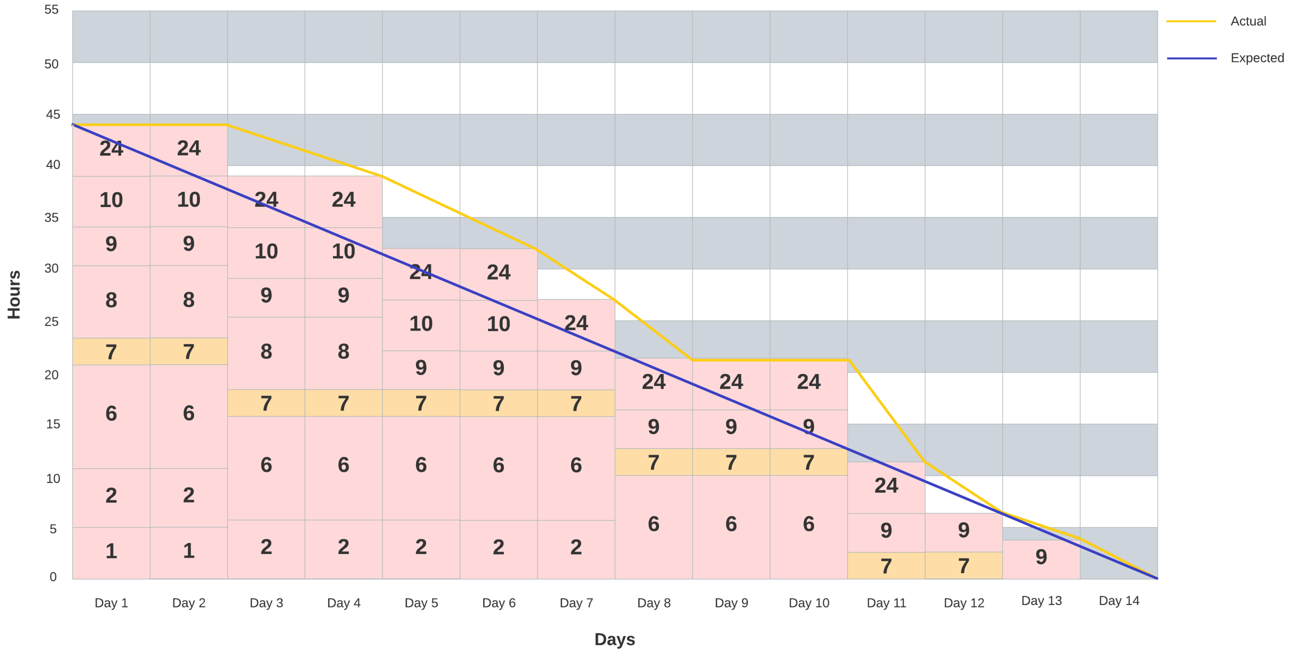


Figure 4: Completed Burn-down Chart for Increment 1 (Deliverable 2)

3.4 Day Zero Burn-down Chart Increment 2 (Deliverable 3)

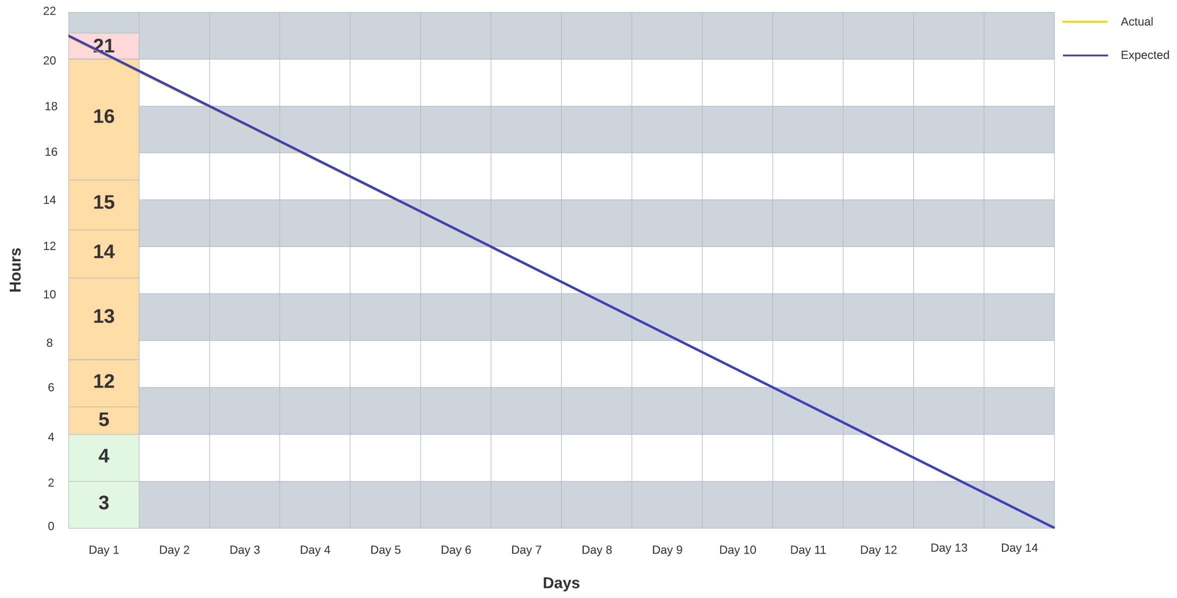


Figure 5: Burn-down Chart for Increment 2(Deliverable 3)

4 Response to feedback in Deliverable 1

No feedback for improvement received for Deliverable 1.

Bibliography

References

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

[2] Civil aviation authority website, 2024.

[3] Shi, J. Seg project definition: Runway redeclaration, Feb 2024.

[4] Shi, J. Seg project deliverable 1: Project envisioning, Feb 2024.