

COMP2211 Deliverable 2: Envisioning

Group 25:

Yuehua Yin (yy2g21)

Kai Chevannes (kc2g21)

Bagir Bazarov (bb1u20)

Georgi Iliev (gdi1u21)

Nathan Fernandes (nf4g21)

Praveen Chandrarajah (pc2g21)

Contents

1	Design	1
1.1	Persona Scenarios	1
1.2	Storyboards	3
1.3	UML Diagrams	9
1.3.1	Class diagram	9
1.3.2	Use case diagram	10
2	Acting on feedback	11
2.1	Feedback	11
2.1.1	Stakeholders	11
2.1.2	Personas	11
2.1.3	User Stories	11
2.1.4	Burn down chart	11
2.1.5	Project plan	11
3	Testing	12
3.1	Calculations Testing	12
3.2	GUI Testing	13
4	Planning	24
4.1	Sprint 1 summary: burndown chart	24
4.2	Sprint 1 summary: sprint summary	25
4.3	Sprint 2 Burndown Chart	26
4.4	Sprint 2 Sprint Plan	27

1 Design

The MVC (Model-View-Controller) software pattern is commonly used to design user interfaces. It separates the program logic into two or three interconnected elements, with the goal of separating the domain data from its visual representation. The model typically maps one-to-one with real-world objects and concepts, and contains relevant business logic. It responds to queries and operations on its state. The view represents the model graphically and is usually passive, responding to update operations. It receives user input and passes it on to the model.

In our group project, we have implemented the MVC pattern to separate concerns and improve maintainability. The Model contains an XML parser that stores data for airports, obstacles, runways, and their associated parameters. The parser also stores obstacle positions, revised logical runways, and a Verifier that ensures only valid inputs are added to these classes. Currently, the XML controller is used to save obstacles, but by the third sprint, users will be able to import and export their own XML files.

The View consists of a main dashboard, a declaration and viewing panel for a certain runway, a breakdown of calculations, and windows for adding and reviewing obstacles. These elements are all designed to provide a clear and intuitive interface for the user.

We started the design process by thinking about scenarios for each of our chosen personas, these are all primary or secondary stakeholders as outlined in our first handin. Using these scenarios as a guideline we created the UML use case diagram, UML class diagram and finally created a storyboard.

1.1 Persona Scenarios

Air traffic controller (Julia, 35):

- Julia opens the runway redeclaration tool.
- Julia imports an existing airport XML file or starts a new blank one .
- If starting a blank airport, the system will have a set of predefined obstacles, otherwise there will be obstacles added specifically to the file selected from a previous use of the tool.
- Julia selects the redeclare obstacle view.
- Julia inputs the airport details.
- Clicks the "Add runway" button
- A new runway section shows up, with space for runway details such as TORA, Clearway and stopway.
- Clicks the "Simulate runway" button
- A runway simulation opens, displaying the runway with a top down view, side on view, with options to rotate and zoom into the visualisations. Underneath the visualisations will be a list of original values and then next to it the recalculate values for the redeclared runway. Finally, underneath that will be a breakdown of calculations, allowing Julia to check her calculations to the system.
- Julia closes the simulation and clicks the "Add obstacle" button for the runway that was created earlier.
- A new drop down menu is shown which allows her to select an obstacle from the obstacles stored in the XML file being worked on.
- Julia selects the "Small Aircraft" obstacle which has a corresponding height.
- Julia inputs the remaining data about the obstacle such as distance from the start of the runway.
- Julia clicks the "Simulate runway" button and the same window from before is brought up, but this time there is a visualisation of the object on the runway with an associated calculation breakdown.
- Julia closes the simulation
- Julia clicks the "Add runway" button and adds a new runway as before.
- Julia clicks the "Export to XML" button which will export all of the runway information to an XML file, this can then be imported the next time that the system is used.
- Julia navigates back to the menu screen by using the escape key or clicking on the back button.
- Julia clicks on the "changelog" button
- All of the changes have been logged successfully, allowing her to look back on previous calculations.

Runway Maintenance Worker (Josh, 46):

- Josh opens the runway redeclaration tool.
- Josh imports an existing airport XML file or starts a new blank XML file.
- Josh selects the report obstacle view.
- Click "Report new obstacle".
- Screen to report an obstacle appears with things about the obstacle such as name, width, height, etc.
- Click "Save obstacle".
- Obstacle is saved to the system.

Airport Accident Investigator (Sarah, 55):

- Sarah opens the runway redeclaration tool.
- Sarah opens an existing XML file for an airport she is working on.
- Sarah clicks the "Redeclare" button and then "Simulate" on the runway she is investigating.
- Sarah then views the visualisation of the obstacle to understand the layout of a potential incident.
- Sarah exits out of the simulation and goes back to the menu using the escape key or clicking on the back button.
- Sarah clicks the "Changelog" button.
- Sarah can now view a history of obstacles and where they have been to help her investigation.

Pilot (Michael, 56):

- Michael opens the runway redeclaration tool.
- Michael opens an existing XML file for the airport they will be landing at.
- Michael clicks the "Redeclare" button and then "Simulate" on the runway he is going to be landing on.
- Michael can now use the values outputted to safely plan his landing.

Flight Attendant (Emily, 25):

- Emily opens the runway redeclaration tool.
- Emily opens an existing XML file for the airport they will be landing at.
- Emily clicks the "Redeclare" button and then look for the runway they will be landing on.
- Emily can now use the obstacle shown by the system combined to advise passengers about why their landing may be delayed.

Airport Operations Manager (Robert, 44):

- Robert opens the runway redeclaration tool.
- Robert opens an existing XML airport document or begins a blank one.
- Robert selects "review obstacle" button.
- Notifications of new reported obstacles by the airport investigation team will appear in the top right of the screen.
- Either click on the notification or click on the "Review Obstacles" view.
- For each obstacle, Robert can verify measurements and remove any obstacles that are not appropriate, this obstacle will now be removed from the system.
- Robert will also have the option to modify an obstacle if there has been a typo for example and a height has been set to 100 metres instead of 10 metres.
- Robert clicks the "Modify obstacle" button
- A pop-up window will come up with fields to change.
- Robert changes the values as appropriate and clicks on the "Save" button.
- The updated obstacle will now be saved to the XML file, ready for the air traffic controller to use.
- Click the "changelog" button .
- Robert can now view all previous changes to the system, with options to filter by notification types such as "Obstacle" or "Runway"
- This allows him to check that the air traffic controllers are performing their jobs appropriately.

1.2 Storyboards

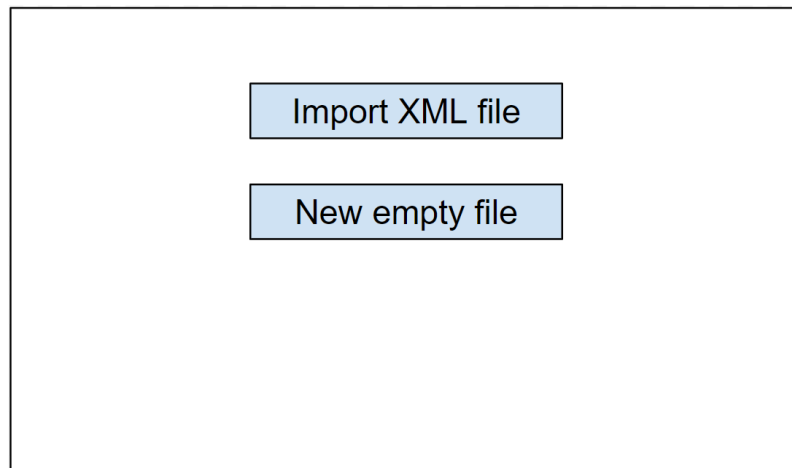


Figure 1: The screen that is shown when the user opens the application. The user can select whether they want to import an XML file into the system or start with a new blank file.

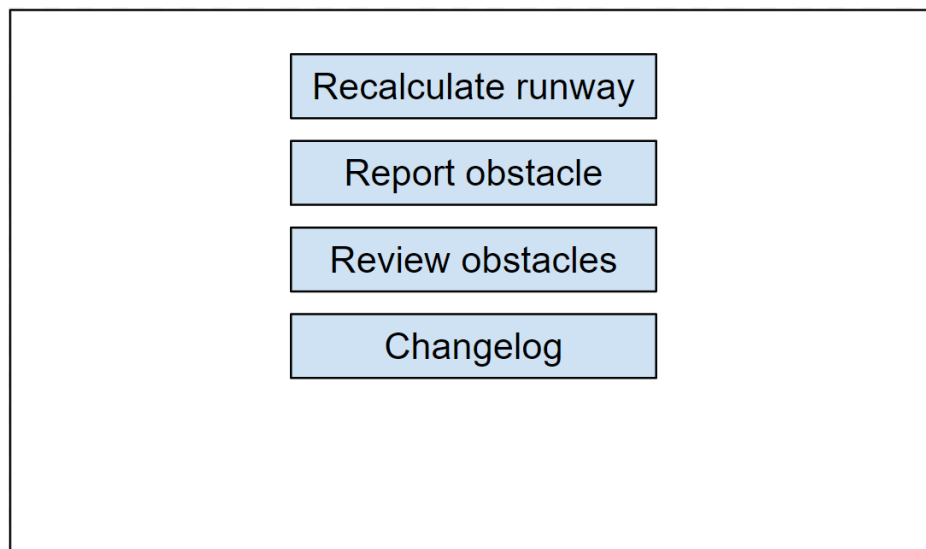


Figure 2: The main menu of the application. This is the screen the user will see after they have selected if they want to import an XML file or start new. This serves as a central hub where the user can select different screens to go to. It will be accessible from all screens using the ESC key or a button.

Airport details: RESA, blast protection etc.

Add runway

Export to XML

Figure 3a: The redeclare runway screen. Here the user will input details of the airport and runway. If a blank file/no file has been selected, the screen will start with no runways, with an option to add runways to the system. There is an option to export the current configurations to an XML file here.

Airport details: RESA, blast protection etc.

Add runway

Export to XML

Designator:

Length:

Clearway:

Stopway:

Displaced Threshold:

Add obstacle

Simulate runway

Figure 3b: The redeclare runway screen after the 'Add runway' button has been pressed. New fields are created where the user can input the dimensions of the runway. For each runway, there are buttons for adding an obstacle and simulating the runway with the provided parameters. If the loaded XML file contains information about runways, they will automatically be created on this screen with the same information as in the file displayed onto the screen.

Airport details: RESA, blast protection etc.

Add runway
Export to XML

Designator: Length: Clearway:
 Stopway: Displaced Threshold:

Select obstacle: Distance from threshold:

Remove obstacle
Simulate runway

Figure 3c: The redeclare runway screen with a runway after the 'Add obstacle' button has been pressed. New fields are created where the user can select the object from a drop-down menu which contains all the objects added to the system. The user can then input information about the object's position on the runway. The 'Add obstacle' button is changed to 'Remove obstacle' so the user can report when a runway has been cleared of obstacles.

Airport details: RESA, blast protection etc.

Add runway
Export to XML

Designator: Length: Clearway:
 Stopway: Displaced Threshold:

Select obstacle: Distance from threshold:

Remove obstacle
Simulate runway

Add runway
Export to XML

Designator: Length: Clearway:
 Stopway: Displaced Threshold:

Select obstacle: Distance from threshold:

Remove obstacle
Simulate runway

Figure 3d: When an airport has more than one runway added to the system, the runways will stack like so, so that each runway can be configured and simulated separately.

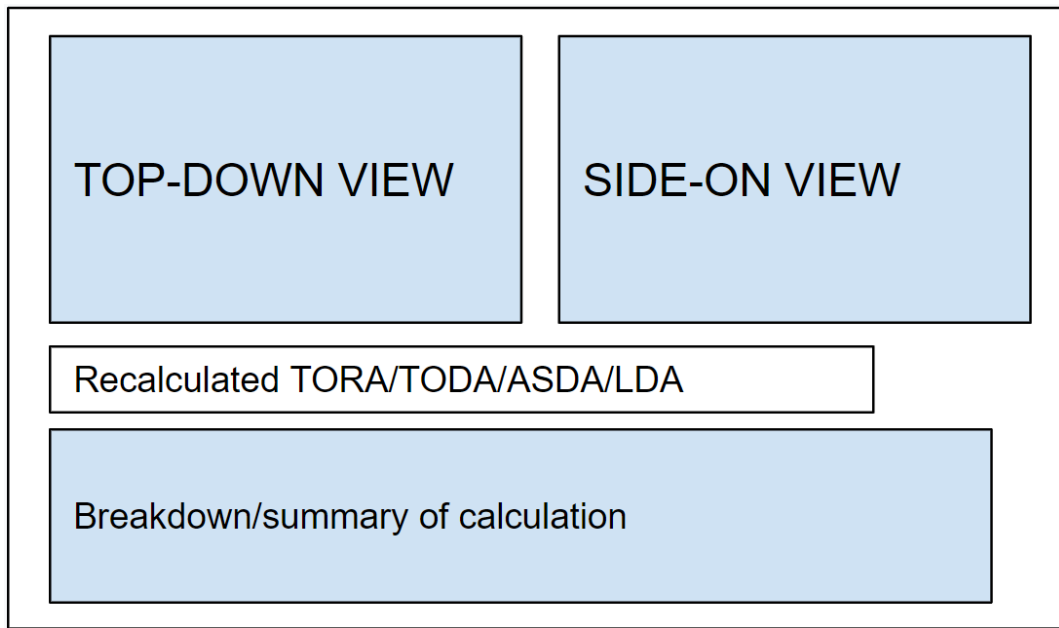


Figure 3e: The simulation scene which the user will see when they click the 'Simulate runway' button. Top-down and side-on views will be displayed onto the screen, along with the recalculated runway parameters and a breakdown/summary of calculations so that the accuracy of the calculations can be easily checked by the user.

The diagram shows the 'Report object on runway' screen. It features a light blue rectangular area with a black border. Inside this area, at the top, is a white button with a black border labeled 'Report object on runway'. Below this button is a horizontal light blue bar with a black border containing four input fields: 'Name: []', 'Height: []', 'Width: []', and 'Length: []'. At the bottom center of the light blue area is a light blue button with a black border labeled 'Add'.

Figure 4a: The report object screen. Here the user can input information about an obstacle, which will be added to the system so the user can add to a runway in the redeclare runway screen.

Confirm: Add this obstacle to system
 Name: — Height: – Width: – Length: –

No Yes

Figure 4b: When the user adds an object to the system, a popup will appear to confirm the details of the object to be added.

Name: — Height: — Width: — Length: —

Delete object Modify object

Name: — Height: — Width: — Length: —

Delete object Modify object

Figure 5a: The review obstacle screen. This will display information of all the obstacles currently saved to the system. For each obstacle, there are buttons for deleting and modifying the obstacle.

The diagram illustrates a user interface for modifying an obstacle. It features a main window with a header bar containing labels for 'Name:', 'Height:', 'Width:', and 'Length:' followed by dashed lines. A 'Name:' label is also present on the left side. A central popup window is shown, which contains four input fields for 'Name:', 'Height:', 'Width:', and 'Length:', and 'Cancel' and 'Ok' buttons at the bottom.

Figure 5b: When modifying an obstacle from the review obstacle screen, a popup window will be displayed which will allow the user to re-input the values of the obstacle.

Filter: <input type="button" value="All"/> <input type="button" value="Runway"/> <input type="button" value="Obstacle"/> <input type="button" value="Simulation"/>		
Action	User	Time

Figure 6: A record of all the actions that have occurred on the system in the current session. This will include added/deleting/modifying runways, adding/deleting/modifying obstacles, simulating a runway etc. The user can also filter based on the type of action.

1.3 UML Diagrams

1.3.1 Class diagram

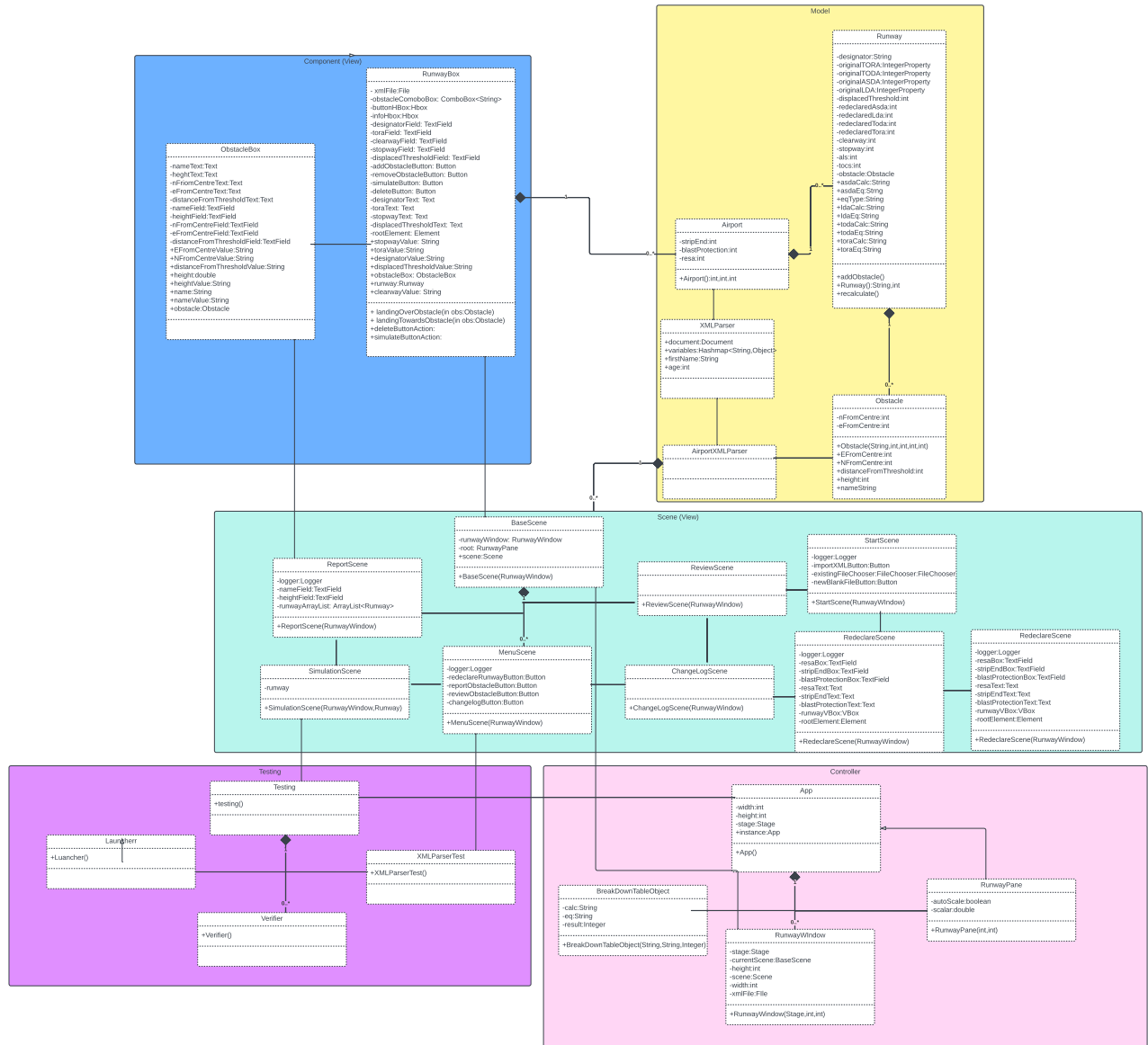


Figure 7: UML class diagram showing the classes we plan to implement and the relations between them. Figure can be zoomed in for better quality

1.3.2 Use case diagram

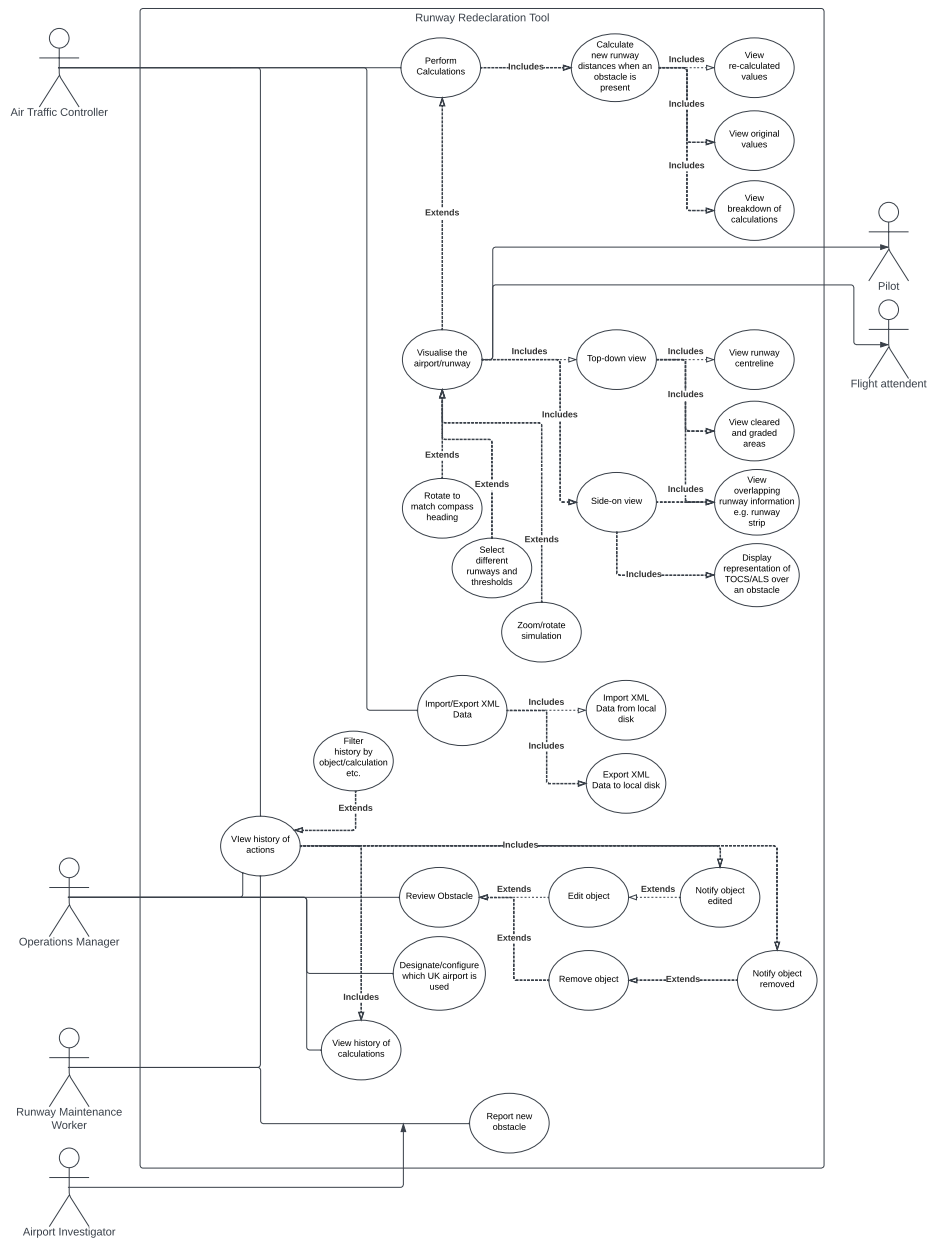


Figure 8:
UML use case diagram showing stakeholders interacting with the system. Figure can be zoomed in for better quality

2 Acting on feedback

2.1 Feedback

On Monday 27th February our group met with our supervisor and a second supervisor acting on feedback to discuss our first hand in that was submitted in the week prior. The supervisors provided the group with feedback on the submitted work during the presentation of the work. The overall feedback was that there was good structure with some of the positives being we had backup members for each task and that our risk assessment was done to a very good standard, below I go through each section in more detail and how we changed and approached the feedback we were given.

2.1.1 Stakeholders

The supervisors found that our stakeholders were of good detail however the was concerned with two of choices where we chose to the runway maintenance in the secondary stakeholder and airport investigation team in the secondary stakeholders we have no changed the order of stakeholders, by moving the runway maintenance worker from secondary to primary and the airport investigation team from primary to secondary. This was done since this order more closely resembles the tasks those job roles do in real life.

2.1.2 Personas

The positives we gathered from our personas was that we covered a good range of ages so we were able to cover all age groups. However it was noted we needed to focus more on the scope of the project and deal with relevant stakeholders so that the customers will get a product closer to what they asked for.

2.1.3 User Stories

The advice we got was with the scope of the project being not paid attention to as we needed to cover the key requirements from the specification first so we decide to the key requirement ID to the user stories. This makes it clear to the customer that all key requirements have been considered as well as which requirement is covered by which user story.

2.1.4 Burn down chart

The feedback we were given for the burn down chart is that we need a description of what the table is and to have the axis labelled we have now implemented and made these changes.

2.1.5 Project plan

One of the points that was mentioned in project planning was that we did not have a SCRUM tool nor a team leadership rota as this would leave to a mess and therefore making the project a mess to complete. We have reacted this and implemented two initiatives for our SCRUM tool with a excel sheet with tasks we are doing and by assigning team leadership roles for each week and we have now implemented a SCRUM chat on our team discord with to only metion what things we are working on.

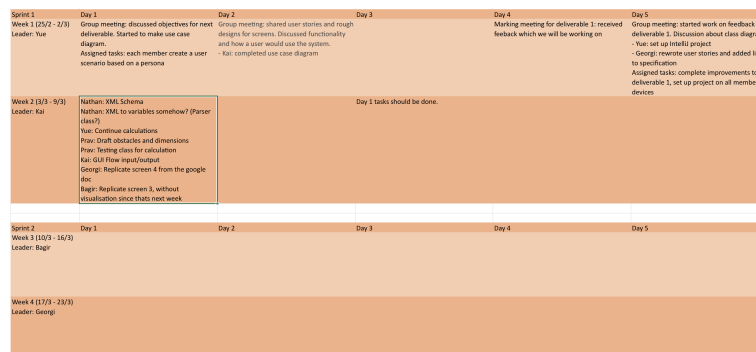


Figure 9: A small section of our SCUM plan

3 Testing

3.1 Calculations Testing

In this increment of the coding, the part of the system that was tested was to do with the calculations - and determining if we had done the calculations correct per the specification. We are using JUnit since this is a testing tool the group was familiar with and had learned how to use in a previous module; we implemented tests for the 4 unique Heathrow test scenarios provided by the spec as well as some of our own. Each redeclared runway is calculated based on the obstacle it has given its position being near the end of the runway or at the start of the runway.

```
@Test
public void testHeathrowScenario3b() {
    runway = new Runway( designator: "27L", originalTora: 3660, clearway: 0, stopway: 0, displacedThreshold: 0);
    obstacle = new Obstacle( name: "test3b", height: 15, nFromCentre: 60, eFromCentre: 0, distanceFromThreshold: 3203);

    runway.addObstacle(obstacle);
    runway.recalculate();

    assertEquals( expected: 60, runway.getDistanceFromCentreline(), delta: 0);
    assertEquals( expected: 3660, runway.originalToraProperty().get());
    assertEquals( expected: 3660, runway.originalTodaProperty().get());
    assertEquals( expected: 3660, runway.originalAsdaProperty().get());
    assertEquals( expected: 3660, runway.originalLdaProperty().get());
    assertEquals( expected: 2393, runway.getRedeclaredTora());
    assertEquals( expected: 2393, runway.getRedeclaredToda());
    assertEquals( expected: 2393, runway.getRedeclaredAsda());
    assertEquals( expected: 2983, runway.getRedeclaredLda());
}
```

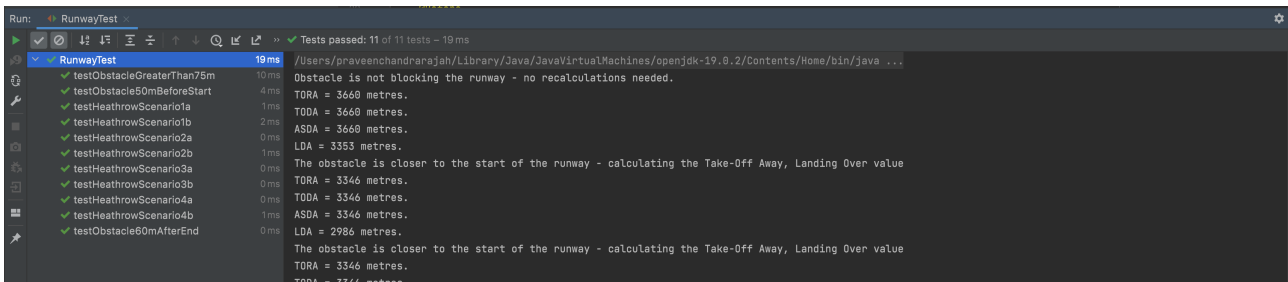
Figure 10: One of the Heathrow test case scenarios provided by the spec

We also did further testing to check if our calculation was correct by creating a scenario if an obstacle is 50m before the start of a runway or 60m after if an obstacles distance from the centre line is 75m would happen.

```
@Test
public void testObstacle50mBeforeStart() {
    runway = new Runway( designator: "09L", originalTora: 3902, clearway: 0, stopway: 0, displacedThreshold: 306);
    obstacle = new Obstacle( name: "test50mBeforeStart", height: 12, nFromCentre: 0, eFromCentre: 0, distanceFromThreshold: -50);
    runway.addObstacle(obstacle);
    runway.recalculate();

    assertEquals( expected: 0, runway.getDistanceFromCentreline());
    assertEquals( expected: 3902, runway.originalToraProperty().get());
    assertEquals( expected: 3902, runway.originalTodaProperty().get());
    assertEquals( expected: 3902, runway.originalAsdaProperty().get());
    assertEquals( expected: 3596, runway.originalLdaProperty().get());
    assertEquals( expected: 3346, runway.getRedeclaredTora());
    assertEquals( expected: 3346, runway.getRedeclaredToda());
    assertEquals( expected: 3346, runway.getRedeclaredAsda());
    assertEquals( expected: 2986, runway.getRedeclaredLda());
}
```

Figure 11: One of the unique test cases we created to check our calculations

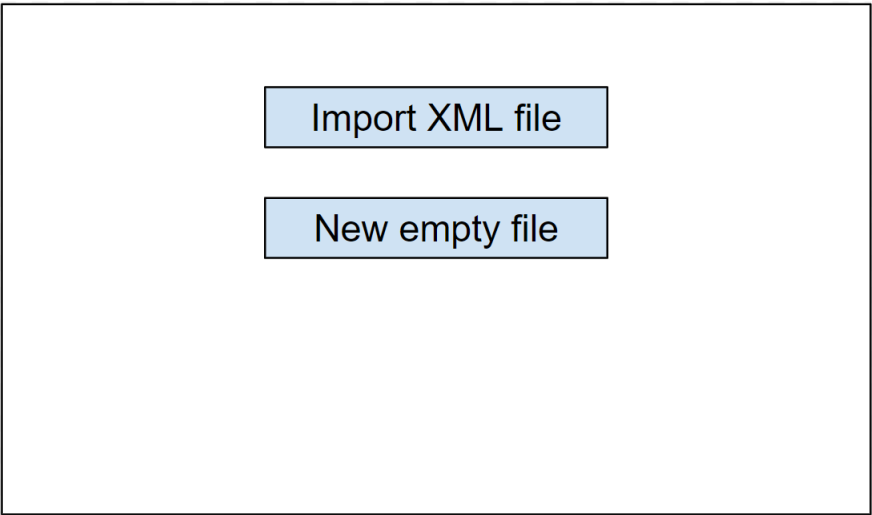


Test	Duration	Output
testObstacleGreaterThen75m	10ms	Obstacle is not blocking the runway - no recalculations needed.
testObstacle50mBeforeStart	4ms	TORa = 3660 metres.
testHeathrowScenario1a	1ms	TODa = 3660 metres.
testHeathrowScenario1b	2ms	ASDa = 3660 metres.
testHeathrowScenario2a	0ms	LDA = 3353 metres.
testHeathrowScenario2b	1ms	The obstacle is closer to the start of the runway - calculating the Take-Off Away, Landing Over value
testHeathrowScenario3a	0ms	TORa = 3346 metres.
testHeathrowScenario3b	0ms	TODa = 3346 metres.
testHeathrowScenario4a	0ms	ASDa = 3346 metres.
testHeathrowScenario4b	1ms	LDA = 2986 metres.
testObstacle60mAfterEnd	0ms	The obstacle is closer to the start of the runway - calculating the Take-Off Away, Landing Over value
		TORa = 3346 metres.
		TODa = 3346 metres.

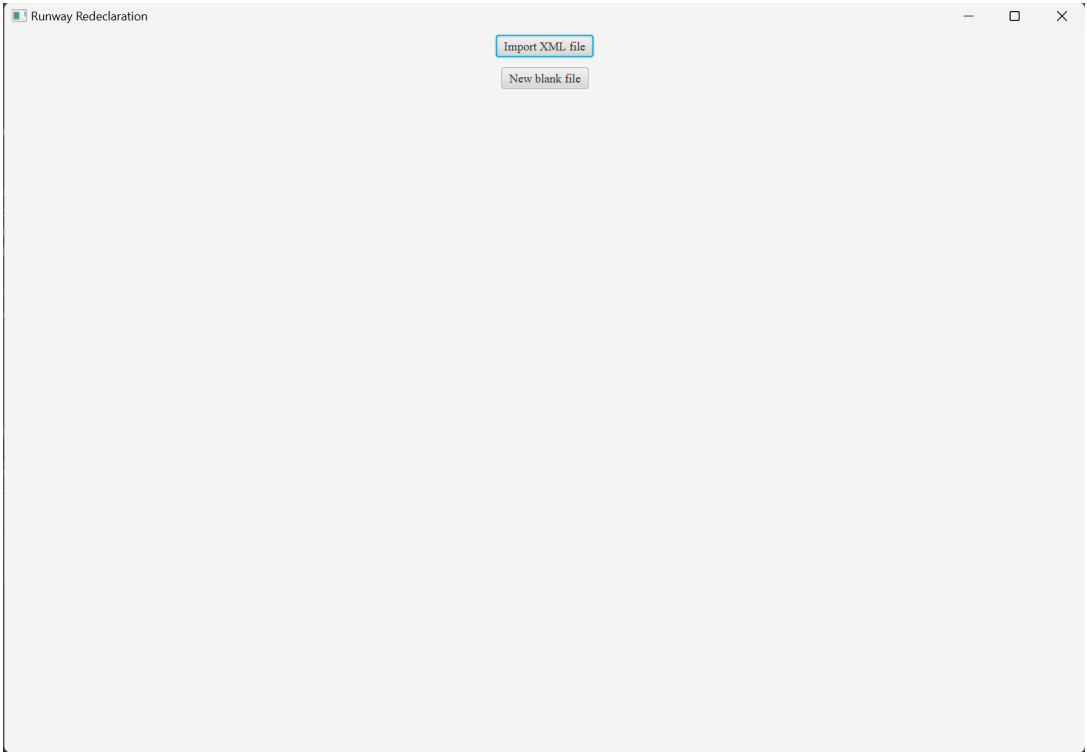
Figure 12: The calculation function passing all of the tests

In the next increment, we have acknowledged that further testing must have been performed on the system to validate its correctness, and we have plans to perform more of this unit testing in the future increments. With the next sprint as we will be focusing more on UI we will be testing the UI by using TestFX which will be used to perform extensive scenario testing on the application. So in conclusion we think with the use of both unit tests and the TestFX framework we can ensure the software operates with the intended behaviour and meets the customer specification.

3.2 GUI Testing

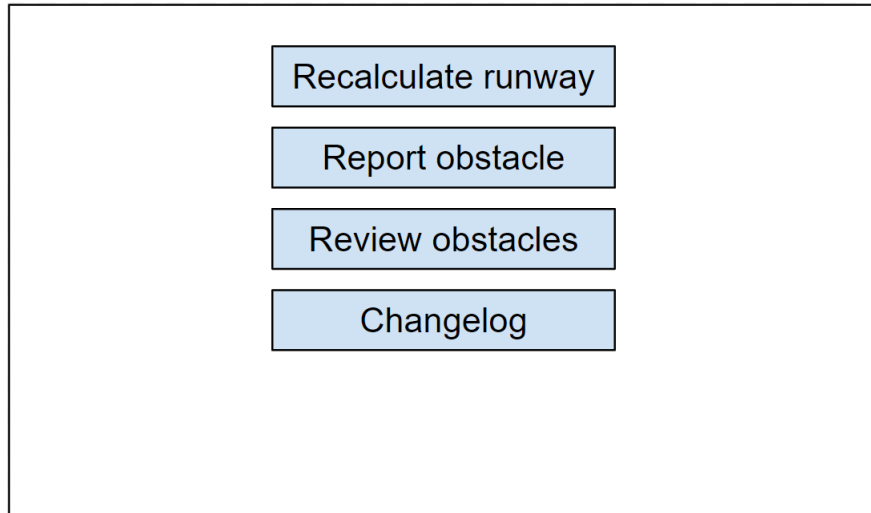


(a) Design

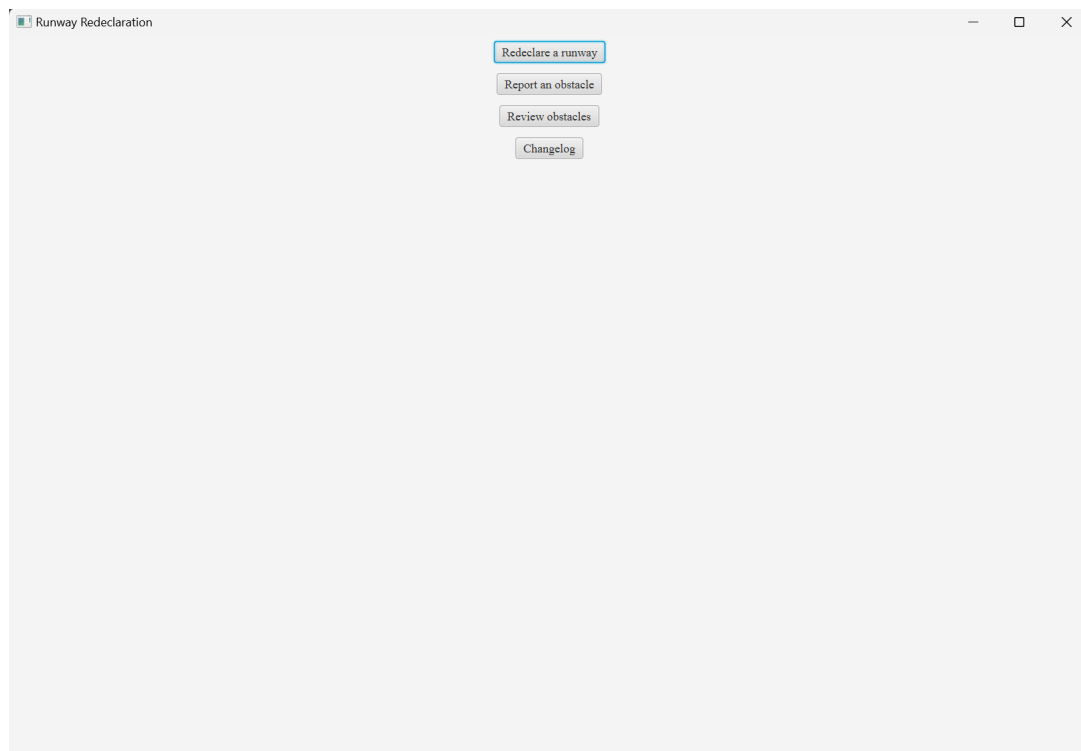


(b) Implementation

Figure 13: Start Screen

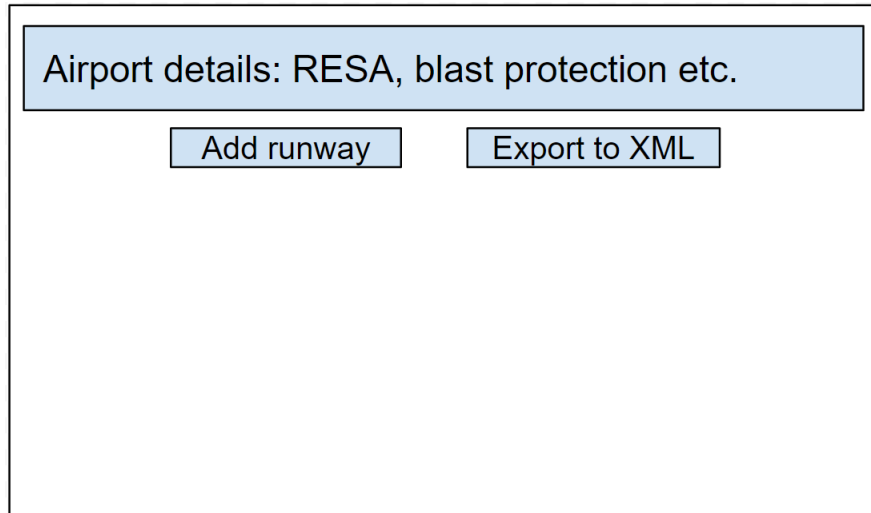


(a) Design

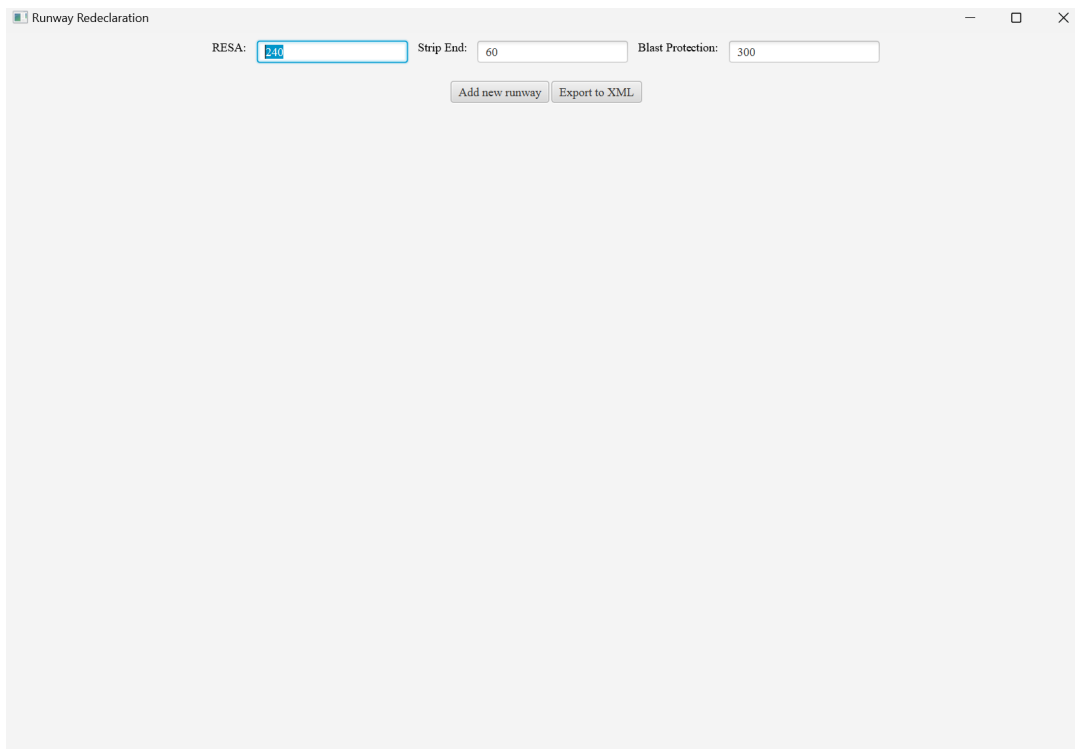


(b) Implementation

Figure 14: Menu Screen



(a) Design



(b) Implementation

Figure 15: Redeclare Screen

Airport details: RESA, blast protection etc.

Add runway
Export to XML

Designator: Length: Clearway:
 Stopway: Displaced Threshold:

Add obstacle
Simulate runway

(a) Design

Runway Redeclaration
— □ ×

RESA: Strip End: Blast Protection:

Add new runway
Export to XML

Designator:
TORA (m):
Clearway (m):
Stopway (m):
Displaced threshold (m):

Simulate runway
Add obstacle
Delete runway

(b) Implementation

Figure 16: Redeclare Screen

Airport details: RESA, blast protection etc.

Add runway

Export to XML

Designator: Length: Clearway:
 Stopway: Displaced Threshold:

Select obstacle: Distance from threshold:

Remove obstacle

Simulate runway

(a) Design

Runway Redeclaration

RESA:

Strip End:

Blast Protection:

Designator:	09L	TORA (m):	3902	Clearway (m):	90	Stopway (m):	0	Displaced threshold (m):	300
Name:	vehicle1	Height (m):	12	North from centreline (m):	-20	East from centreline (m):	0	Distance from threshold (m):	0

vehicle1 - 12m

▼

(b) Implementation

Figure 17: Redeclare Screen

Airport details: RESA, blast protection etc.

Add runway
Export to XML

Designator: Length: Clearway:
 Stopway: Displaced Threshold:

Select obstacle: Distance from threshold:

Remove obstacle
Simulate runway

Add runway
Export to XML

Designator: Length: Clearway:
 Stopway: Displaced Threshold:

Select obstacle: Distance from threshold:

Remove obstacle
Simulate runway

(a) Design

RESA:
Strip End:
Blast Protection:

Add new runway
Export to XML

Designator: TORA (m): Clearway (m): Stopway (m): Displaced threshold (m):
 Name: Height (m): North from centreline (m): East from centreline (m): Distance from threshold (m):

vehicle1 - 12m

Simulate runway
Remove obstacle
Delete runway

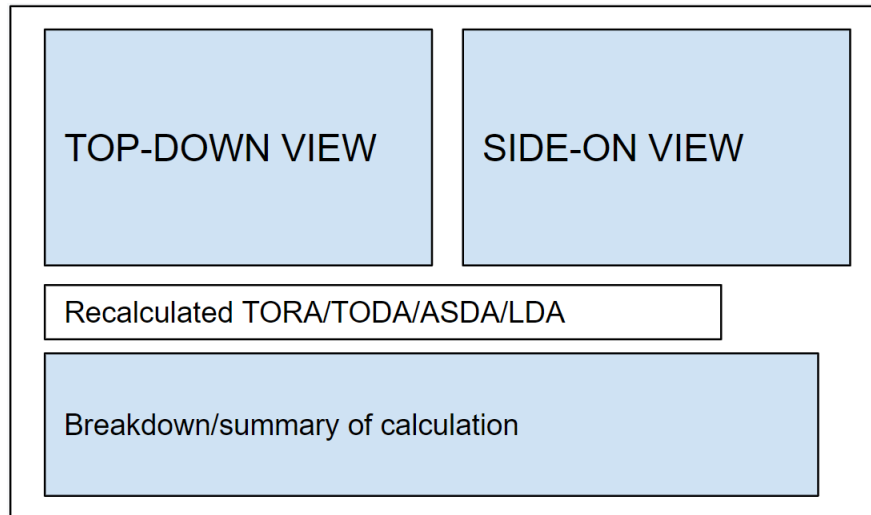
Designator: TORA (m): Clearway (m): Stopway (m): Displaced threshold (m):
 Name: Height (m): North from centreline (m): East from centreline (m): Distance from threshold (m):

aircraft1 - 15m

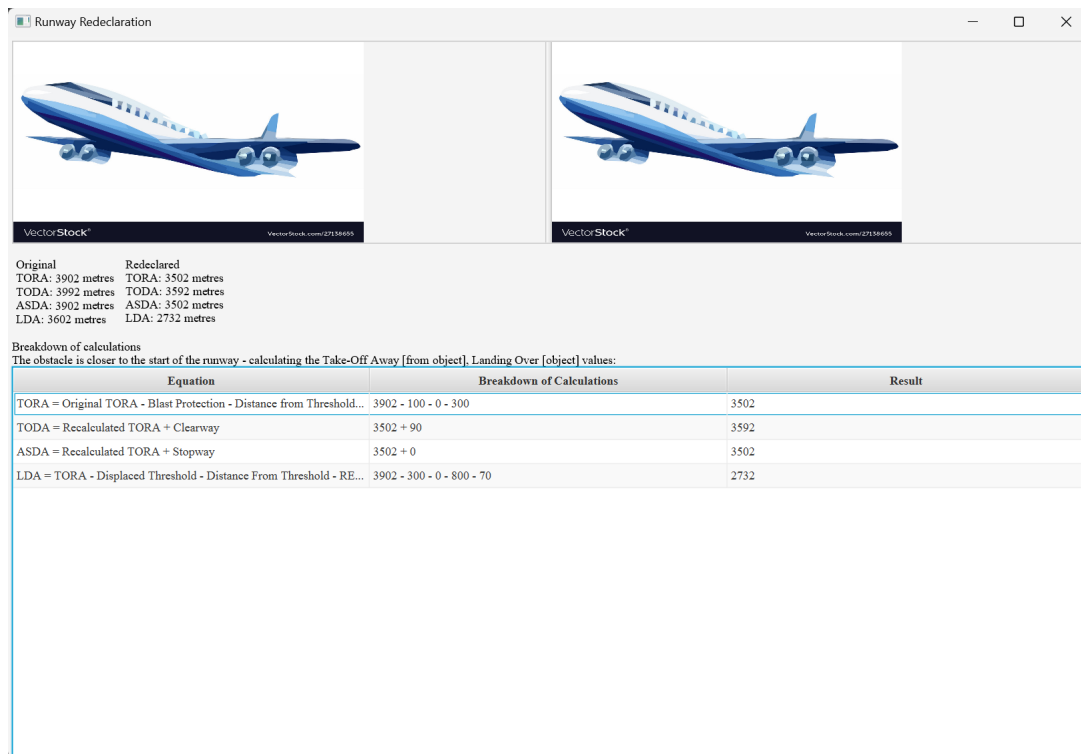
Simulate runway
Remove obstacle
Delete runway

(b) Implementation

Figure 18: Redeclare Screen



(a) Design



(b) Implementation

Figure 19: Simulate Screen

```

classDiagram
    class ReportScreen {
        +Report object on runway
        +Name: 
        +Height: 
        +Width: 
        +Length: 
        +Add
    }
  
```

(a) Design

Runway Redeclaration

Back

Name of obstacle

Information about the name of the obstacle:

width of obstacle (metres)

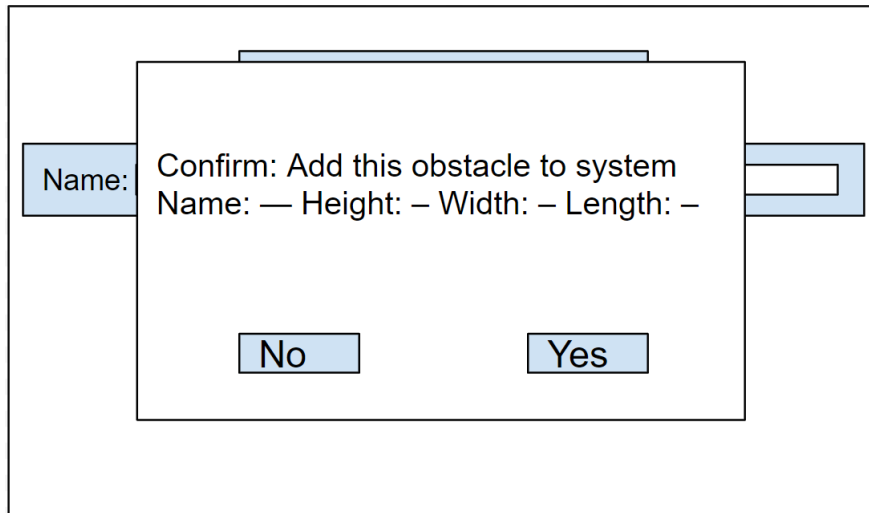
height coordinate of obstacle (met...

length coordinate of obstacle (met...

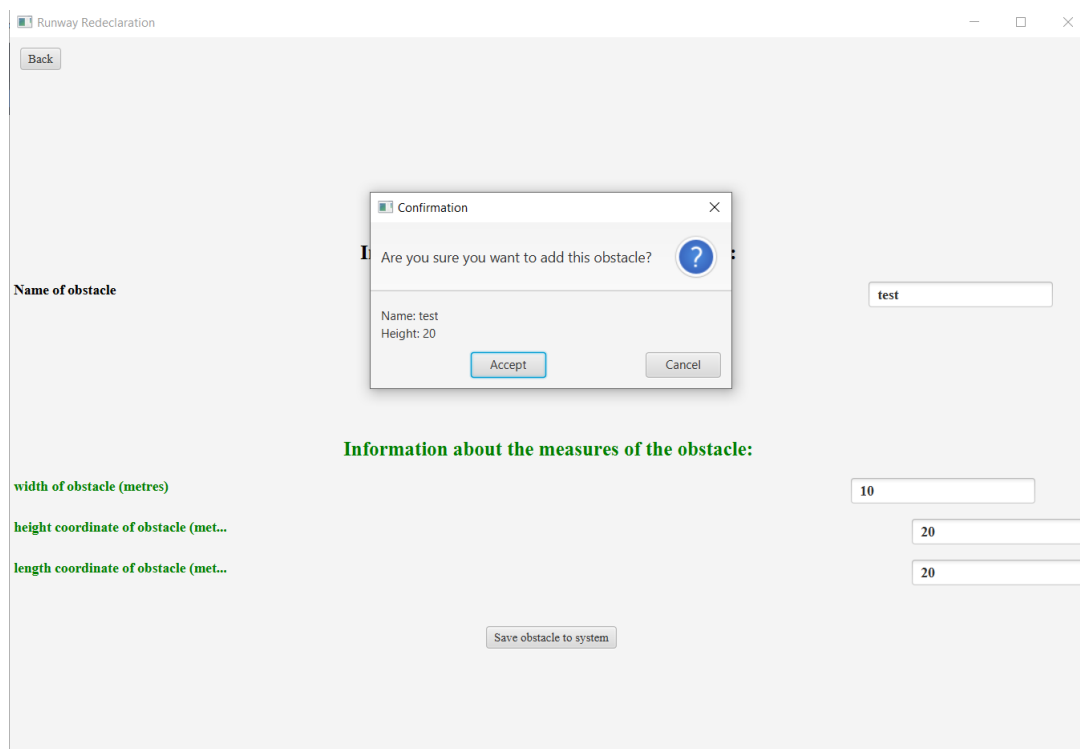
Save obstacle to system

(b) Implementation

Figure 20: Report Screen



(a) Design



(b) Implementation

Figure 21: Report Screen

Name: — Height: ——— Width: ———— Length: ———

Delete objectModify object

Name: ——— Height: ——— Width: ———— Length: ————

Delete objectModify object

(a) Design

Runway Redeclaration

Name: vehicle6Height (m): 19

Modify obstacleDelete obstacle

Name: aircraft1Height (m): 15

Modify obstacleDelete obstacle

Name: test9Height (m): 7

Modify obstacleDelete obstacle

Name: testHeight (m): 20

Modify obstacleDelete obstacle

(b) Implementation

Figure 22: Review Screen

Design diagram of a Runway Redeclaration dialog box. The dialog box has a title bar and a central content area. The content area contains a 'Name:' label and a text input field. Below this, there are four labels: 'Name:', 'Height:', 'Width:', and 'Length:', each followed by a text input field. At the bottom of the content area are 'Cancel' and 'Ok' buttons.

(a) Design

Implementation screenshot of the Runway Redeclaration dialog box. The main window shows a list of obstacles with their names and heights. The dialog box is open, showing the current values of the selected obstacle (vehicle6) and the new values for the obstacle. The new name is 'vehicle6' and the new height is '19'.

Name	Height (m)
vehicle6	19
aircraft1	15
test9	7
test	20

Current values of obstacle:
Name: vehicle6
Height: 19

New values for obstacle:
New name: vehicle6
New height (metres): 19

(b) Implementation

Figure 23: Review Screen

4 Planning

4.1 Sprint 1 summary: burndown chart

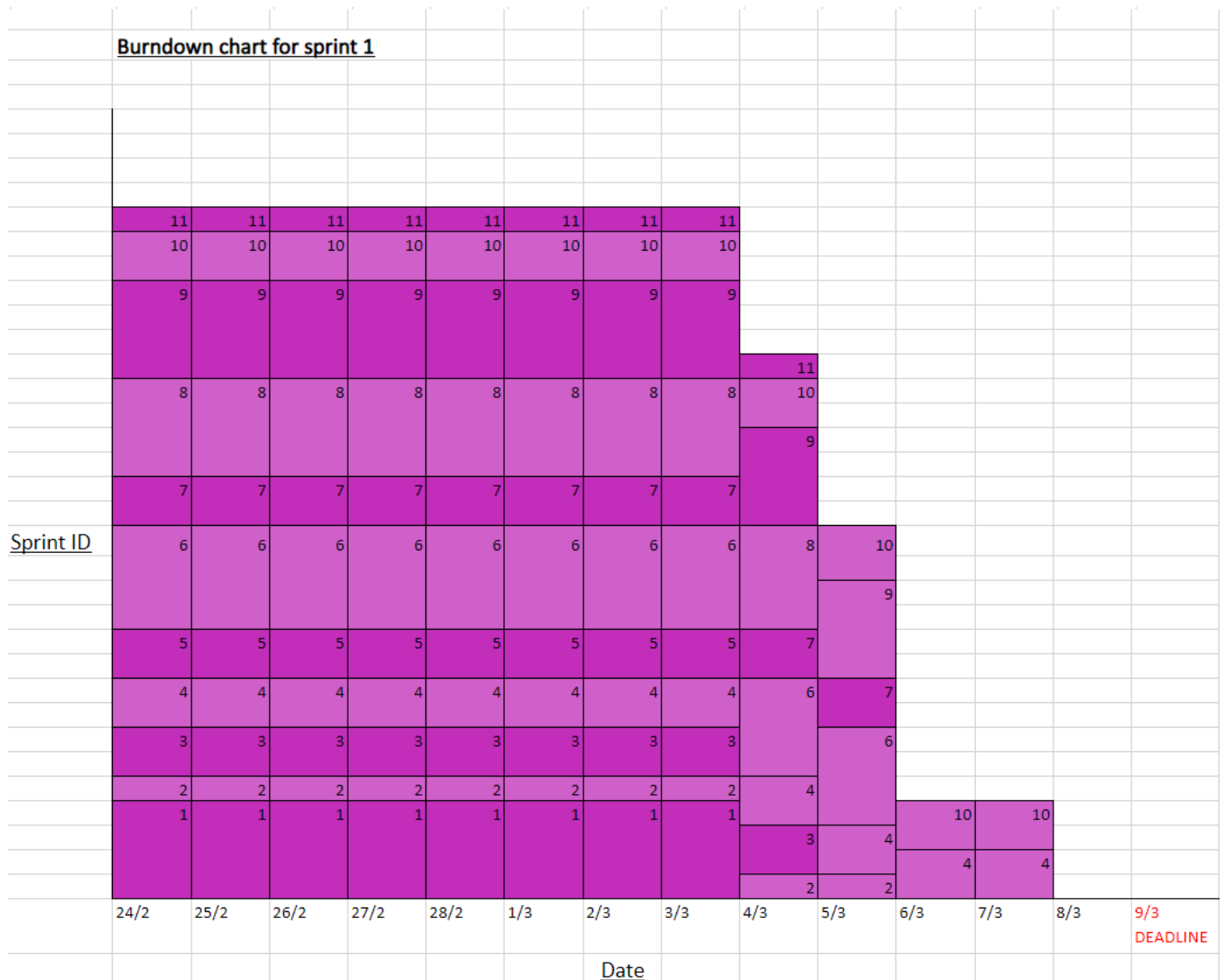


Figure 24:
Burndown chart showing the sprint ID (user story) left to implement against the date for the first sprint.

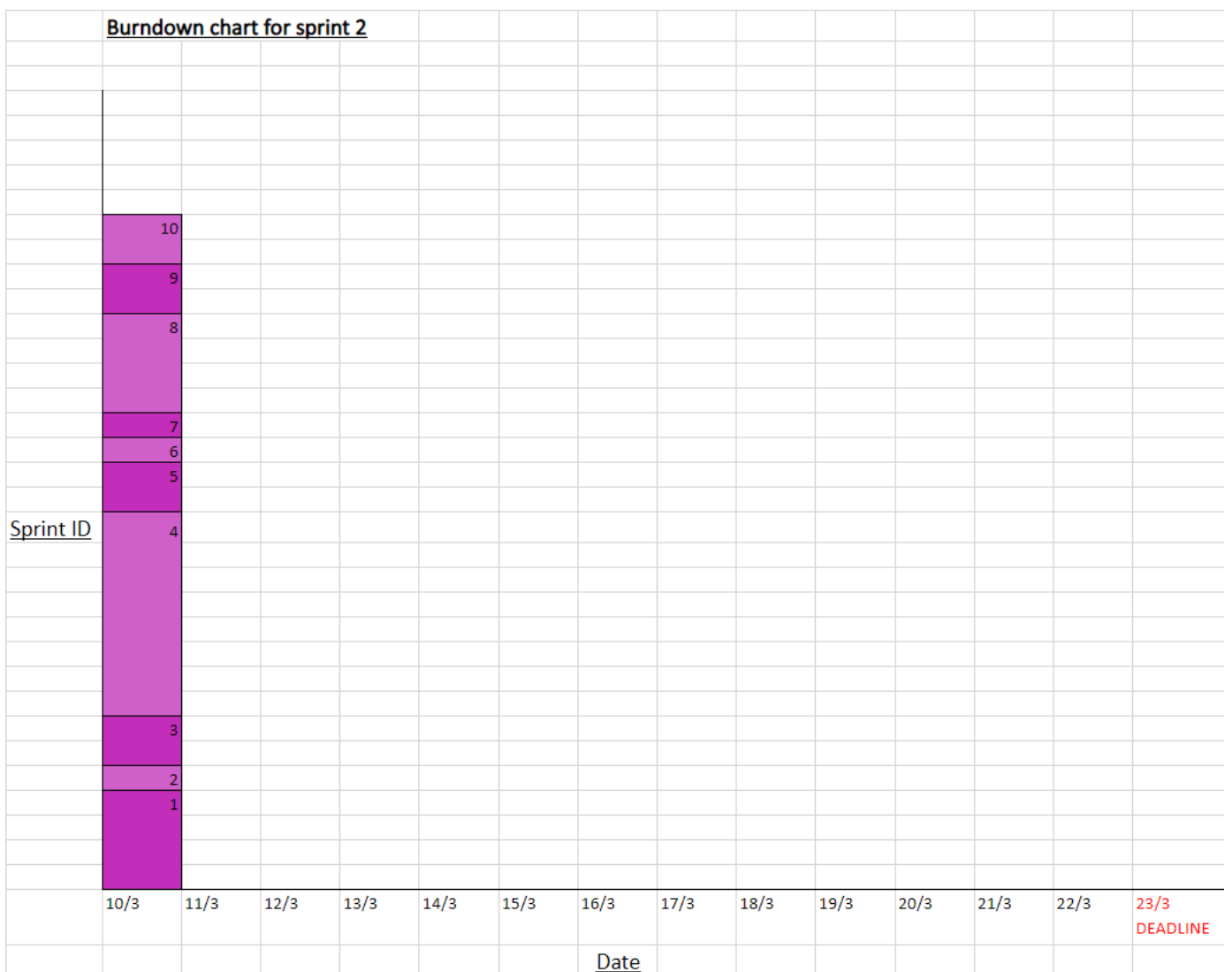
4.2 Sprint 1 summary: sprint summary

#	User story	Subtasks	Estimated Size	Allocation	Time spent
1	As an air traffic controller, I want the software to give accurate and reliable calculations so that I can feel reassured at work that my instructions to pilots are correct	<ul style="list-style-type: none"> Research the specific calculations that are necessary to meet air traffic controller needs Design and test the functionality of performing the required calculations 	L	Main:Yue Backup: Praveen Backup: Nathan	10 hrs
2	As an air traffic controller, I want to be able to see the breakdown of the calculations done so that I can check them for myself to ensure they are correct.		S	Main:Kai Backup: Georgi	5 hrs
3	As an air traffic controller, I want to be able to configure the runway's dimensions in the simulation so that the calculations done will be correct for a specific runway.		M	Main:Bagir Backup: Georgi	3 hrs
4	As an airport operations manager, I want to be able to review and correct new obstructions input into the system so that I can ensure the air traffic controllers have the most up-to-date and accurate information.	<ul style="list-style-type: none"> Create an interface for viewing obstructions and their details Develop a function that allows to correct obstructions available in the system 	M	Main:Georgi Backup: Kai	4 hrs
5	As an airport operations manager, I want the system to be configurable to permit its use at any UK commercial airport so that I can ensure the same level of safety at any airport I work at.		M	Main: Praveen Backup: Yue	2 hrs
6	As an airport operations manager, I want the system to be able to import and export data such as obstacles and airports so that I can perform high level analysis to spot trends allowing me to make the airport safer.	<ul style="list-style-type: none"> Implement import functionality Implement export functionality 	L	Main: Nathan Backup: Bagir Backup: Praveen	8 hrs
7	As a runway maintenance worker, I want to easily report airdrome obstacles or damages to the airport operations team so that they can adjust the runway parameters and inform pilots of changes made.	<ul style="list-style-type: none"> Provide maintenance workers with tools that provide instant and reliable communication for reports Develop an efficient and real-time algorithm for communications 	M	Main: Georgi Backup: Yue	6 hrs
8	As a flight attendant, I want to quickly access the updated runway parameters after an obstruction, so that I can keep the passengers informed about the flight status and avoid stress.		L	Main:Kai Backup: Georgi	2 hrs
9	As a flight attendant, I want the runway tool to generate a summary of the calculations for the updated runway parameters, so that I can quickly evaluate the feasibility of continuing the operations and make informed decisions.		L	Main:Bagir Backup: Nathan Backup: Kai	2 hrs
10	As an accident investigator, I want the program to come with a list of predefined common obstacles so that I don't have to waste time measuring them out.		M	Main: Praveen Backup: Georgi	1 hr

11	As an accident investigator, I want to directly provide measurements for new obstacles into the system so that I can provide the air traffic controller with all the information for them to do their job.		S	Main:Yue Backup: Nathan	3 hrs
----	--	--	---	-------------------------------	-------

The team has decided to move user story 18, which places the threshold with the lowest entry on the left side of the screen, from the First Sprint plan to the Second Sprint plan. As a result, the implementation of this feature will move from the second to the third increment. This change was made because most of the visualization work is defined within the Second coding iteration, and the team has not yet fully considered the aviation regulations, such as those set by the International Civil Aviation Organization (ICAO). However, it is important to note that these regulations will be considered in the next stage of the project, as they are a crucial requirement that must be implemented.

4.3 Sprint 2 Burndown Chart



Burndown chart showing the sprint ID (user story) left to implement against the date for the second sprint.

4.4 Sprint 2 Sprint Plan

#	ID	User story	Subtasks	Size	Allocation
1	12	As a pilot, I want to be able to zoom in on the system so that I can see more details of the simulation.		L	Main: Yue Backup: Kai Backup: Bagir
2	17	As a runway maintenance worker, I want to receive real-time signals concerning any obstructions or damages on the runway so that I am able to respond quickly and fix the issue which will prevent delays or accidents.	1. Display pop up in real time on the screen 2. Add the change to the changelog screen	S	Main: Georgi Backup: Nathan
3	28	As an accident investigator, I want an option to automatically rotate the runway strip to match its compass heading so that I can orientate myself more effectively.		M	Main: Bagir Backup: Praveen
4	04	As an air traffic controller, I want to be able to have top-down and side-on views of a simulation for a plane taking off/landing, so that I can understand how the plane moves in a 3 dimensional space and ensure it does not crash into anything outside of the simulation.		XL	Main: Kai Backup: Yue Backup: Bagir
5	30	As an accident investigator, I want to view the cleared and graded areas around the runway strip so that I can investigate the correct locations.		M	Main:Nathan Backup: Georgi
6	18	As a runway maintenance worker,I want to be able to have displayed the standard and the redefined values after recalculations, so that I can compare and double check them and assure the safe takeoff and landing of aircrafts.	1. Show the original values 2. Show the recalculated values	S	Main: Praveen Backup: Nathan
7	20	As a runway maintenance worker, I want the threshold with the lowest entry to always be positioned on the left side of the screen, so that it complies with the aviation regulations, such the ones set by the International Civil Aviation Organization (ICAO).		S	Main: Georgi Backup: Praveen
8	11	As a pilot, I want a visualisation of the plane landing/taking off with an obstacle on the runway on the system, so that I can feel more confident that it is safe to do in real life.		L	Main: Kai Backup: Yue Backup: Praveen
9	23	As a flight attendant, I want the runway tool to provide a visualisation of the obstacle on the runway, so that I can have a better understanding of the situation and share the information with passengers in a clear manner.		M	Main: Yue Backup: Georgi
10	09	As an airport operations manager, I want the system to display notifications to the user indicating any actions that have taken place so that I can monitor changes that employees make.	1. Implement notification feature 2. Implement filtering of notification type (e.g. filter by obstacle notification, filter by runway notification) in the changelog screen	M	Main: Praveen Backup: Kai