

COMP2211 SEG

Songyang Ding (sd8g22@soton.ac.uk)
Samuel Hyder (sh5g22@soton.ac.uk)
Andrew Shipway (as27g22@soton.ac.uk)
Muhammad Izi Arman (maia1g22@soton.ac.uk)
Samuel Wiles (sw14g22@soton.ac.uk)

August 29, 2025

1 Introduction

This report provides a concise overview of our runway re-declaration project, emphasizing the key design artifacts and team discussions throughout the development process. The completed software has been rigorously tested against the scenarios envisioned at the project's outset, and the results have been carefully evaluated. Additionally, feedback from the project supervisor and ongoing responses during development are detailed within this document, serving as constructive suggestions for further development.

2 Key Design Artifacts

This section will introduce the key design artifacts of the software, focusing on the team discussions that influenced each decision throughout the development process.

2.1 Log in and register page

When opening the application the user is shown a login page with the availability to log in with a username and password (if there is an already existing account), register a new account or continue as a guest. This page was designed with only a few features to ensure a straightforward login process for the user and minimise any confusion that could arise. The credentials of users are stored in a CSV file along with their relevant role: guest, client or admin.

There is a "show password" feature for the login page so that the user can check the password inputted, however, it's set as hidden initially to provide privacy protection for the user.

On the register page, there is a basic password strength checker included. This aims to provide the user with an understanding of what is a strong password and what isn't, encouraging the user to choose carefully.

Once the user has completed logging in, registering or decided to continue as a guest, they will continue to the main screen of our application with certain tasks only being available for the relevant roles that a user has.

2.2 General Layout

The software interface is designed to display all essential information within a single window, allowing users to view and edit data efficiently. Minimizing the number of actions required to perform functions is crucial for enhancing the user experience. Consequently, the layout is optimized to enable data editing with the fewest possible clicks.

A menu bar is located at the top of the window and has common menu options as people's habits, including input/export files and switch views. The main graphical display is centrally located, as shown in 3.2, to simulate the view of the runway. It occupies the majority of the window area, enhancing the user's ability to intuitively perceive the spatial relationships of objects on the runway. Editable values necessary for recalculations are positioned to the right of the window. This layout helps users concentrate on a smaller area during data entry, thereby reducing effort. Recalculated results, including detailed calculation breakdowns, are displayed on the left side of the window for the same reason.

2.3 Color Scheme

The software is specifically designed for airport runway management. Blue has been chosen as the primary colour for the user interface, as our team believes it not only symbolizes the sky but also creates a comforting and relaxing visual experience for users. To enhance readability, a light green colour has been selected to contrast with textual information, allowing users to read more easily.

Although not originally specified in the user requirements, our team has decided to add a feature that adjusts the colour scheme to accommodate colour-blind users. This enhancement increases the software’s accessibility, ensuring it can be used by a broader audience and enhancing its overall accountability.

2.4 Import & Export

We believe that breaking down the import/export process into smaller activities could better suit user needs. This approach would allow users to add or export smaller items individually, without the necessity of handling an entire file each time.

The software comes with a selection of predefined airports and obstacles. Anticipating frequent modifications to obstacle information, the software enables users to quickly import custom obstacles via the “custom” option in the drop-down menu. Additionally, users have the capability to import airports and obstacles through XML files. The software is designed to be adaptable to any airports or obstacles, provided that all required fields are present.

After each recalculation, users have the option to export the detailed breakdown of each distance into a text file. This file also includes all essential information pertinent to the calculation, such as the current airport, runway, and obstacle involved.

Furthermore, users have the option to generate a comprehensive PDF report that details the current state of the software.

2.5 Information Display

To enhance the intuitiveness of the central graphical display, based on the user’s choice, the software is designed to show distances and indicators for only one direction. Furthermore, the relative position of the airplane on the runway is displayed only after the user specifies the operation the plane is performing, whether landing or taking off. We believe this design approach minimizes the learning cost and maximizes usability for the users. These user-defined values do not affect the re-calculation of results, which are updated immediately whenever an obstacle is added to the runway. This will be illustrated with more details in the next section.

3 Storyboard

We have designed three main storyboards for this software, along with a login window for authentication purposes. These storyboards serve as the foundational guidelines for implementing all user requirements and provide evidence of all the design artifacts discussed in the previous section. A brief explanation will be given to all storyboards.

3.1 Log In

This storyboard outlines the design of the login window. It prominently features the software’s icon and clearly displays input fields for users to enter their credentials. If a user does not have an account, options to continue as a guest or create a new account are available.

here'."/>

Enter your Username:

Enter your Password: Show

Log In
Continue as Guest

Don't have account? Register [here](#)

Figure 1: Log In Window

3.2 Side View

This image shows the side-view storyboard design, which serves as the default scene displayed to users following the authentication process. The runway is centrally depicted on the screen, with stopways and clearways illustrated using blue blocks and dashed boxes. After each recalculation, all distances and the relative positions of the obstacle and airplane are updated based on the input values provided.

File View Help

Log Out

Results:

Runway	TORA	TODA	LDA	ASDA
27R	2860.0m	3060.0m	1850.0m	3060.0m
09L	1234.2m	1234.2m	1234.2m	1240.2m

Original Values

Runway	TORA	TODA	LDA	ASDA
27R	1231.1m	3522.0m	1232.0m	1232.4m
09L	1234.2m	1242.2m	1233.2m	1239.9m

Re-Calculated Values

Calculation Breakdown:

TORA TODA LDA ASDA

NOTIFICATION

LDA RESA TORA ASDA TODA Blast Allowance

Airport: (LHR) London Heathrow

Runway: 27R/09L

Length: 5555.0m

Displaced Threshold: 1111.0m

Clearway Distance: 100.0m

Stopway Distance: 80.0m

Airplane:

Operation:
☐ Taking Off
☐ Landing

Designator:
☐ 09L
☐ 27R

Obstacle:

Obstacle: XXXX Plane

Height:

Distance from centreline:

Distance from threshold 1:

Distance from threshold 2:

Figure 2: Side View Scene

3.3 Top View

The top view scene represents the runway as seen from above. In addition to the side view board, the top view scene contains a compass that can rotate the runway display by 45 degrees when it's clicked.

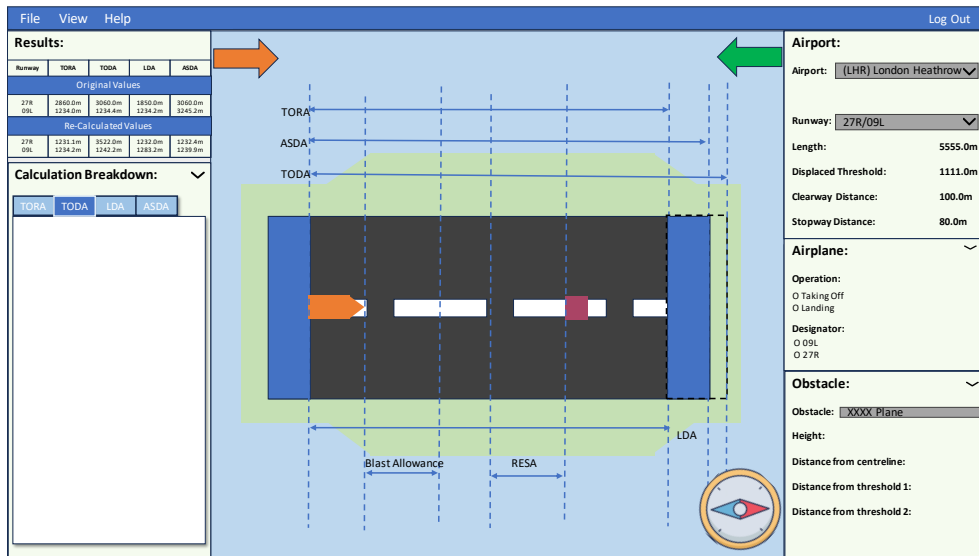


Figure 3: Top View Scene

3.4 Simultaneous View

Both the side and top views are displayed simultaneously in this scene. To simplify the interface, distances are initially hidden when entering this scene. However, the user makes the distances visible again by checking a designated checkbox.

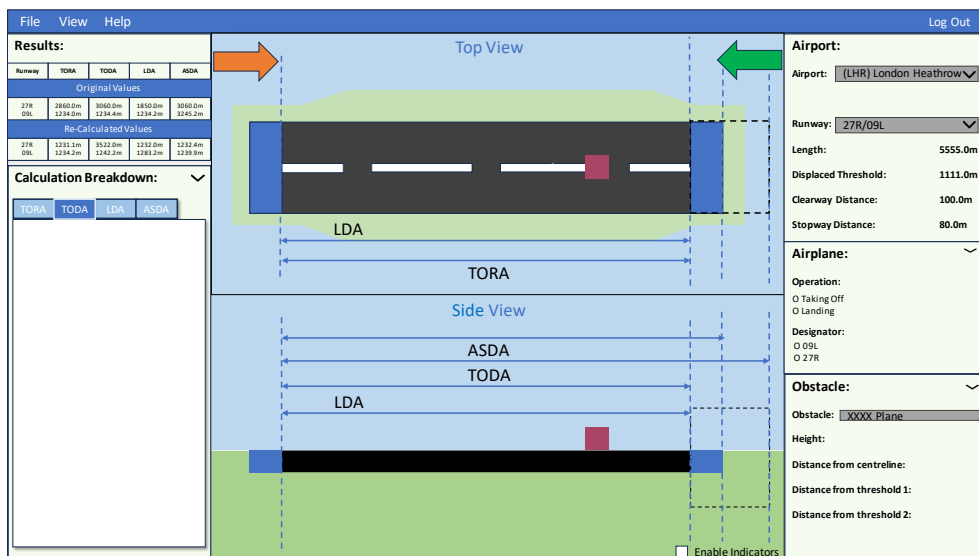


Figure 4: Simultaneous Scene

4 Response to feedback

The feedback received was, overall, positive and we built upon the areas in which we received praise. Our testing strategies and work planning were recognised as effective and so we employed similar techniques for this increment.

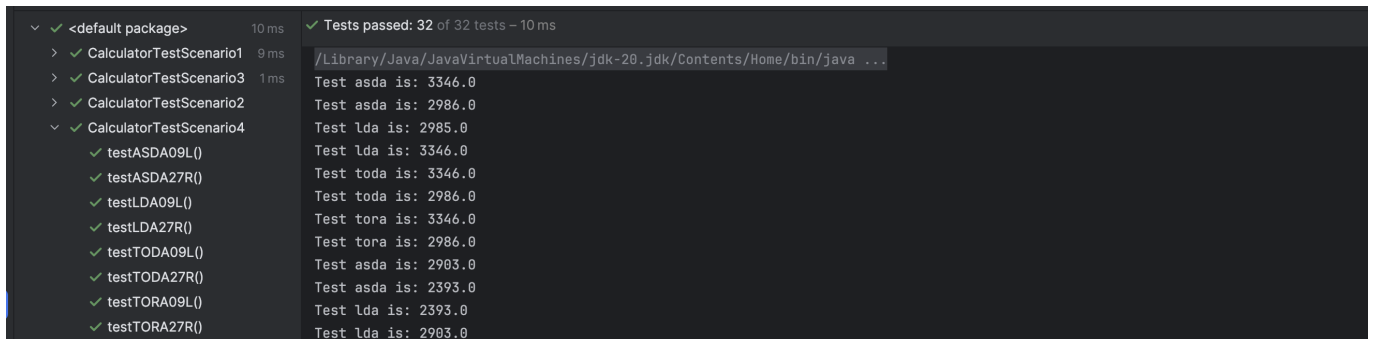
We were told that we should enhance the user controls to allow for distinguishing between users, admin etc. We had this planned for this increment and have been able to complete this, including a login and register page which works by accessing a CSV file.

5 Testing

Various testing techniques are employed to evaluate the software. This section will outline all the techniques used, with each method explained through one specific example.

5.1 Unit Testing

To verify the accuracy of the calculations, we have implemented the given examples as test cases and compared them with the results produced by the system. The system has successfully passed all tests, as demonstrated in the results presented here:



```
<default package> 10 ms
> CalculatorTestScenario1 9 ms
> CalculatorTestScenario3 1 ms
> CalculatorTestScenario2
> CalculatorTestScenario4
  testASDA09L()
  testASDA27R()
  testLDA09L()
  testLDA27R()
  testTODA09L()
  testTODA27R()
  testTORA09L()
  testTORA27R()

Tests passed: 32 of 32 tests - 10 ms

/Library/Java/JavaVirtualMachines/jdk-20-jdk/Contents/Home/bin/java ...

Test asda is: 3346.0
Test asda is: 2986.0
Test lda is: 2985.0
Test lda is: 3346.0
Test toda is: 3346.0
Test toda is: 2986.0
Test tora is: 3346.0
Test tora is: 2986.0
Test asda is: 2903.0
Test asda is: 2393.0
Test lda is: 2393.0
Test lda is: 2903.0
```

Figure 5: Unit Testing

5.2 Boundary Testing

5.2.1 Object Height

Our program accepts objects of height greater than or equal to 0.01m and height less than or equal to 100m. We made this design choice as we believe heights that aren't in this range would be unreasonable. When inputting 0.01 for the obstacle height, it is able to handle this.

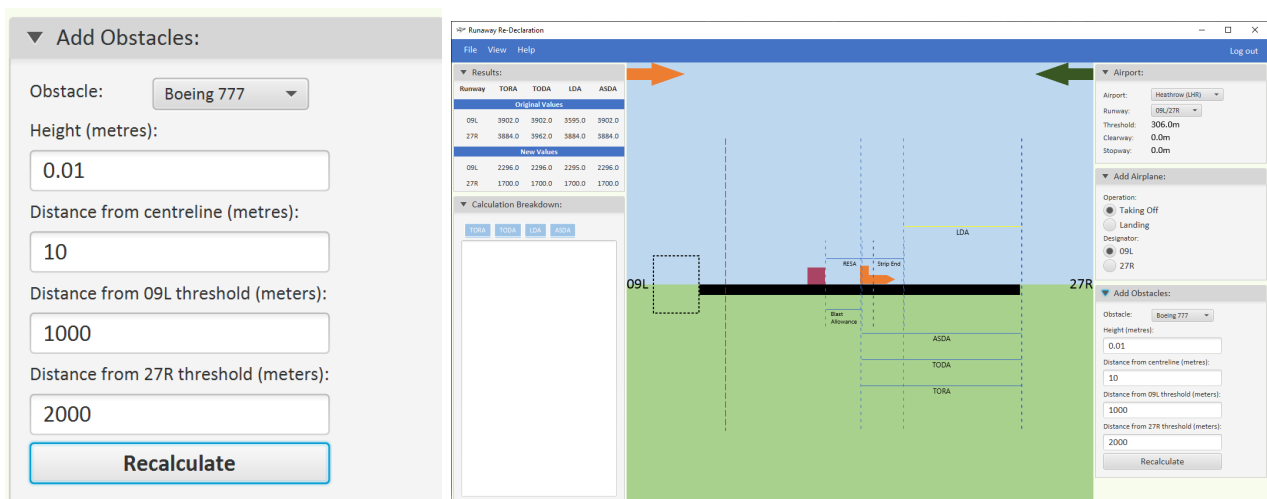


Figure 6: 0.01 meters is accepted for boundary testing of obstacle height

Anything below 0.01 results in an error, as well as anything above 100. Below is an example of inputting 101m height.

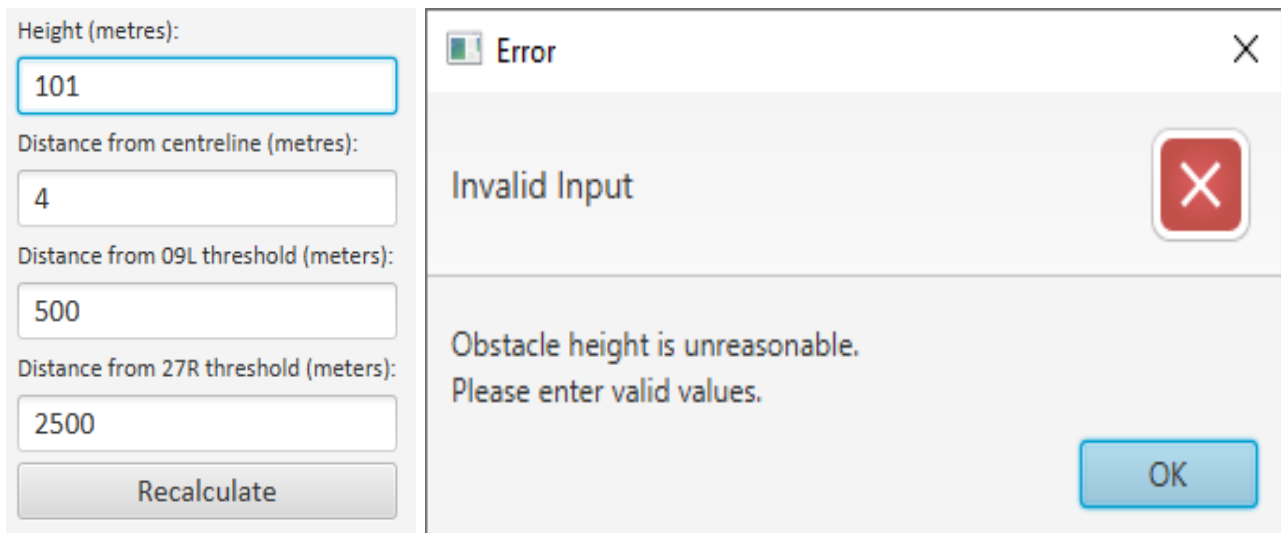


Figure 7: 101 meters is rejected for boundary testing of obstacle height

5.3 Extreme Value Testing

5.3.1 Obstacle Height

Our program can handle extremely large or small numbers for obstacle height, although there is no specified value that objects should exceed height-wise. However, we believed that our program shouldn't accept unrealistic values- such as values taller than skyscrapers. With this in mind, as shown in the boundary testing, we adapted our code such that it doesn't handle values above 100 meters or below 0.01 meters.

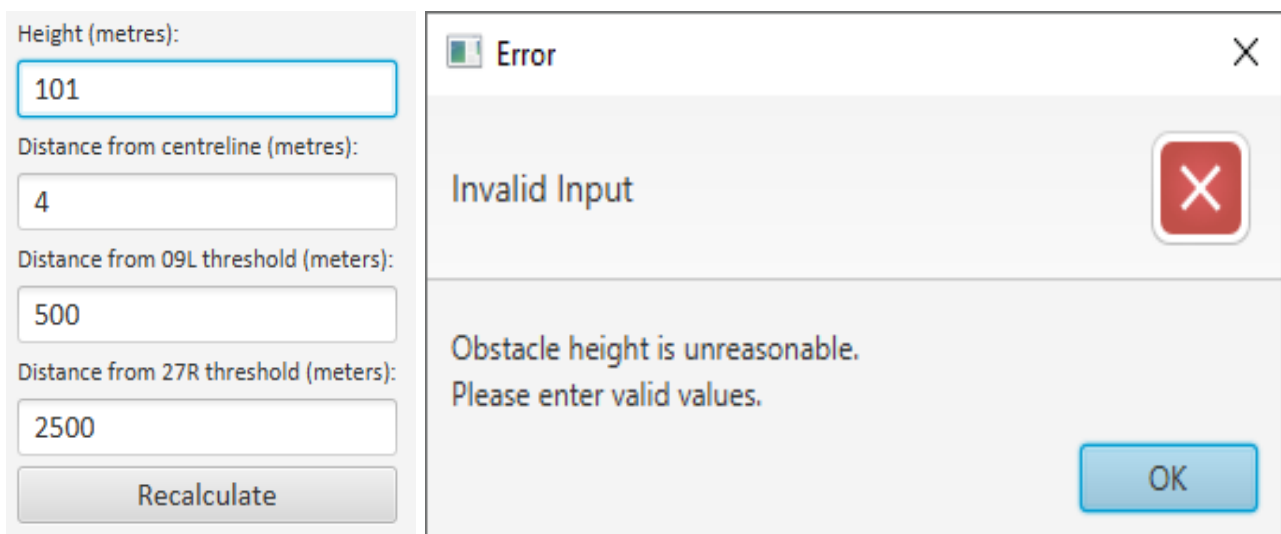


Figure 8: 1000 meters is rejected for boundary testing of obstacle height

5.4 Invalid Value Testing

All inputs should be numeric and convertible to a number. If an invalid value is inputted then the software will reject and require the user to input again.

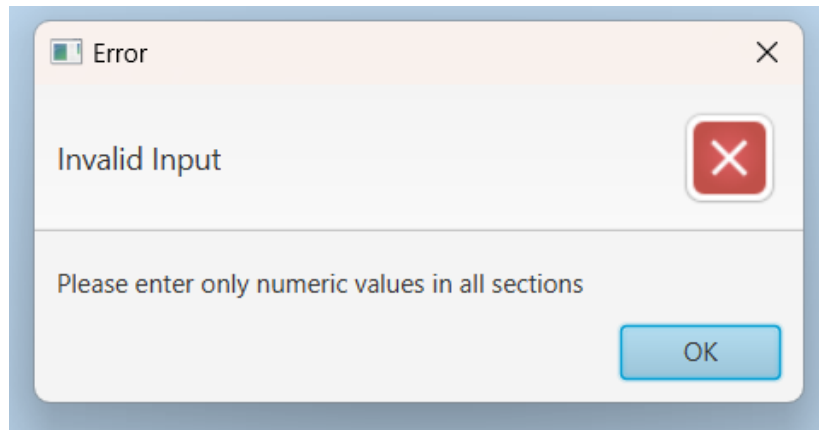


Figure 9: An invalid input type will provide an error

5.5 Zero Value Testing

5.5.1 Obstacle height

Our program's functionality includes a feature that denies obstacles with a height value of zero, a condition that lacks realism. In practical terms, a zero-height obstacle essentially translates to the absence of any obstacle. By enforcing this logic, our simulation maintains coherence and reflects real-world scenarios more accurately.

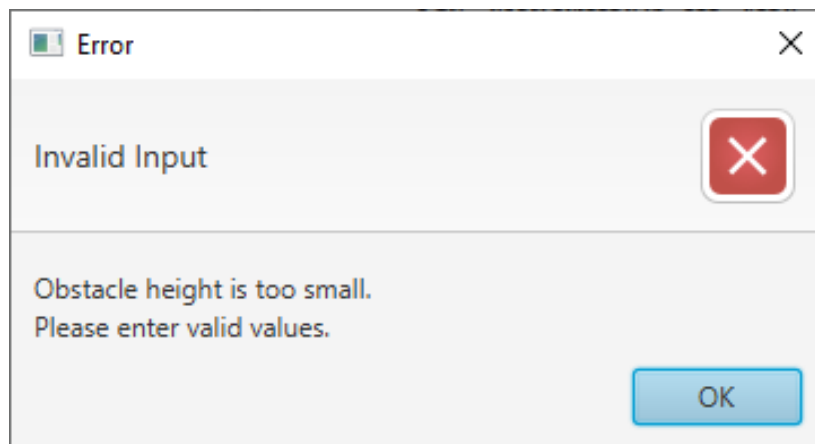


Figure 10: Result when an input of 0 is given for the height of the obstacle

5.6 Negative input

5.6.1 Obstacle height

Negative values inputted into the program should reject these inputs as a negative height for obstacle isn't possible. Our program will show an error message letting the user know what is wrong and then it will also clear the text field of the height, ready for the user to input valid inputs.

5.7 Regression Testing

Regression testing is a software testing technique that involves retesting previously tested features to ensure that changes or additions to the software haven't introduced new defects or unintended alterations to existing functionality. It aims to validate that the recent modifications haven't adversely affected the system's performance or behaviour in areas that were previously functioning correctly. Regression testing typically involves running existing test cases against the updated version of the

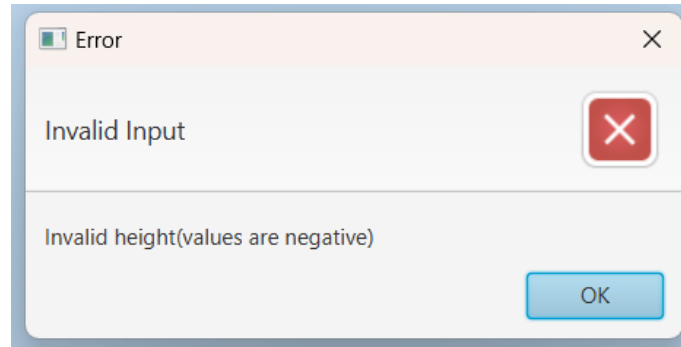


Figure 11: Negative number is inputted for the obstacle height.

software to confirm that it still behaves as expected. This process helps maintain the stability and reliability of the software product across different iterations or updates, ensuring that any regressions (unintended changes or defects) are identified and addressed promptly.

In the previous increment, we conducted 15 comprehensive tests to refine our program and align it precisely with the necessary requirements. In this current increment, all these meticulously executed tests continue to demonstrate flawless functionality, affirming that our program operates seamlessly and meets all specified criteria.

5.8 Tests against Scenarios

As detailed in previous reports, we have identified six scenarios for this project. This section will briefly revisit the criteria for each scenario, conduct tests against these criteria, and provide an evaluation of the outcomes.

5.8.1 Qualified Calculator of Air Traffic Control, Ava

This scenario describes Ava, a qualified calculator of the air traffic control, undergoing standard calculations. Ava firstly logged into the system using her credentials and assuming she is stored to be working at Gatwick airport, then the system will load the Gatwick airport information.

5.8.2 Qualified Calculator of Air Traffic Control, Ava

This scenario describes importing and exporting XML files. Ava could import local XML files to add more airports and obstacles to the software. This allows the software to be configurable to all airports and obstacles, as long as they have sufficient information in the XML files. On the other hand, Ava could export all airports and obstacles that are stored in the system to a XML file that could be reused for another time.

5.8.3 Regulatory Authorities(CAA), Lisa

This scenario describes the process of logging in with specialized credentials. Lisa, an employee of the Civil Aviation Authority (CAA), regularly needs to access airport runway information. Consequently, she has been granted an Admin account, which provides the highest level of system permissions, including the ability to view runway information directly from the interface. With this access, Lisa can now view detailed data stored for each airport, such as strip end lengths and blast protection measures from the system's database.

5.8.4 Qualified Calculator of Air Traffic Controller With A Disability, Maria

This scenario describes a user who wants to change the colour scheme on the application to accommodate for her colour blindness. Our application passes this text by allowing all users to change the

color scheme from the menu. We also adjusted our application slightly to not have the colour schemes directly for colourblindness, instead just have them describe what colour combination they are using. This is because there are so many types of colourblindness that would need to be accounted for, where many would have the same colour scheme for each.

5.8.5 Pilot, James

Because this scenario is just a pilot asking air traffic controllers to complete the calculation, this scenario can be tested equivalent to the first scenario where our program passes.

5.8.6 Qualified Calculator of Air Traffic Control, Ava

This scenario describes Ava wanting to change between different views on the application. Our application passes this exactly as described. Ava can switch between the side-view, top-view or simultaneous scenes from the toolbar.

6 Sprint 3 backlog

The sprint backlog is shown in the following table. The stories that are implemented follow the MoSCoW prioritisation:

- **Must**
- **Should**
- **Could**

Certain tasks are prioritized over others due to their direct relevance to the tasks within their respective increments. This incremental plan, summarily derived from user stories, primarily serves as a consultative guide for the final increment. One of the optional **could** functions, “Map & Zoom View”, was not implemented as originally planned because overlaying the software on an actual map proved ineffective. Instead, we implemented only the zoom function and made enhancements to the runway’s appearance to provide a more realistic visual representation as compensation

Story ID	Task	Workforce	Estimated	Actual
2	Configurable To Airport	Eric & Sam H	8	4
7	Authentication & Authorization	Eric	10	10
8	Export Reports	AJ	8	4
12	Lower Threshold Left	Muhammad	8	4
14	Notification	Eric & Sam H	8	4
15	Import & Export XML	Eric & Sam H	8	5
27	Rotate	Muhammad & AJ	6	5
11	Export Displays	Eric & AJ	5	5
29	Map & Zoom View	Eric	6	4
N/A	Total	(All)	67	45

7 Sprint 3 Burndown Chart

This is the burndown chart for our last increment.

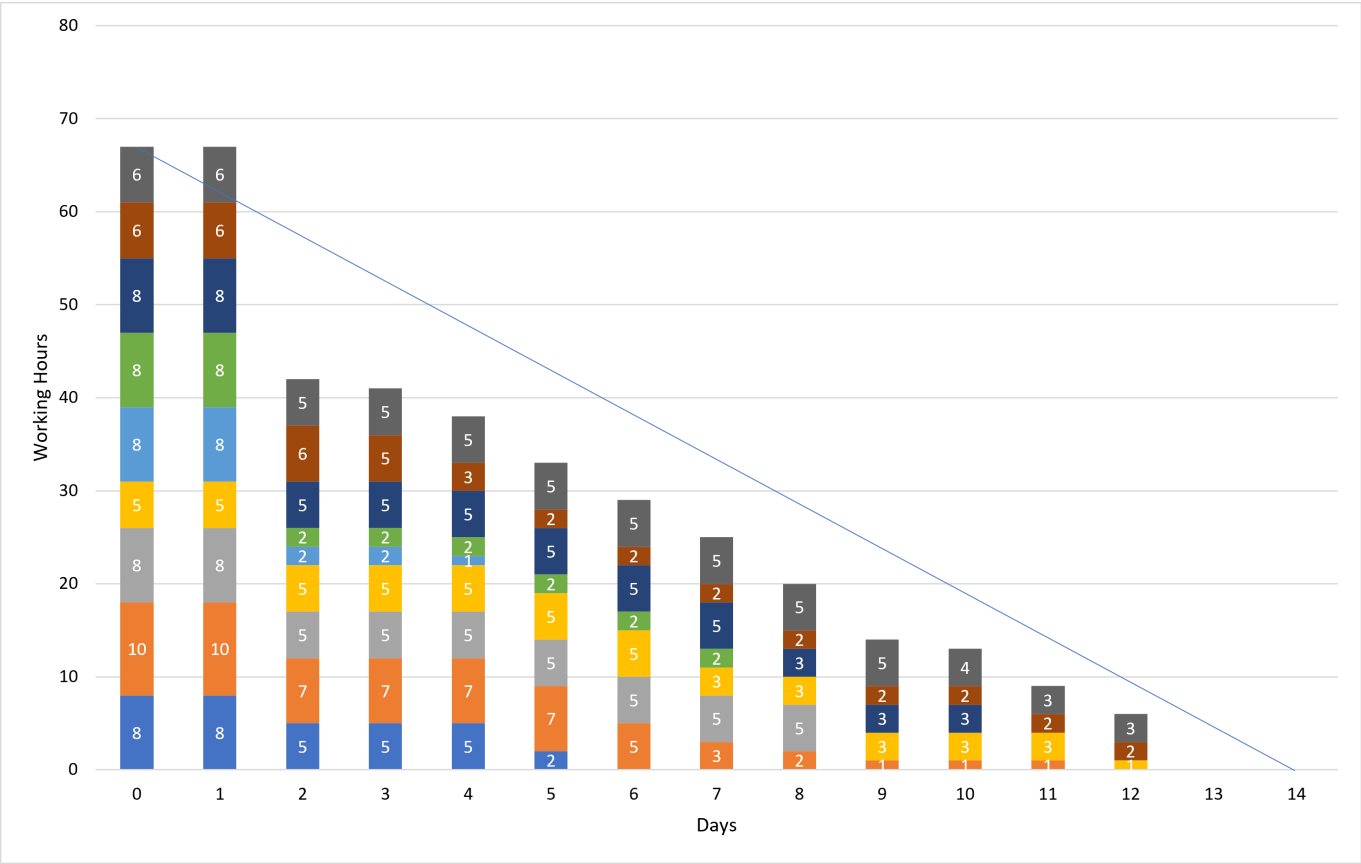


Figure 12: Sprint 3 initial burndown chart