UNIVERSITY OF Southampton

Group 23: Deliverable 3: Increment 2

COMP2211: Software Engineering Group Project

Jury D'Alessio (jd3n18), Mikolaj Kolybko (mk2u19), Rowan Kettle (rgk1g19), Velimir Nikolaev Anastasov (vna1u19), Charles Powell (cp6g18), Amir Abbasgholi Ghafghazi (aag1u18)

Electronics and Computer Science
University of Southampton

Contents

Introduction

This document discusses the results of the second increment of COMP2211: Software Engineering Group Project, for Group 23. We start by considering the feedback given for the First Increment followed by design details of the sprint, a product showcase and conclude with product testing, sprint reviewing and planning for the third increment.

1 Response to Increment 1 Feedback

On March 13th, Group 23 met with the group supervisor and an additional examiner to discuss the first increment that was submitted in the week prior. The supervisors provided the group with comprehensive feedback relating to the different aspects of the Increment as well as general comments discussing the product as a whole. A summation of this feedback, as well as the Group's response is provided below.

1.1 Application

1.1.1 Feedback

Overall, both supervisors were impressed with the application delivered in the first increment, being particularly pleased with it's "clear" and "intuitive" user-interface, although comments were made that it was possibly be "too plain".

During the live-demonstration of the application, a system error was present and highlighted by the supervisors. If the user was to edit details of the objects stored in the application (e.g., airports and obstacles), these changes were not reflected in the Dashboard's Calculation Panel. In order for the user to see these changes, the application would require restarting, so that the listing of items was refreshed.

Additionally, both supervisors pointed out that some of the methods used to gather user input can present themselves as being inconvenient. When adding or editing obstacles and airports, the user is displayed with forms that allow for them to input information and submit their request. However, in the case that the submitted information is error-nous (e.g., invalid data type for an input field), an error is displayed to the user and the form is closed. When the user re-opens the form, the previously entered data is lost, and the user must start the process again. The supervisors suggested that a more convenient approach would be for the form to display the error message and remain open, so that the user is not required to re-enter the information in the case of an invalid input.

1.1.2 Response to Feedback

In response to the above feedback, the Group has taken time in this increment to improve the application.

The System Controller and Calculation Panel were adjusted to fix the error that was present when editing objects whilst the logic of the input forms was changed so that they do not close in the case of the user submitting invalid inputs. These improvements, along with others, are demonstrated in Product Adjustments.

Regarding the comments of the user-interface being "too plain", the supervisors agreed with the justification provided by the team during the review meeting. It should also be noted that the features being implemented in the second increment should provide the user with a more complex interface, combating this issue.

1.2 Design

1.2.1 Feedback

The supervisors were pleased with design work carried out in preparation for the first increment. In particular, the wide range of UML tools presented a well structured development sprint.

The supervisors also noted the level of detail provided in the response to feedback, highlighting it's importance in the agile development methodology.

1.2.2 Response to Feedback

No response necessary. The Group will maintain the same standard of work within future product design.

1.3 Testing

1.3.1 Feedback

The supervisors found that the group had provided an extensive range of test cases to assess the product functionality that was developed in the first increment.

During the meeting, the group also discussed the further testing that will be performed using the 'TestFX' framework and additional unit tests. 'TestFX' allows for automated GUI testing to ensure that the application is presenting information correctly and responding to user-inputs appropriately. The additional unit tests will be developed to further examine some of the system core operations.

In response to the feedback provided for the product Envisioning, the Group included completion criteria as part of the amended sprint plan for the first increment. The supervisors found that the completion criteria had been used effectively within the Product Showcase to prove the completion of product backlog items - a successful form of product testing.

1.3.2 Response to Feedback

No response necessary. The Group will maintain the same standard of work within future product testing.

1.4 Planning

1.4.1 Feedback

The supervisors were impressed with the sprint review provided, and in particular the summation of successes and challenges experienced during the increment. The supervisors were also pleased with the burn-down chart that showcased the development progress of the group throughout the sprint, along with the provided justification of it's structure.

The inclusion of completion criteria within the second increment plan received positive feedback from the supervisors.

1.4.2 Response to Feedback

No response necessary. The Group will maintain the same standard of work within future planning stages.

2 Design Choices

In the first increment, the Model-View-Controller architecture was chosen as the system architecture. As this sprint aims to extend the functionality of the Increment 1 product, a change to the system architecture would not be appropriate or necessary. Therefore, there are no further design choices to be presented for this increment.

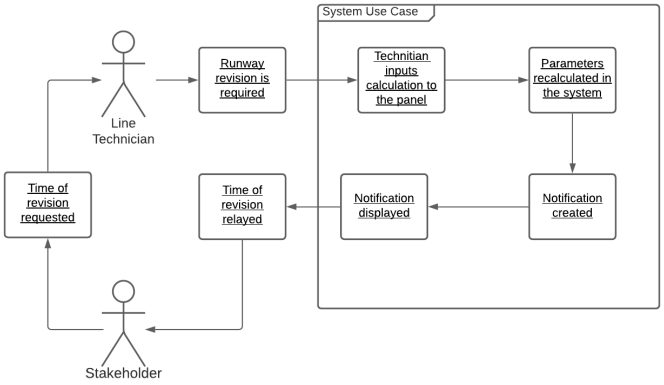
3 Design Artifacts

During the second development sprint, The group made use of UML techniques to effectively plan and design the product that would be produced. The resulting artifacts from this design process are presented in this section of the report. The presented artifacts should represent the product being developed in this sprint, and reflect how it is an extension of the Increment 1 product.

3.1 User Scenarios

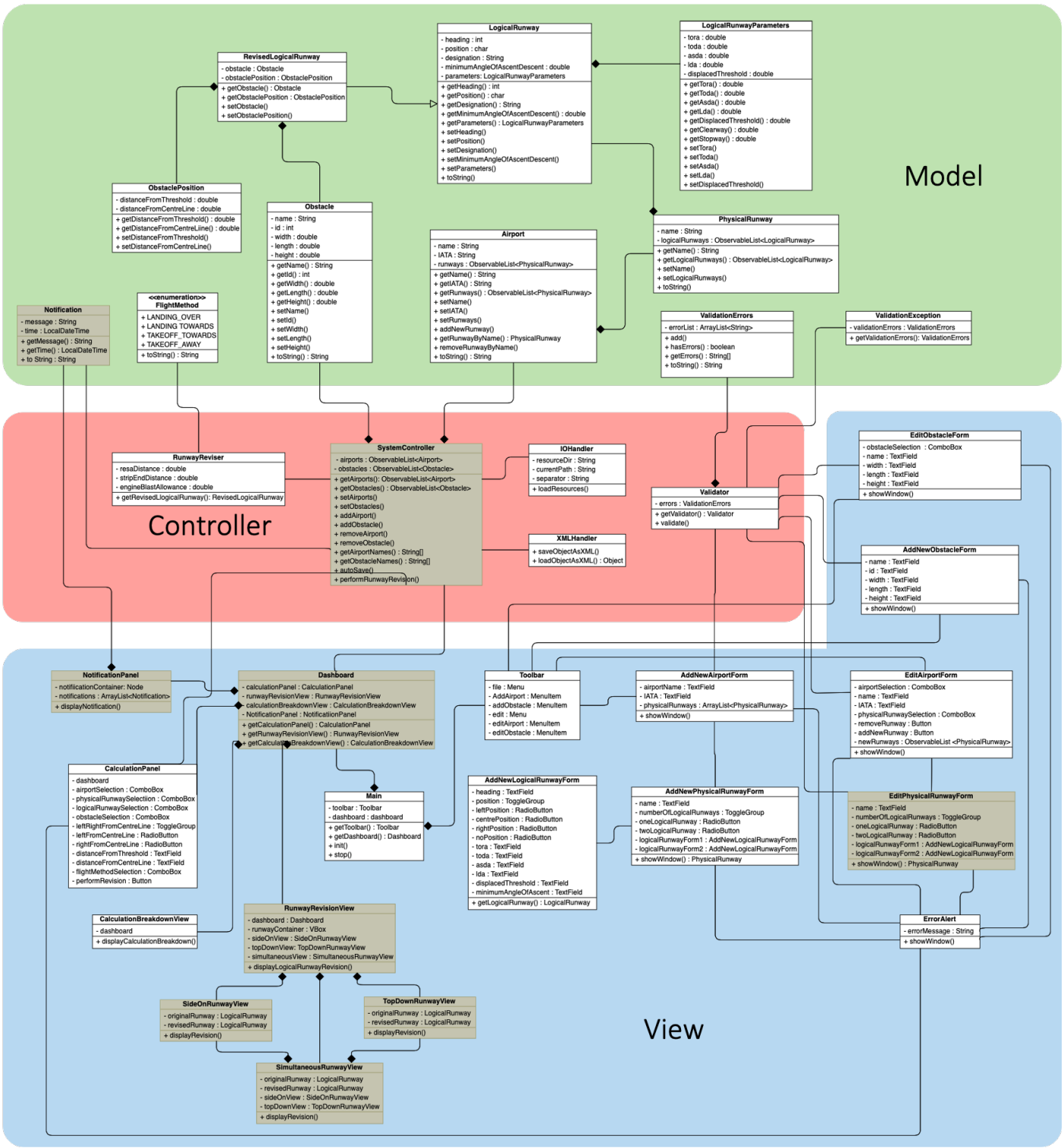
A single scenario was developed during the design process to illustrate how the primary stakeholders would interact with the system, and the value that they gain from using the product. This artifact is shown below and describes a scenario in which a Line Technician is asked for the time of when an obstacle appeared on a specific runway.

1. Line Technician notices obstacles on the runway.
2. Line Technician logs the obstacle into the Re-declaration system.
3. The change is updated and changes the runway parameters in the system.
4. A notification with the time of revision is logged
5. An interested stakeholder requests for the time of obstacle sighting to log for a report
6. Technician checks the notification log and relays the time of revision to the interested party



3.2 Class Diagram

The Class Diagram produced in the first Increment has been extended to support the functionality being developed in the second Increment. This diagram is shown below and showcases the classes within the system, along with their associated attributes and (public) methods. For convenience, the classes new to or that have been changed during this increment are highlighted.



3.3 Product Storyboards

Product story boards were devised during the design process to provide a clear indication of how the user-interface would be extended in this Increment. These story boards are shown below and a short description is provided with each to explain its context and use.

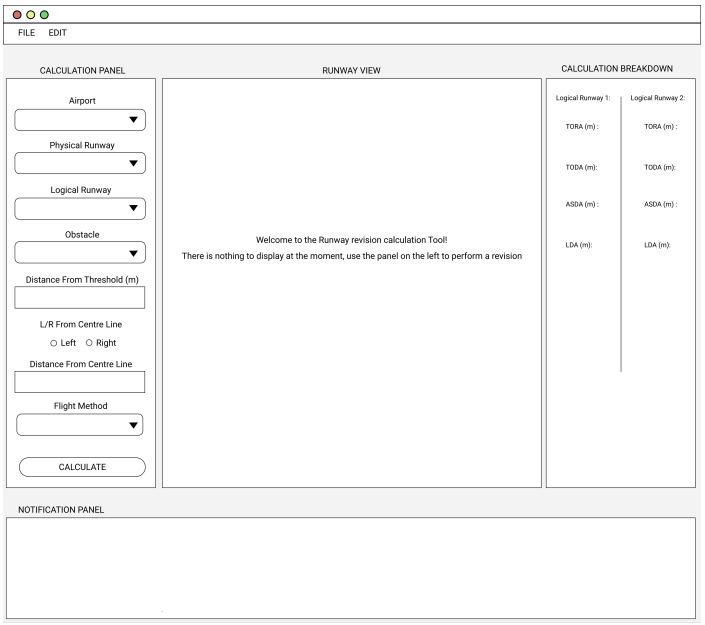
In accordance with the Increment plan, the only changes to the user-interface are present within the application Dashboard, and thus storyboards are provided for only this section of the application.

The story boards shown depict "mock" user-interfaces for each of the different screens within the program. Each "mock" user-interface showcases a core functionality of the program, and provides an insight into how a primary stakeholder would be able to make use of the system.

3.3.1 Dashboard

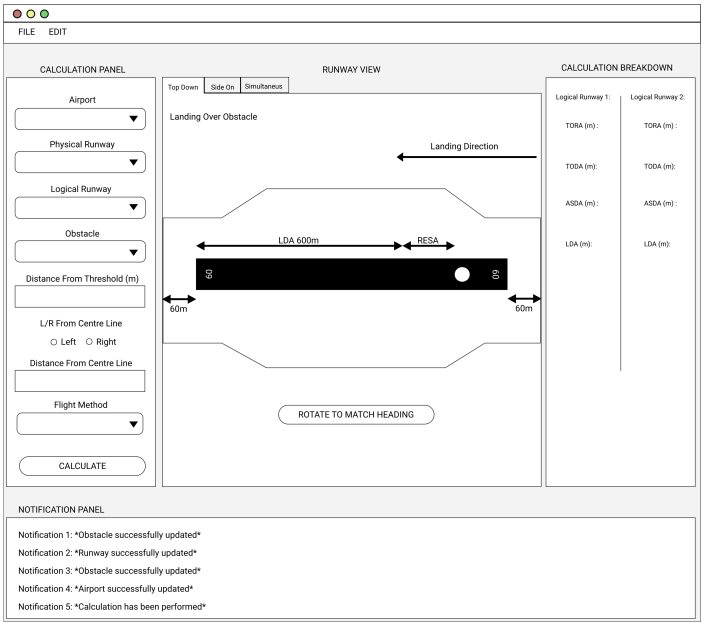
Dashboard : Main

The Dashboard should be extended to support the graphical representation of runway revisions and the display of system messages. The previous runway view will be replaced with a Tab Pane, and new 'Notifications' panel at the bottom of the application will display system messages. The tab pane should allow for the user to view the runway from different perspectives when performing revisions.



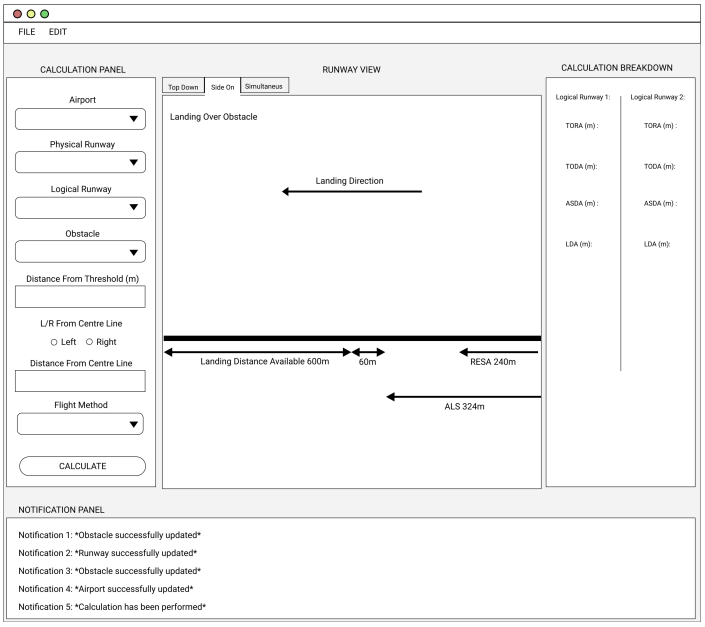
Dashboard : Top-Down Runway View

The top-down runway view will show the revision performed from a top-down perspective, displaying the revised runway parameters overlay-ed on an image of the runway. The view should also display the clear and graded area of the runway, and allow the rotation of the graphic to match the heading of the runway.



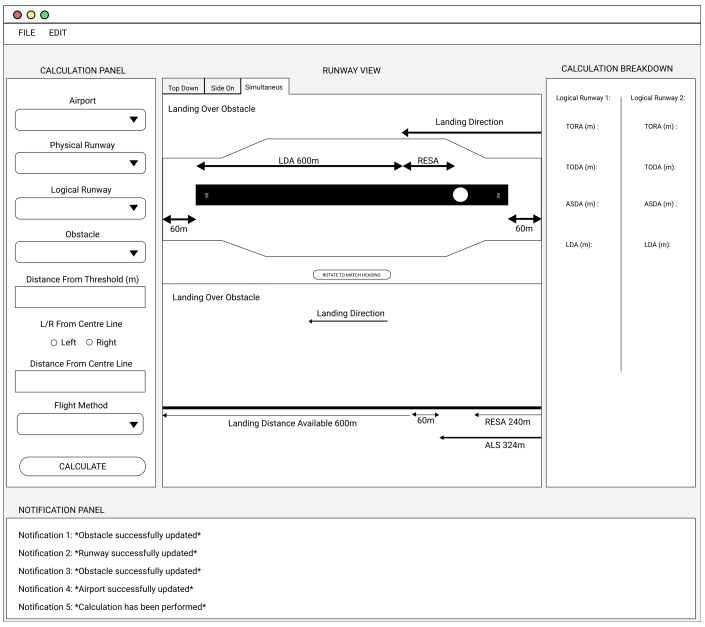
Dashboard : Side-On Runway View

The side-on runway view will show the revision performed from a side-on perspective, displaying the revised runway parameters overlay-ed on a image of the runway. The view should also display the ALS/TOCS value according to the obstacles placement.



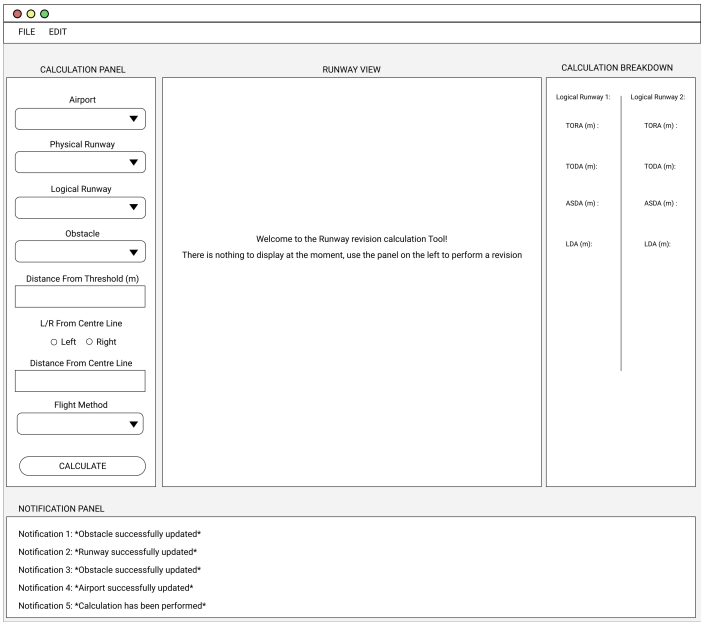
Dashboard : Simultaneous Runway View

The simultaneous runway view will combine the content from the top-down and side-on runway views to display both graphics simultaneously.



Dashboard : Notifications

When interacting with the application (e.g., adding a new airport), the result of the interaction shall be displayed as a notification within the 'Notifications' panel of the Dashboard. Notifications will be displayed as text, and list the details of the performed action, as well as a timestamp representing when the task was carried out.



4 Product Adjustments

This section of the document details the adjustments that were made to the first increment product following the sprint review meeting and feedback from the customer. These adjustments were made before development began on increment 2 backlog items, and aim to improve the overall usability of the program and ensure that the customer's requirements are met.

4.1 Response to Feedback

The following adjustments to the product were made following feedback from the customer, and aim to provide the customer with an application that better suits their requirements.

4.1.1 Input Window Remaining Open After Invalid Input

The logic of the input forms (used to gather information about objects) has been altered so that the windows remain open in the case of invalid input, instead of closing. With this adjustment, the user is no longer required to re-enter the data in the case of invalid input, but can simply change the invalid fields. It should also be noted that the edit airport and obstacle forms have been improved to extend their error detection.

These adjustments shall be shown to the customer during the next meeting, as they cannot be demonstrated through screenshots.

4.1.2 Fixing 'Calculation Panel' Bug

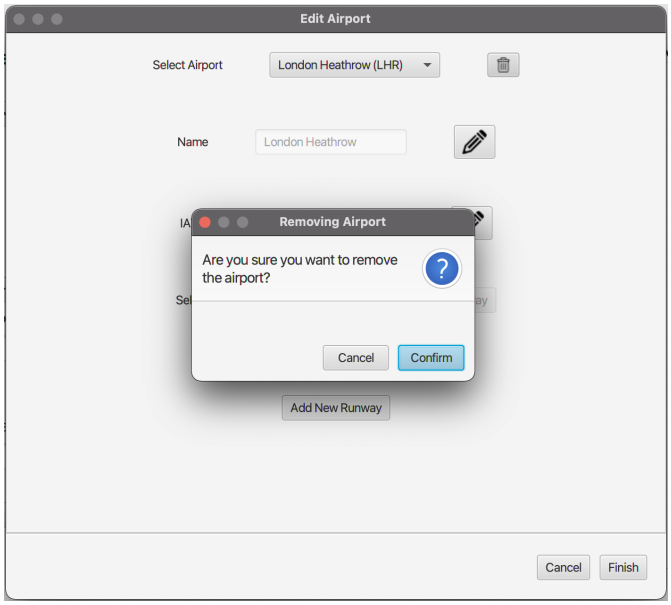
The Dashboard's 'Calculation Panel' has been adjusted to fix a bug that was present. When editing an airport/obstacle, the changes made are now updated within the 'Calculation Panel', without the application requiring a restart.

This adjustment shall be shown to the customer during the next meeting, as it cannot be demonstrated through screenshots.

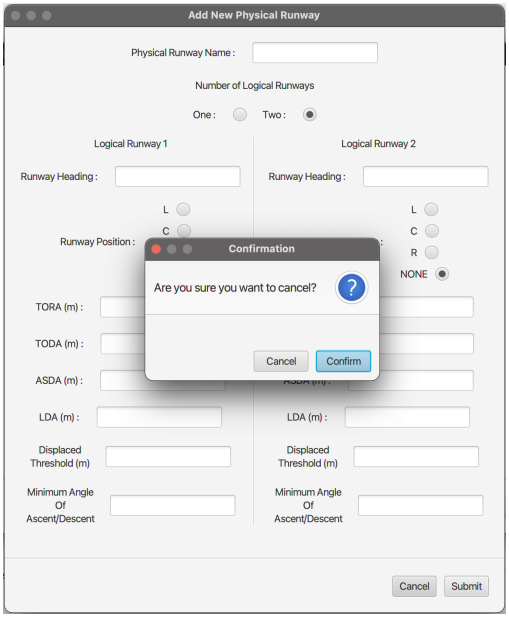
4.1.3 Adding Confirmation Windows

After discussions with the customer, it was agreed that the product would benefit from confirmation windows that would be displayed to the user when carrying out tasks. Such windows would be shown to the user when submitting input, or when editing system objects (e.g., deleting an airport), and would ensure that the user does not perform these actions accidentally. Ultimately, this makes the product more user-friendly and forgiving of mistakes.

The below screenshots show two examples of the confirmation windows that are displayed to the user. The first shows the message displayed to the user when attempting to delete an airport from the system, while the second shows the message displayed when the user selects to 'Cancel' their input when adding a new physical runway to an airport.



(a) Confirmation window displayed when removing an airport



(b) Confirmation window displayed after pressing 'Cancel'

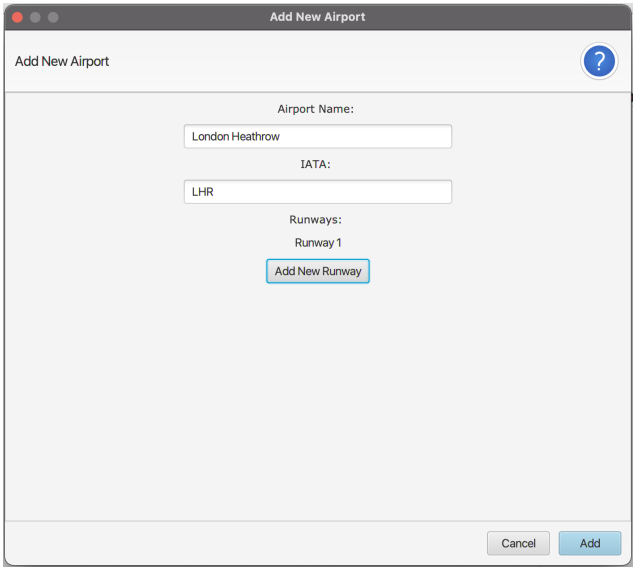
4.2 Improvements

The following adjustments were made in order to improve the application's usability and extend it's basic functionality.

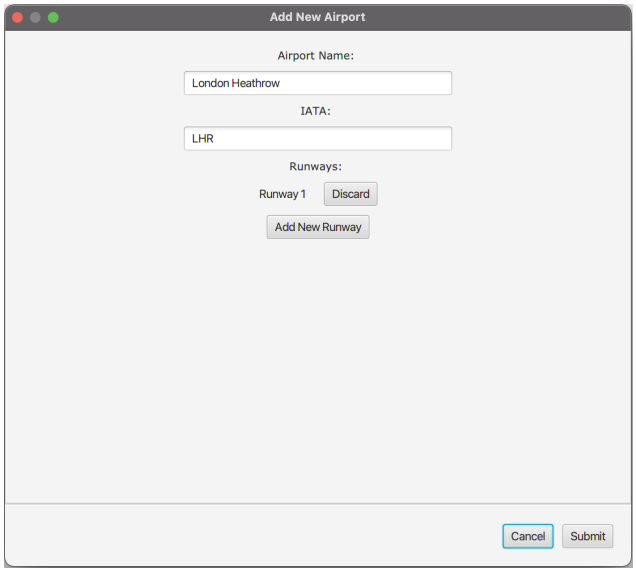
4.2.1 Discarding Added Runways When Configuring an Airport

The add/edit airport forms have been extended to allow for the user to discard an added runway whilst configuring an airport (i.e., undo the addition of the runway before submitting the airport). When configuring an airport, and after adding a runway to it, a button is now provided next to the runway's designation that allows for that said runway to be removed. Previously, in order to remove the added runway, the user was required to submit the airport completely, and remove the runway through the 'Edit Airport' window.

The below screenshots show a comparison of the 'Add Airport' input form before (left) and after this change (right).



(a) Increment 1 'Add New Airport' window

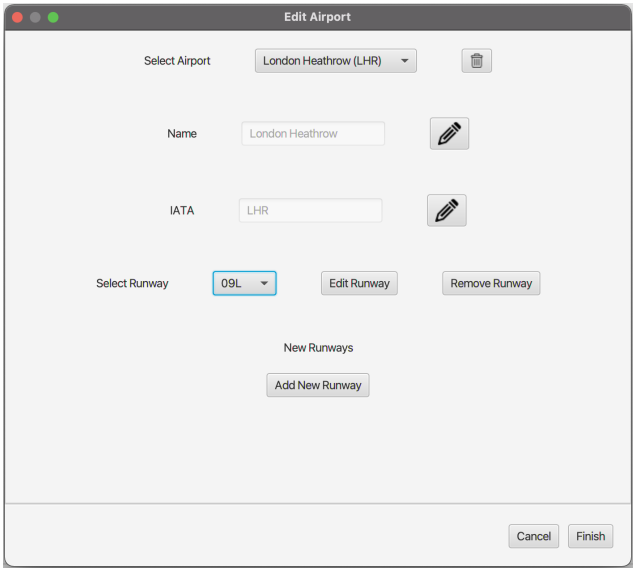


(b) Updated 'Add New Airport' window

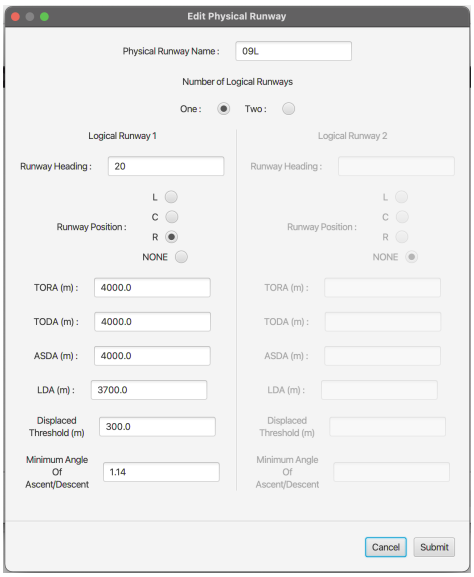
4.2.2 Editing Individual Runways

The user is now able to edit individual runways through the 'Edit Airport' window. After selecting an airport, and choosing one of the airport's runways, the user will now be able to select to edit this runway's details, giving them the ability to change all of the values of the runway that were set when the runway was created. Any changes the user makes are then immediately reflected in the application, and the updated runway is ready to use for revisions.

The below screenshots show both the updated 'Edit Airport' window that now allows the user to select a runway and edit it's details and the new 'Edit Runway' window that is displayed when to the user whilst editing a runway.



(a) Updated 'Edit Airport' window

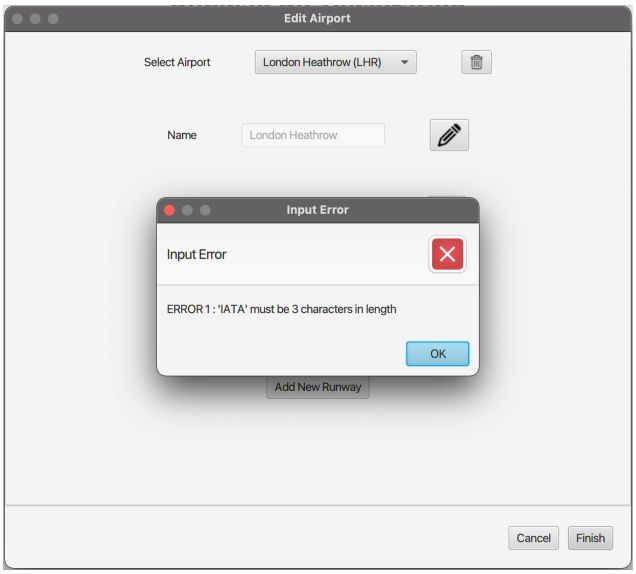


(b) New 'Edit Runway' window

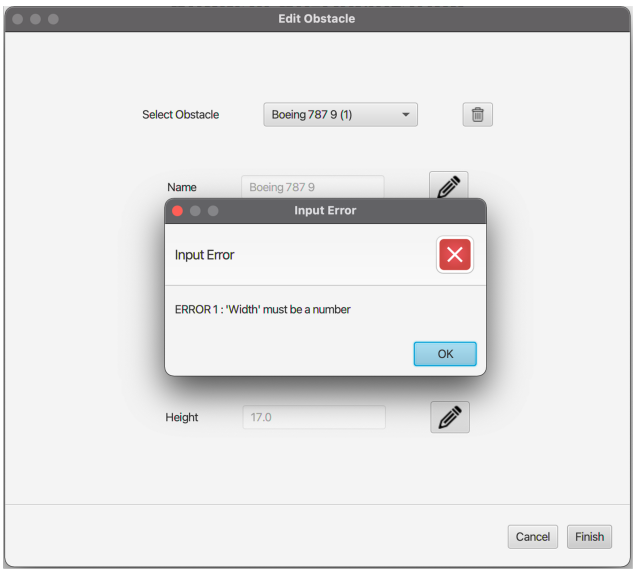
4.2.3 Improved Error Detection in 'Edit' Forms

The 'Edit Airport' and 'Edit Obstacle' forms have been improved to extend their error detection. When editing the details of an airport, or an obstacle, if the user provides invalid details (e.g., incorrect data types), the input will be rejected. Previously, error checking on objects was only performed when the objects were created, which allowed for the user to "bypass" the constraints on an object's details using the 'Edit' forms.

The below screenshots shows example error alerts that are displayed to the user when they submit invalid details regarding an airport (left) and obstacle (right).



(a) Updated 'Edit Airport' window detecting invalid input



(b) Updated 'Edit Obstacle' window detecting invalid input

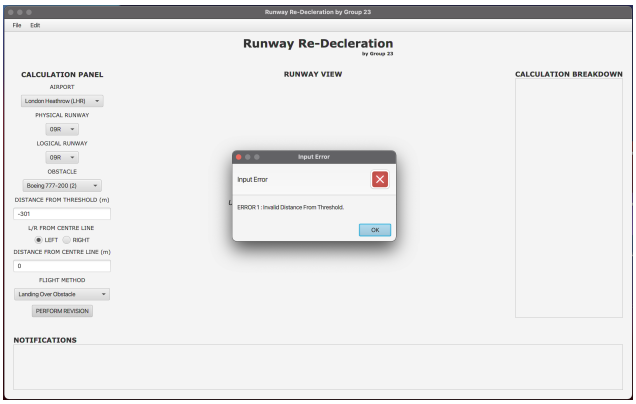
4.2.4 Improved Error Detection in 'Calculation Panel'

The application's error detection within the 'Calculation Panel' has been improved to increase product usability. When performing revisions, the user is required to enter the obstacle's distance from the runway threshold into the Calculation Panel. In the case where the runway being revised has a displaced threshold, it is possible for the obstacle to be positioned behind the threshold (i.e., negative distance), provided that it's distance behind the threshold is not greater than the size

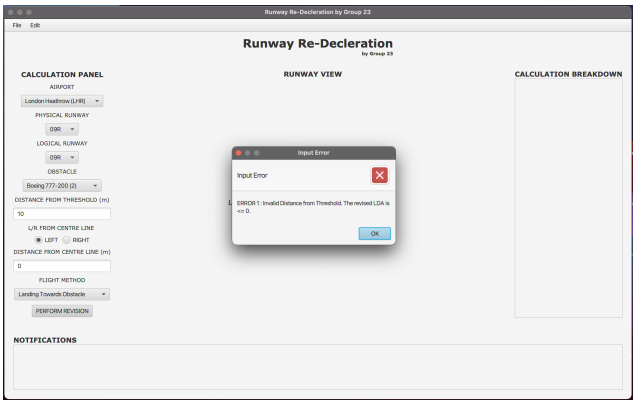
of the displaced threshold itself (which would place the obstacle off of the runway). Previously, if the user was to provide a negative value for the 'Distance From Threshold' parameter, the application would reject this input as invalid. With the adjusted product, the Calculation Panel only rejects the negative input in the case where it is greater than the size of the displaced threshold, and therefore allows for all possible revisions to be calculated..

Furthermore, in the case where the provided inputs result in a revised runway with parameter values less than or equal to zero, the Calculation Panel now rejects these inputs and displays an error alert to the screen. Previously, the user would be shown the updated runway parameters with negative values, which is not physically possible.

The below screenshots demonstrate the Calculation Panel rejecting inputs based on the above mentioned adjustments. The first image shows an error alert being displayed after entering an invalid distance from threshold based on the displaced threshold of the chosen runway. The second shows an error alert being displayed after the provided inputs result in non-positive parameter values, which are invalid.



(a) Updated 'Calculation Panel' detecting invalid distance



(b) Updated 'Calculation Panel' detecting negative results

5 Product Showcase

5.1 Product Value

The focus of this second sprint was to build on the core functionality implemented in the first increment. The Group sought to do this by providing the user with two features - a graphical representation of runway revisions, and a notification system.

In addition to the calculation breakdown (listed as text), the application has been extended to provide the user with a graphical representation of runway revisions. This graphical representation displays an image of the runway being revised, overlay-ed with the obstacle placement and updated parameters. The user is able to navigate between two perspectives of the runway, side-on and top-down, or can view both simultaneously. Furthermore, the top-down view shows the clear and graded area of the runway, whilst viewing from side-on displays the ALS/TOCS distance based on the obstacle's height and the runway's minimum angle of ascent or descent.

Providing the user with a graphical representation of revisions allows for them to gain a much more detailed understanding of how an obstacle is impacting the usable runway. Ultimately, this allows for the user to make much more informed and justified decisions when performing revisions, leading to an efficient working environment.

The notification system provides the user with feedback whilst interacting with the system, in the form of text messages within the 'Notifications' panel of the Dashboard. Notifications are created when the user completes tasks such as adding a new airport into the system, or completing a runway revision. Each notification displays a message and a timestamp, representing when the action described in the message was performed.

When working in as strict and regulated environment as the Aviation industry, it is crucial that the workers make certain that their tasks are completed correctly. The notification system offers the necessary feedback to ensure this is the case. Furthermore, the use of timestamps within the notifications provides the user/system administrator with a history of actions performed in the application, and can therefore be used as a record of airport activity.

It should also be noted that the adjustments made to the first increment product, as listed in Product Adjustments, provide the customer with a more functional and user-friendly application that better meets their requirements, increasing the value they receive as a result.

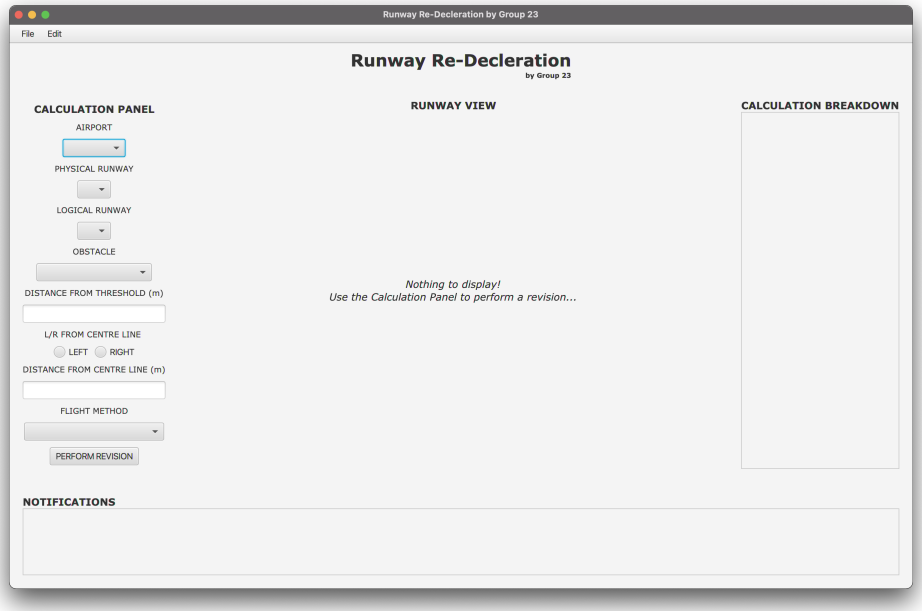
5.2 Product Demonstration

The Increment 2 product will now be showcased to demonstrate its capabilities. The different aspects of the system are presented in terms of this increment's product backlog items, with a description and image of the application being provided for each.

Note that, in some cases, the running of the application may be required to verify that a backlog item has been implemented correctly (e.g., notifications displayed when actions are performed).

5.2.1 Dashboard

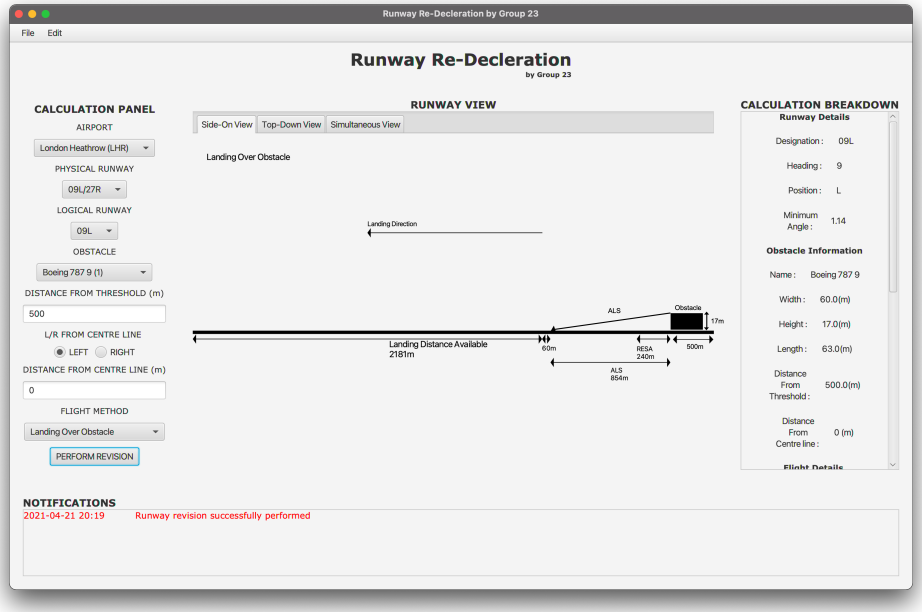
The 'Dashboard' functions as the home screen of the application, and has been updated to include a 'Notifications' panel (bottom) and support the graphical display of runway revisions. When a revision is performed, the 'Runway View' is converted to a tab pane that displays the results of the revision in a graphical form from both side-on and top-down perspectives. There is also a tab that allows the user to view both of these perspectives simultaneously.



5.2.2 Viewing Revisions: Side-On Runway View

ID	Identifier	Priority	Completion Criteria
8	Runway Side-On View	MUST	The runway can be viewed side on, and displays the runway itself, the obstacle and the updated runway parameters.
9	Runway Side-On View TOCS/ALS Slope (9)	SHOULD	When the runway is being viewed from the side, the TOCS/ALS slope is clearly shown.

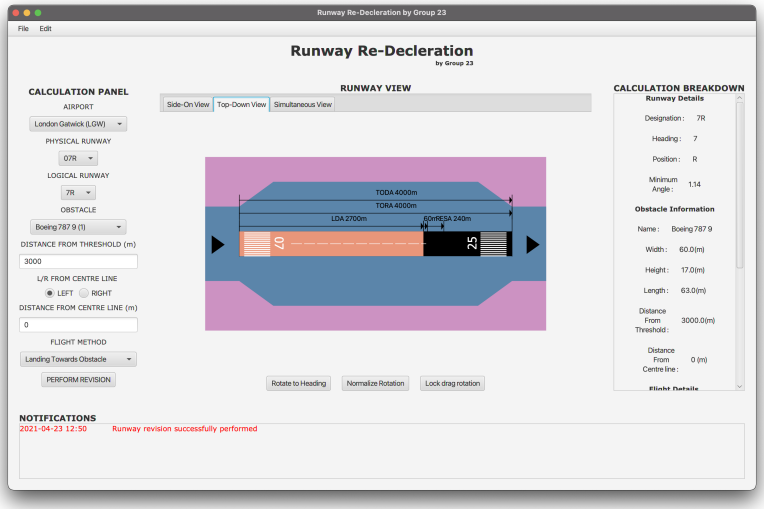
After performing a revision through the 'Calculation Panel', the user can view a side-on perspective of the revised runway within the 'Side-On' tab of the 'Runway View'. This perspective shows the updated runway parameters, the values that were involved in the calculation (including the ALS/TOCS) and the obstacle causing the re-declaration. The information is overlay-ed onto the image of the runway to provide the user with an appreciation of how the runway has been affected by the revision.



5.2.3 Viewing Revisions: Top-Down Runway View

ID	Identifier	Priority	Completion Criteria
10	Runway Top-Down View	**MUST**	The run-way and any obstacles and updated parameters can be viewed from top-down.
11	Runway Top-Down View Clear and Graded Area	**SHOULD**	When using the top-down viewing method, the clear and graded area is marked out and clearly visible.

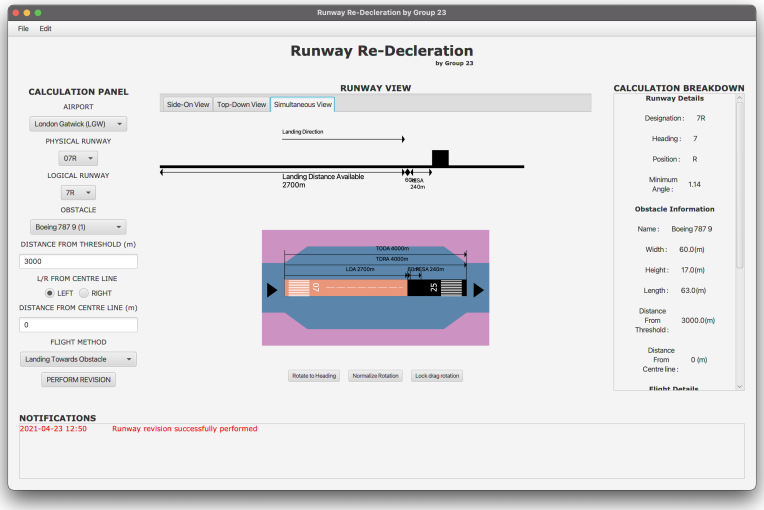
After performing a revision through the 'Calculation Panel', the user can view a top-down perspective of the revised runway within the 'Top-Down' tab of the 'Runway View'. This perspective shows the updated runway parameters, the values that were involved in the calculation and the obstacle causing the re-declaration. The information is overlay-ed onto the image of the runway to provide the user with an appreciation of how the runway has been affected by the revision. The graphic also showcases the clear and graded area surrounding the runway.



5.2.4 Viewing Revisions: Simultaneous Runway View

ID	Identifier	Priority	Completion Criteria
12	Simultaneous Runway View	**COULD**	When using side on and top-down views, the runway, its obstacles and updated parameters are all shown at the same time.

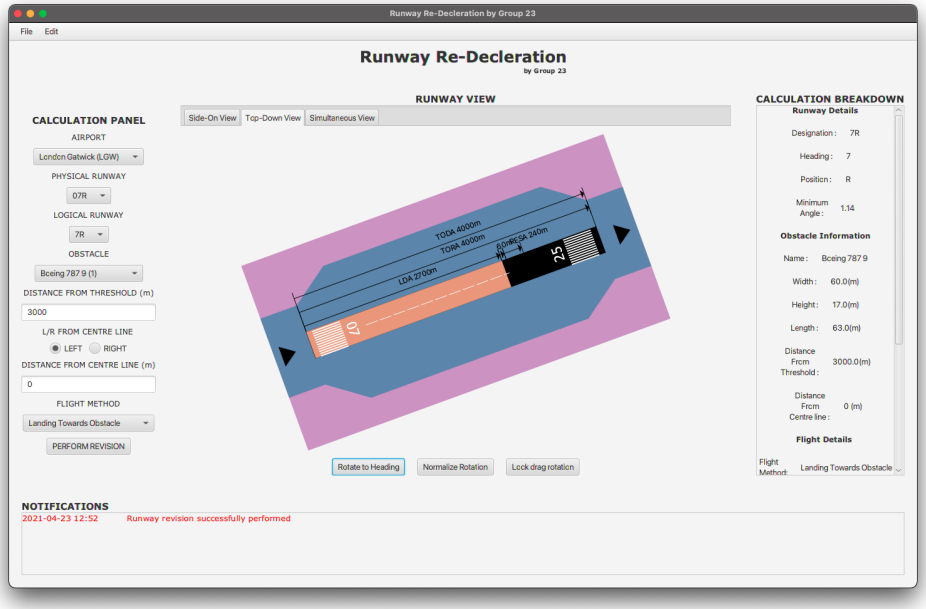
After performing a revision through the 'Calculation Panel', the user can navigate to the 'Simultaneous' tab within the 'Runway View' to be presented with a combination of both the top-down and side-on perspectives of the revision. The simultaneous view combines the displays from the 'Side-on' and 'Top-Down' tabs, presenting all of the information that is available in the individual sections.



5.2.5 Viewing Revisions: Rotating Top-Down Runway View

ID	Identifier	Priority	Completion Criteria
16	Runway Rotation	COULD	The runway rotates to match it's compass heading.

When viewing a revision inside the 'Top-Down' tab, the user is able to select the 'Rotate' button to rotate the graphic to match the heading of the runway currently being displayed. The system considers a heading of zero to be pointing North in the application (i.e., up) and all other headings can be derived from this basis. The 'Normalize' button can be used to 'un'-rotate the runway, back to its original position. The user is also able to drag the runway to rotate it manually.



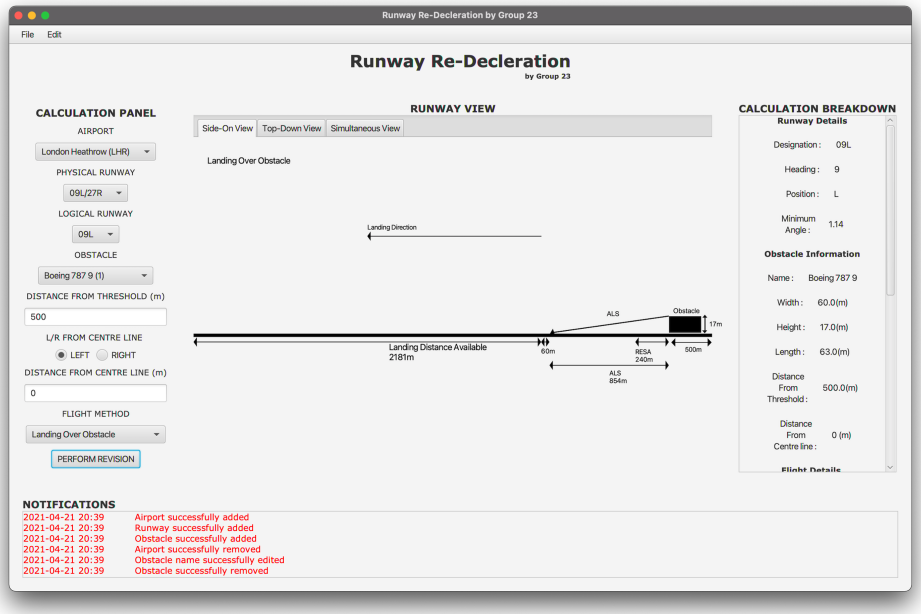
5.2.6 Notifications

ID	Identifier	Priority	Completion Criteria
21	Obstacle Placement Notification	SHOULD	When an obstacle has successfully been placed on a runway, a notification appears to say inform the user the obstacle has been placed.
22	Updated Runway Parameters Notification	SHOULD	When updated runway parameters have been calculated, a notification confirming this appears for the user to read.

A notifications system has been implemented to provide the user with messages when interacting with the application. Each message is displayed to the user as text in the 'Notifications' panel of the dashboard, describing the action performed and providing a timestamp of that said action.

In addition to notifying the user when a revision has been performed (as per the product backlog items), several other actions will also trigger notifications. The full list of actions is described below.

- Placement of obstacle/Performing a revision
- Adding an airport to the system
- Adding an obstacle to the system
- Removing an airport from the system
- Removing an obstacle from the system
- Editing an airport in the system
- Editing an obstacle in the system



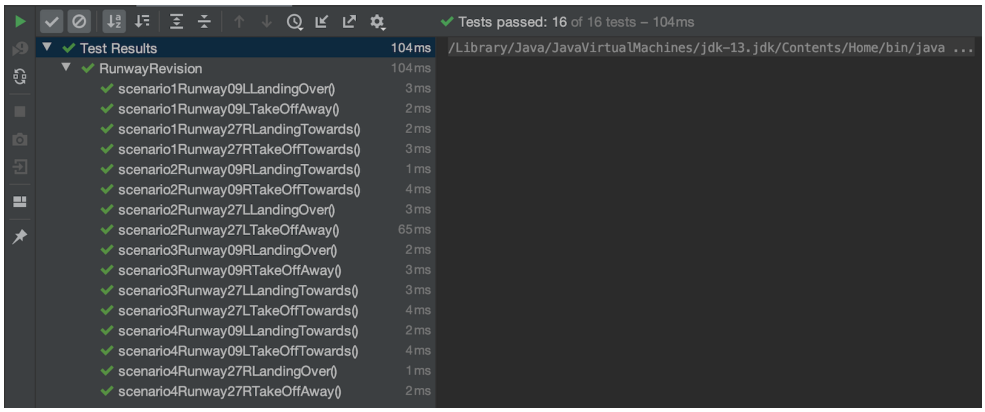
6 Product Testing

Following the testing carried out in the first increment, the Group has written further unit tests as well as performed extensive scenario testing. The combination of unit and scenario tests ensures that the application is correctly performing tasks and appropriately responding to user input.

6.1 Unit Testing

Automated unit tests were carried out using the 'JUnit' framework to assess the product's ability to perform runway revisions. The example calculations provided in the specification and the supplementary document were used to write test cases, which were then run against the application. These test cases ensure that the program is able to correctly calculate updated runway parameters when performing revisions.

The written unit tests can be found within the code submission for this increment. Shown below is screenshot taken from the IntelliJ IDEA IDE, that provides the results of the test suite when run on the application.



6.2 Scenario Testing

The Group defined a series of scenario tests for the product, which were both manually carried out, and automated using the 'TestFX' framework. The 'TestFX' framework performs testing by taking control of the mouse pointer and keyboard to operate the JavaFX application and evaluate it's behaviour. Performing the tests manually allows for the identification of abnormal behaviour that would otherwise go undetected by the automated tests.

6.2.1 Description of Tests

The below table describes each of the tests carried out on the product during scenario testing, providing the actions performed in the test, the expected outcome of the application after these actions, and the result of the test when performed manually.

ID	Identifier	Scenario	Expected Outcome	Pass/Fail
1	**Adding Airport**	The user opens the 'Add Airport' window from the 'File' menu, enters the details of an airport and submits.	A new airport is created is using the input details and a notification of this event is displayed to the user. The user is able to use the airport in revisions, and edit it's properties through the 'Edit' menu.	**PASS**
2	**Adding Airport: Invalid Input**	The user opens the 'Add Airport' window from the 'File' menu, enters the details of an airport details and submits. The airport details are invalid.	A new airport is not created, and an error window is displayed to the user informing them of why the given inputs are not valid.	**PASS**
3	**Adding Runway to Airport**	The user opens the 'Add New Runway' window from the 'Add Airport' window, enters the details of an airport of the runway and submits.	A new physical runway is created and displayed to the user inside the 'Add Airport' window. A button is displayed next to the runway that allows for the new runway to be removed from the airport	**PASS**
4	**Adding Runway to Airport: Invalid Input**	The user opens the 'Add New Runway' window form the 'Add Airport' window, enters the details of a runway and submits. The runway details are invalid.	A new physical runway is not created, and an error window is displayed to the user informing them of why the given inputs are not valid. The 'Add New Runway' window remains open to allow the user to correct their mistakes.	**PASS**
5	**Adding Obstacle**	The user opens the 'Add Obstacle' window from the 'File' menu, enters the details of an obstacle and submits.	A new obstacle is created using the input details and a notification of this event is displayed to the user. The user is able to use this obstacle in runway revisions, and edit it's properties in the 'Edit' menu.	**PASS**
6	**Adding Obstacle: Invalid Input**	The user opens the 'Add New Obstacle' window from the 'File' menu, enters the details of an obstacle and submits. The obstacle details are invalid.	A new obstacle is not created, and an error window is displayed to the user informing them of why the given inputs are not valid.	**PASS**
7	**Removing Airport**	The user opens 'Edit Airport' window from the 'Edit' menu, chooses an airport stored in the system and selects to remove it.	The user is presented with an alert message, asking them to confirm the decision to remove the airport. If the user confirms, the airport is removed from the system completely and a notification is displayed to the dashboard informing the user of this event. If the user does not, no action is taken.	**PASS**
8	**Editing Airport Properties**	The user opens the 'Edit Airport' window from the 'Edit' menu, chooses an airport stored in the system, and edits it's details.	The airport's details are changed according to the given input, and a notification is displayed to the dashboard informing the user of this event.	**PASS**
9	**Editing Airport Properties: Invalid Input**	The user opens the 'Edit Airport' window from the 'Edit' menu, chooses an airport stored in the system and edits it's details. The updated details are invalid.	The airport's details are not changed to the updated values, and an error window is displayed to the user informing them of why the given inputs are not valid.	**PASS**
10	**Editing Airport: Adding Runway**	The user opens the 'Edit Airport' window from the 'Edit' menu, chooses an airport within the system, selects to 'Add New Runway', enters in the details of a runway and submits.	A new physical runway is created and displayed to the user inside the 'Edit Airport' window. A button is displayed next to the runway that allows for the new runway to be removed from the airport.	**PASS**

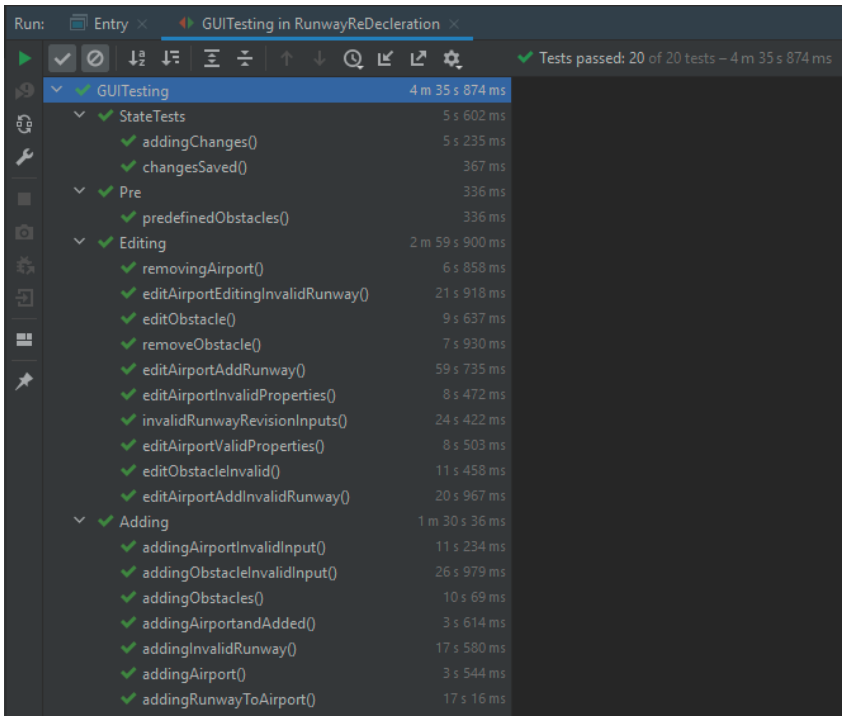
11	**Editing Airport: Adding Runway: Invalid Input**	The user opens the 'Edit Airport' window from the 'Edit' menu, chooses an airport within the system, selects to 'Add New Runway', enters in the details of a runway and submits. The runway details are not valid.	A new physical runway is not created, and an error window is displayed to the user informing them of why the given inputs are not valid. The 'Add New Runway' window remains open to allow the user to correct their mistakes.	**PASS**
12	**Editing Airport: Editing Runway**	The user opens the 'Edit Airport' window from the 'Edit' menu, chooses an airport within the system, chooses one of the airport's runways and selects to edit the runway. The user enters the updated details of the runway into the 'Edit Runway' window, and submits.	The runway's details are changed to the updated values, and a notification is displayed to the dashboard informing the user of this event.	**PASS**
13	**Editing Airport: Editing Runway: Invalid Input**	The user opens the 'Edit Airport' window from the 'Edit' menu, chooses an airport within the system, chooses one of the airport's runways and selects to edit the runway. The user enters the updated details of the runway into the 'Edit Runway' window, and submits. The runway details are not valid.	The runway's details are not changed to the updated values, and an error window is displayed to the user informing them of why the given inputs are not valid.	**PASS**
14	**Editing Airport: Removing Runway**	The user opens the 'Edit Airport' window from the 'Edit' menu, chooses an airport within the system, chooses one of the airport's runways and selects to remove the runway.	he user is presented with an alert message, asking them to confirm the decision to remove the runway. If the user confirms, the runway is removed from the system completely and a notification is displayed to the dashboard informing the user of this event. If the user does not, no action is taken.	**PASS**
15	**Removing Obstacle**	The user opens 'Edit Obstacle' window from the 'Edit' menu, chooses an obstacle stored in the system and selects to remove it.	The user is presented with an alert message, asking them to confirm the decision to remove the obstacle. If the user confirms, the obstacle is removed from the system completely and a notification is displayed to the dashboard informing the user of this event. If the user does not, no action is taken.	**PASS**
16	**Editing Obstacle Properties**	The user opens the 'Edit Obstacle' window from the 'Edit' menu, chooses an obstacle stored in the system, and edits it's details.	The obstacle's details are changed according to the given input, and a notification is displayed to the dashboard informing the user of this event.	**PASS**
17	**Editing Obstacle Properties: Invalid Input**	The user opens the 'Edit Obstacle' window from the 'Edit' menu, chooses an obstacle stored in the system and edits it's details. The updated details are invalid.	The obstacle's details are not changed to the updated values, and an error window is displayed to the user informing them of why the given inputs are not valid.	**PASS**

18	**Performing Runway Revisions**	The user uses the 'Calculation Panel' on the 'Dashboard' to select a logical runway and obstacle, and provides the details of revision.	A notification is displayed to the 'Dashboard' informing the user of the runway revision, it's results are displayed to the user in the 'Runway View' panel, and a breakdown of the calculation's are displayed in the 'Calculation Breakdown' panel. The user is able to view the revised runway from a side-on or top-down perspective, as well as viewing both perspectives simultaneously.	PASS
19	**Performing Runway Revisions: Invalid Input**	The user uses the 'Calculation Panel' on the 'Dashboard' to select a logical runway and obstacle, and provides the details of revision. The provided values are invalid.	The revision is not calculated or displayed to the user. An error window is displayed to the user informing them of why the given inputs are invalid.	PASS
20	**Changes Saved After Application Close**	The user makes changes to the system objects within the application, closes the application, and then re-opens it.	The changes the user had made to the systems objects before closing the application have persisted.	PASS
21	**Pre-defined Obstacle List**	The user opens the application for the first time, and views the obstacles stored in the system.	The user is presented with a pre-defined list of commonly occurring airfield obstacles that are ready for use in revision calculations.	PASS

6.2.2 Automated Test Results

The below screenshot shows the output from the IntelliJ IDEA IDE when the above scenario tests were ran using the 'TestFX' framework. The written test cases can be found within the code submission for this increment.

Note that no automated test case is provided for tests 12, 14 and 18, as the framework was not able to properly assess the state of the program following the scenario. However, the correctness of these tests as determined using manual testing as described above.



7 Second Increment Sprint Review

7.1 Sprint Review

Following the completion of this development sprint, the group met to conduct a sprint review. The group shared their opinions on the success of the sprint, and considered areas that could be improved for further sprints. This review allowed group members to reflect on their work, and ensures an efficient and agile development process.

7.1.1 Successes

The group were pleased with the outcome of the sprint, and thought it necessary to highlight certain aspects that helped achieve this outcome. The group placed a lot of effort into taking on board the customers' feedback from the Increment 1 review meeting. Upon receiving this feedback, the group made appropriate changes to the product to ensure that the application meets the customers requirements. Time was also taken in the sprint to improve the overall usability of the product, implementing features that help to make the product more user-friendly and therefore more fit for purpose.

Additionally, the group placed a key focus on the design of the product at the start of this sprint, allowing for a more efficient development process. The Group also took time towards the end of the sprint to perform extensive testing of the application. Testing is a vital part of the development process as it ensures that the product is operating correctly and allows for the identification of any bugs or flaws in the system. This testing was done thoroughly and efficiently, so as to allow for any detected issues to be resolved before the end of the sprint.

7.1.2 Challenges

Despite being pleased with the progress made in this increment, the Group agrees that there is still room for improvement.

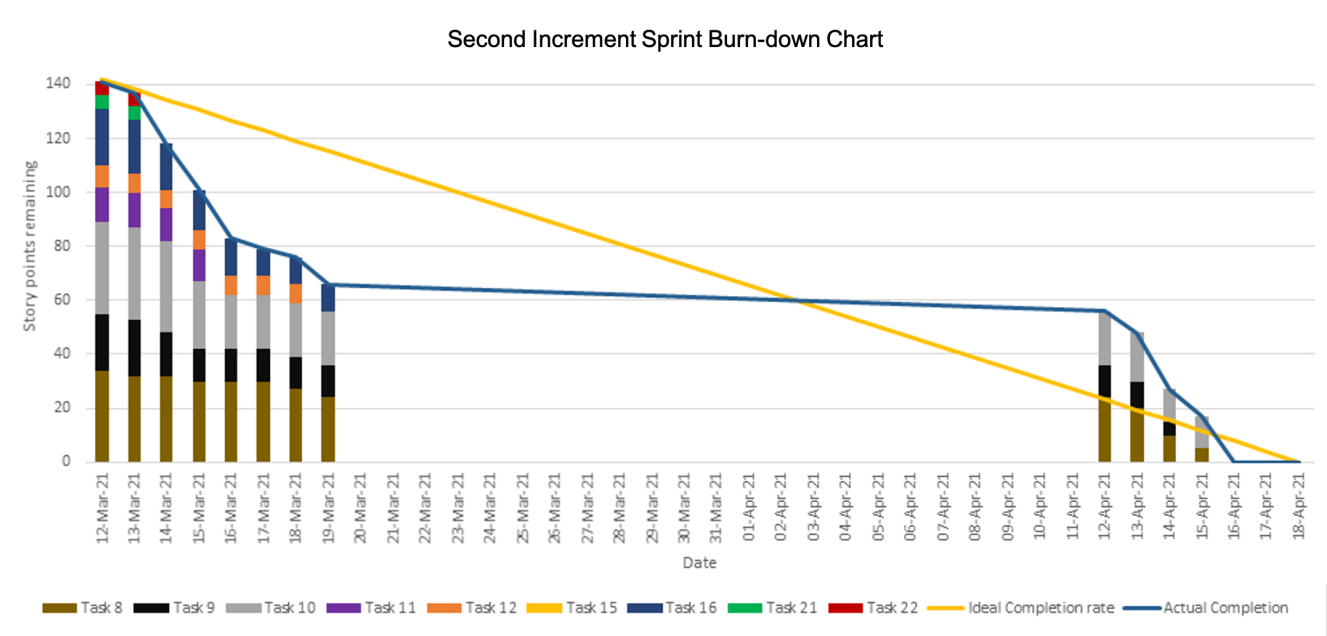
One of the challenges faced by the Group was managing the workload of this project against that of the other modules currently being studied by group members. This challenge was handled by re-distributing tasks amongst group members if and when they were available, following agile methodologies.

Another challenge came when identifying which packages to use for the graphical representation of runway revisions. The Group discussed whether or not a secondary graphical framework, Java.awt, should be implemented, or if the Group should continue to use only the current framework, JavaFX. After deliberation, the Group came to the conclusion that the most suitable option was to remain using JavaFX to maintain the structure of the code, despite the inconvenience it presents when aligning objects.

7.2 Sprint Burn-down Chart

The below chart shows the timeline of completion for product backlog items across the course of the sprint. The group began the sprint by tackling issues raised in the customers' feedback and designing the second increment product. A focus on design in the early stages of the sprint allowed for an efficient development process.

Note that, while the day-zero burn-down chart had originally spread the sprint over the Easter break, the team decided to not use the break for project development, and thus the burn-down chart does not show any tasks being completed over this time period.



8 Third Increment Plan

During the third increment, the Group will start by developing the last of the product backlog items. This involves extending the application to allow the user to import and export system objects (e.g., airports and obstacles) into and from the application in the form of XML files. An Airfield Operations Manager/Line Technician will be able to use the third increment product to transfer objects between different systems without having to manually declare them, allowing for a more efficient working environment.

Additionally, the final sprint will be used to extend the system beyond it's initial requirements, providing a primary stakeholder with a number of additional features. These extensions are detailed in Product Extensions.

8.1 Product Backlog Items

The Table below outlines the backlog items that the Group aim to complete in the upcoming increment. For each item, the number in brackets signifies it's ID, referencing the product backlog created in the envisioning stage of the development. The priority of each item is also provided, indicating how important it is that each item implemented.

Increment 3 Backlog Items	**Priority**
XML Airport Import (17)	**COULD**
XML Obstacle Import (18)	**COULD**
XML Airport Export (19)	**COULD**
XML Obstacle Export (20)	**COULD**

8.2 Product Extensions

As mentioned in the Project Envisioning, the Group plans to use the third increment to extend the product beyond it's initial requirements. The below table lists the extensions that the group aims to complete in this increment in addition to the above mentioned backlog items. Each extension is described in terms of a User Story, and has an allocated priority.

Note that, while items 24, 26, 27 and 28 were present in the original product backlog (Project Envisioning), items 32 and 33 are new.

ID	**Identifier**	**Description (User Story)**	**Priority**
24	Zoom and Pan	As an Airfield Operations Manager, I want to be able to zoom and pan on the view so that I can examine a runway in more detail.	**WON'T**
26	JPEG Export	As an Airfield Operations Manager, I want to be able to export the views to JPEG image so that I can share/use the view outside of the system.	**WON'T**
27	PNG Export	As an Airfield Operations Manager, I want to be able to export the views to PNG image so that I can share/use the view outside of the system.	**WON'T**
28	GIF Export	As an Airfield Operations Manager, I want to be able to export the views to GIF image so that I can share/use the view outside of the system.	**WON'T**
31	Print Results	As an Airfield Operations Manager, I want to be able to print the results of the current scenario so that the information can be shared/used outside of the system.	**WON'T**
32	Custom Object Images	As an Airfield Operations Manager, I want to be able to load custom images for the each of the objects stored in the system, so that I am provided with a more accurate representation of the object when performing a revision.	**WON'T**
33	Notifications Export	As an Airfield Operations Manager, I want to be able to export the system notifications, so that I have a record of the system activity.	**WON'T**

8.3 Sprint Plan

The table below breaks down the backlog items to be developed in Increment 3, and provides an estimation as to the number of story points that will be required for each. The group has continued to use the original choice of the Fibonacci Series as the story point range, which maintains consistency throughout the project. The estimated number of story points required for the increment is 98.

ID	Identifier	Sub-tasks	Story Points
17	XML Airport Import	• Develop Back-End • Develop Middle-tier • Delevop UI • Testing	**3**
18	XML Obstacle Import	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**3**
19	XML Airport Export	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**5**
20	XML Obstacle Export	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**5**
24	Zoom and Pan	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**21**
26	JPEG Export	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**8**

27	PNG Export	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**8**
28	GIF Export	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**13**
31	Print Results	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**8**
32	Custom Object Images	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**21**
33	Notifications Export	• Develop Back-End • Develop Middle-tier • Develop UI • Testing	**3**

8.4 Completion Criteria

The table below outlines the criteria that must be met to deem an item from the product backlog complete. It also includes the priority of each item, indicating the importance of meeting the provided criteria.

ID	Identifier	Priority	Completion Criteria
17	XML Airport Import	**COULD**	Airports and their runways will be able to be imported into the system via XML.
18	XML Obstacle Import	**COULD**	Predefined obstacles will be able to be imported into the system via XML.
19	XML Airport Export	**COULD**	Airports and it's runways can be exported into XML for use in other systems.
20	XML Obstacle Export	**COULD**	Obstacles can be exported into XML for use on other systems.
24	Zoom and Pan	**WONT**	The view of the runway can be zoomed in on and panned to get alternate views of the runway.

26	JPEG Export	**WONT**	The views shown on the screen can be exported into a JPEG image file.
27	PNG Export	**WONT**	The views shown on the screen can be exported into a PNG image file.
28	GIF Export	**WONT**	The views shown on the screen can be exported into a GIF image file.
31	Print Results	**WONT**	The results of the current scenario can be printed for use externally.
32	Custom Object Images	**WONT**	The user can choose an image for a certain obstacle which will be then scaled to give an accurate representation of the current scenario.
33	Notifications Export	**WONT**	The notifications from the current session can be exported to show an accurate log of system activity.

8.5 Day-Zero Burn-down Chart

The below chart describes the Day Zero Burn-down for the third increment. The chart shows the 'Ideal Completion Rate' of tasks throughout the sprint in terms of their corresponding story points, with all tasks yet to be completed on day-zero of the sprint.