SOFTWARE ENGINEERING GROUP PROJECT: RUNAWAY RE-DECLARATION

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

Author By: Group 48

University of Southampton

1 Introduction

This increment report details our progress to our runway re-declaration project which primarily focusing on our continuing phase from the Increment 1. This report will first introduce the explanation of improved design artifacts in the order of storyboards, scenarios and UML diagrams. It then move on to discuss the tests we applied on our software. Additionally, it will include our usual burndown charts, work summaries and our response to the feedback from the last increment. Alongside our report, the project itself is enclosed within a separate zip file, accessible for review. It utilises a Maven JavaFX installation which can then be ran through a normal JavaFX execution.

2 Response to feedback

The feedback we were given was to split up the UML diagrams and make sure that they are more readable for the markers. We were also told to provide more in-depth explanations for the diagrams provided in our report to make the document easier to follow. We have incorporated this feedback by enlarging the sizes of the images for all UML diagrams. Additionally, for the class diagram, all of the content that was previously too small to read was described by an ID, that was later further expanded upon to allow for greater ease of understanding.

3 Storyboard

The storyboards have been further developed in this increment. We made a decision to balance the amount of information that is displayed to the user, so that the interface doesn't contain too much information and it's easier for the user to extract useful information. This would be explained in more details in each of the view. Some new key design artifacts are:

- The user could choose the operation (landing / taking off) of the target plane. The airplane will be displayed on the runway in the correct relative position depending on the operation.
- The user could choose the designator the target airplane is using. The indicators for that specific runway will be displayed when one of the toggle button is selected. This allows all the indicators displayed in the correct position.
- For the obstacle, the user now could define the distance from the centreline. This allows the system to reject the obstacle if it's located outside the worth consideration zone.
- The user could define the distances from each of the thresholds, these values are used directly for the re-calculations and the results will be displayed in the results box.

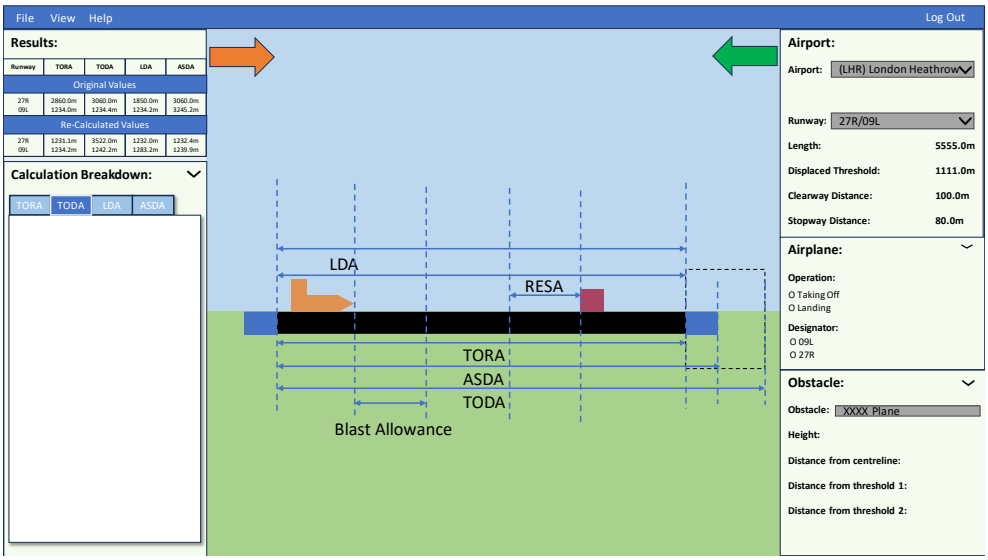


Figure 1: In the side-view scene, airplanes and indicators could be displayed in both direction, representing the current designator that is chosen.

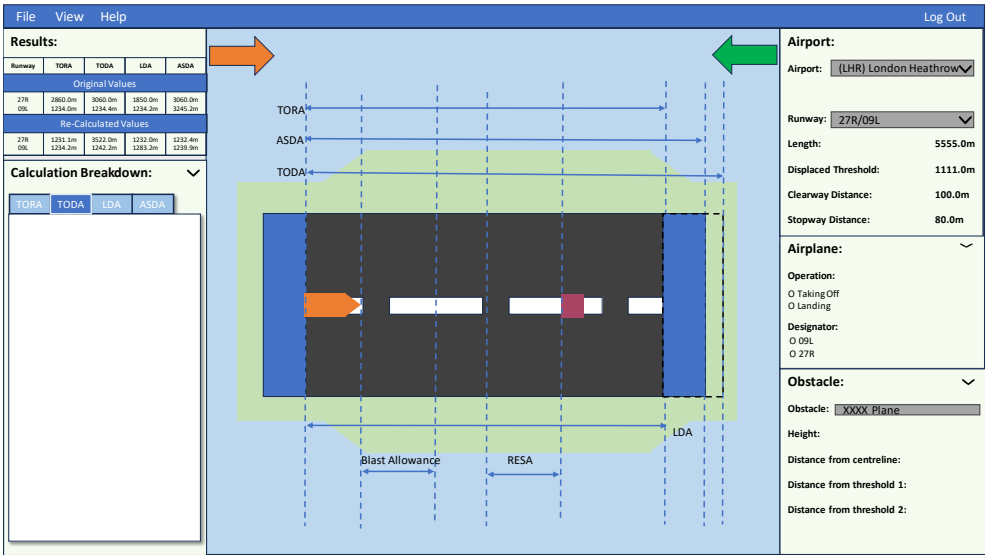


Figure 2: In the top-view scene, airplanes and indicators could also be displayed in both direction.

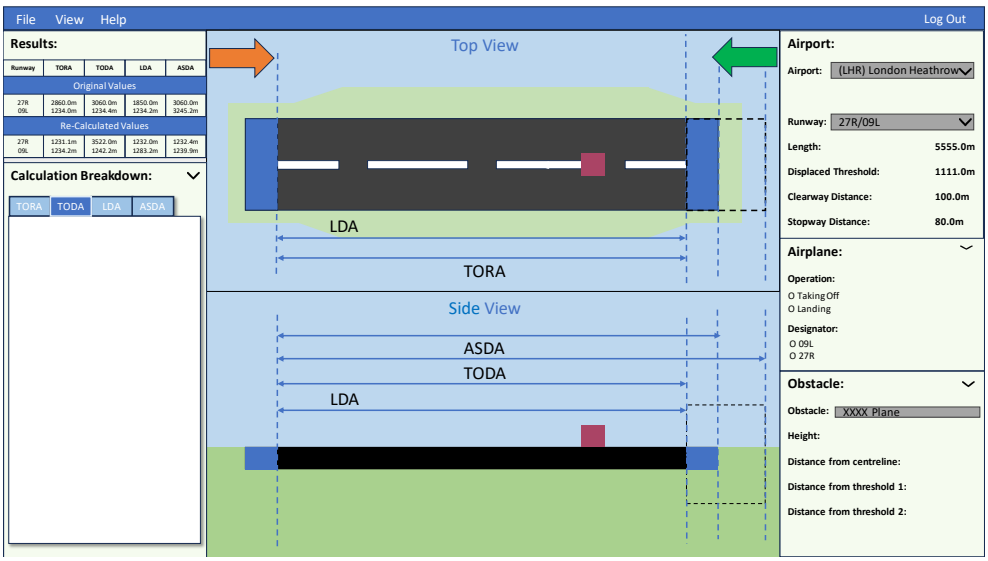
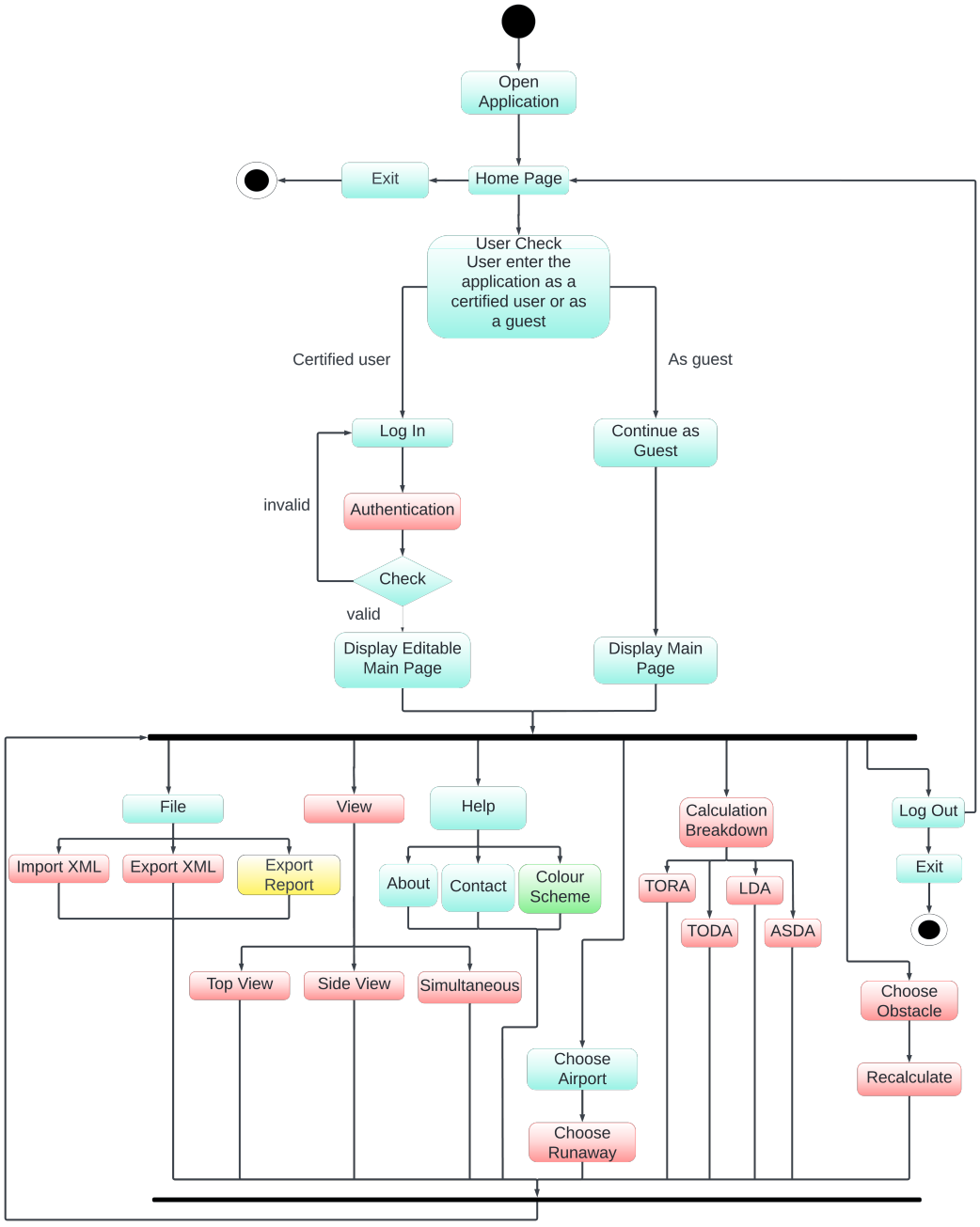


Figure 3: In the simultaneous scene, airplanes are not displayed to provide more space for the indicators.

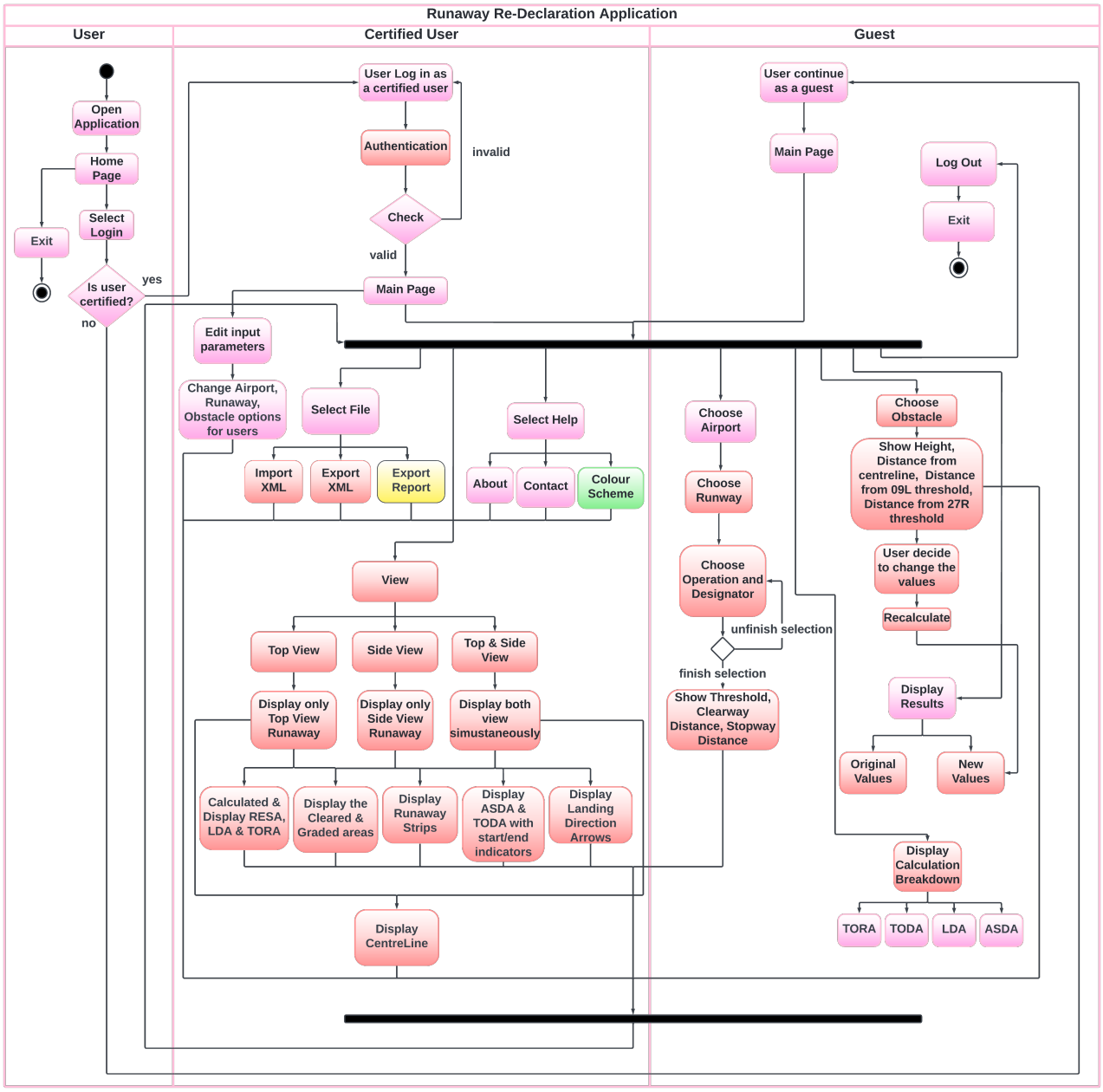
4 UML Diagrams

Unified Modeling Language (UML) serves as a comprehensive visual language utilized for specifying, constructing, and documenting various artifacts within systems. Within this section, we aim to elucidate the operational mechanisms of our applications through the application of three distinct UML design types, each offering unique perspectives and insights into the software's functionality and structure:

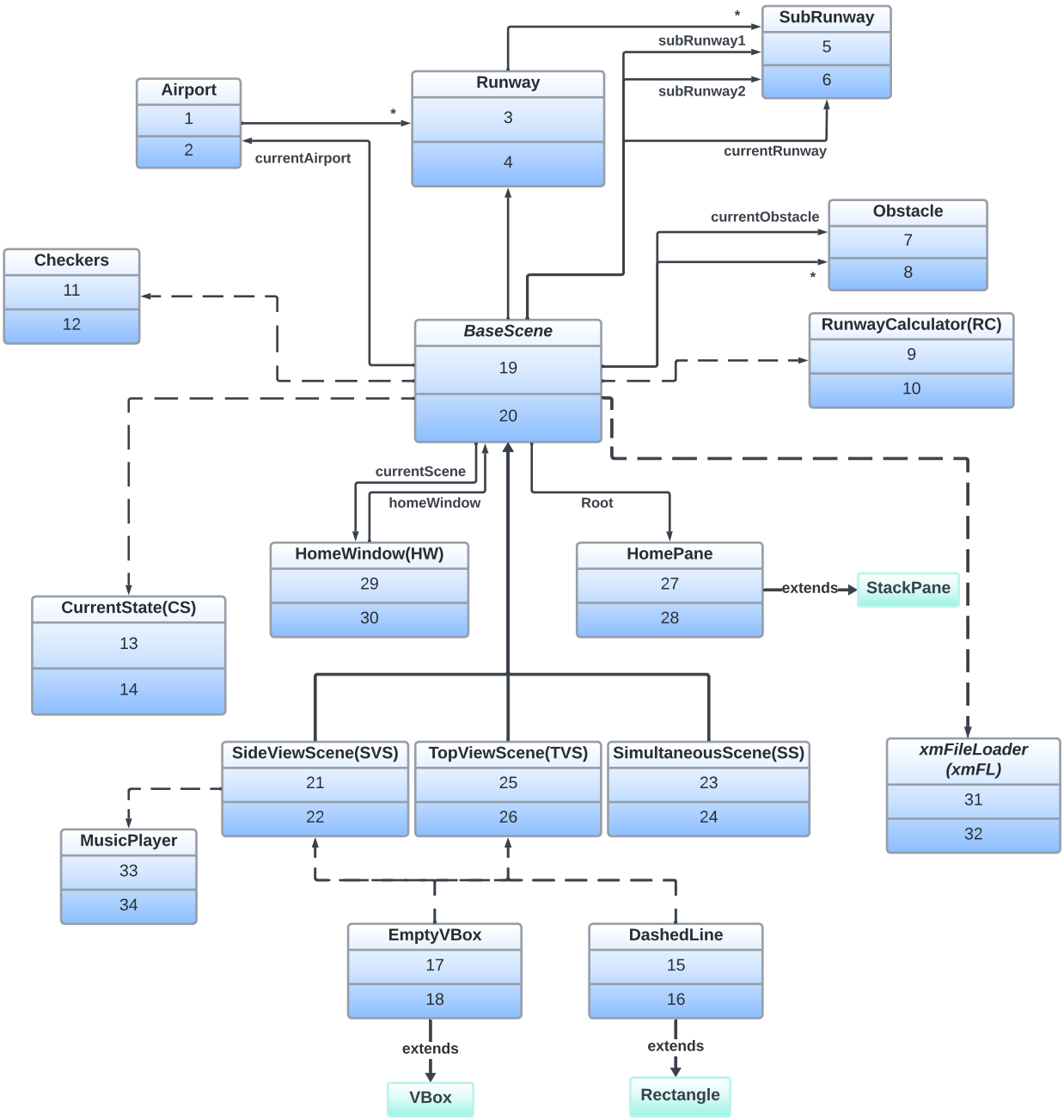
1. **State Diagram**: This graphical representation offers a profound insight into the operational dynamics of our applications, illustrating the intricate transitions between different states. By delineating the sequence of states and the transitions between them, stakeholders gain a comprehensive understanding of how the application functions under diverse conditions. State diagrams not only depict the current state of the application but also provide clarity on how it responds to various stimuli, thereby aiding in the identification of potential bottlenecks and optimization opportunities.



2. **Activity Diagram**: As an essential tool for depicting the flow of activities within the applications, activity diagrams provide a holistic view of the operational workflow. By visualizing the sequence of actions and decision points within the application, stakeholders can discern the logic governing its behavior. This graphical representation enables stakeholders to identify procedural inefficiencies, streamline workflows, and optimize resource utilization. Moreover, activity diagrams facilitate effective communication among development teams, business analysts, and stakeholders by providing a clear and intuitive depiction of the application's operational logic.



3. **Class Diagram**: Class diagrams serve as foundational structures for elucidating the architecture of our applications by modeling classes, relationships, attributes, and operations. By visually representing the structural elements of the application, including classes and their associations, class diagrams provide a blueprint for developers to comprehend and manipulate the underlying codebase effectively. Furthermore, class diagrams foster modularity and encapsulation, thereby promoting code reusability, maintainability, and scalability. Additionally, class diagrams serve as invaluable documentation assets, enabling stakeholders to grasp the application's architecture and facilitating seamless collaboration among development teams.



The following tables below provide a comprehensive breakdown of the content encapsulated within the Class Diagram, delineating each component's attributes and methods. These tables serve as a representation of the Class Diagram's components:

- **Class**: Shows the classes that we implemented in the source code to create the application.
- **ID**: Represents the id for the content in the Class Diagram for each section.
- **Content**: Shows the contents that should be in the Class Diagram which consists of variables and methods used such as mutator, accessor, property, setter and getter methods.

Class	ID	Content
Airport	1	- name: String - runways: ArrayList<Runway>
	2	+ Airport(name:String) + setName(name:String)):void + getName():String + addRunway(runway:Runway)):void + getRunways(): ArrayList<Runway>
Runway	3	- name: String - obstacle: Obstacle - subRunways: ArrayList<SubRunway<Obstacle,Double>>
	4	+ Runway(name:String) + addSubRunway(subRunway:SubRunway):void + getSubRunways():ArrayList<SubRunway> + getObstacle(): Obstacle + setName(name:String):void + getName():String
SubRunway	5	+ oldParameters: ArrayList<Double> + newParameters: ArrayList<Double> + obstacle: Obstacle + obstacleDistance:double
	6	+ SubRunway(designator:String, tora:double, toda:double, asda:double, lda:double, displacedThreshold:double, stripEndLength:double, blastProtection:double) + SubRunway(new_subRunway:SubRunway) + update(new_subRunway:SubRunway):void + getDesignator():SimpleStringProperty + getOriginalTORA():SimpleDoubleProperty + getOriginalTODA():SimpleDoubleProperty + getOriginalASDA():SimpleDoubleProperty + getOriginalLDA():SimpleDoubleProperty + getTORA():SimpleDoubleProperty + setTORA(tora:double):void + getTODA():SimpleDoubleProperty + setTODA(toda:double):void + getASDA():SimpleDoubleProperty + setASDA(asda:double):void + getLDA():SimpleDoubleProperty + setLDA(lda:double):void + getStripEndLength():SimpleDoubleProperty + getBlastProtection():SimpleDoubleProperty + getDisplacedThreshold():SimpleDoubleProperty + getRESA():SimpleDoubleProperty + getClearwayLength():SimpleDoubleProperty + getStopwayLength():SimpleDoubleProperty + setObstacle(obstacle:Obstacle, distance:double):void + getObstacle():Obstacle + getObstacleDistance(): double

Class	ID	Content
Obstacle	7	- name:String - height:double - width:double - length:double
	8	+ Obstacle(name:String) + Obstacle(name:String, height:double, width:double, length:double, position:double) + setName(name:String)):void + getName():String + setHeight(height:double)):void + getHeight():double + setWidth(width:double)):void + getWidth():double + setLength(length:double)):void + getLength():double + toString():String
RC	9	no variables used
	10	+ calculateTORA(subRunway:SubRunway, obstacle:Obstacle, distance:double):double + breakdownTORA(subRunway:SubRunway, obstacle:Obstacle, distance:double):String + calculateTODA(subRunway:SubRunway, obstacle:Obstacle, distance:double):double + breakdownTODA(subRunway:SubRunway, obstacle:Obstacle, distance:double):String + calculateASDA(subRunway:SubRunway, obstacle:Obstacle, distance:double):double + breakdownASDA(subRunway:SubRunway, obstacle:Obstacle, distance:double):String + calculateLDA(subRunway:SubRunway, obstacle:Obstacle, distance:double):double + breakdownLDA(subRunway:SubRunway, obstacle:Obstacle, distance:double):String
Checkers	11	- height:double - width:double - distance:double
	12	+ Checkers(height:double, width:double, distance:double) + obstacleChecker(height:double, width:double, distance1:double, distance2:double)
CS	13	- colourSetting:String + musicrSetting():String = ”Off”
	14	+ CurrentState() + getColourSetting():String + setColourSetting(colourSetting:String):void + getMusicSetting():String + setMusicSetting(musicSetting:String):void
DashedLine	15	no variables used
	16	+ DashedLine(width:double, height:double) + DashedLine(width:double, height:double, isDashed:boolean) + DashedLine(width:double, height:double, color:String)
EmptyVBox	17	no variables used
	18	+ EmptyVBox(height:double, width:double)

Class	ID	Content
BaseScene	19	# homeWindow:HomeWIndow # root:HomePane # scene:Scene # currentView:String # airportList:ArrayList<Airport> # currentAirport:Airport # currentRunway:Runway # subRunway1:SubRunway # subRunway2:SubRunway # currentSubRunway:SubRunway # currentObstacle:Obstacle # toraBox:HBox # todaBox:HBox # ldaBox:Hbox # asdaBox: HBox # blastAllowanceBox: HBox # resaBox: HBox # stripEndBox: HBox # toraBox2:HBox # todaBox2:HBox # ldaBox2:Hbox # asdaBox2: HBox # blastAllowanceBox2: HBox # resaBox2: HBox # stripEndBox2: HBox + obstacleHeightD:double + obstacleWitdthD:double + distanceD:double + operationSelected:String = null - clearwayLengthDisplay:Text - stopwayLengthDisplay:Text - thresholdLengthDisplay:Text # firstDirectionButton:RadioButton # secondDirectionButton:RadioButton - buttonTORA:Button - buttonTODA:Button - buttonASDA:Button - buttonLDA:Button - allButtons:Button[] - displayArea:TextArea

Class	ID	Content
BaseScene	20	+ BaseScene(homeWindow:HomeWindow) + *initialise():void* + *build():void* + makeResultsTPane():TitledPane + makeCalcBreakTPane:TitledPane - updateButtonStyles(selectedButton:Button, allButtons:Button[], displayArea:TextArea):void + makeAirportTPane():TitledPane + makeObstacleTPane():TitledPane + makeAirplaneTPane():TitledPane + clearAllButtons():void + setScene():Scene + getScene():Scene + changeBaseSceneColours():void
SVS	21	- left_box:VBox - right_box:VBox - menuBox:HBox - middleDisplayBox:StackPane - displayStakePaneTop:StackPane
	22	+ SideViewScene(homeWindow:HomeWindow) + initialise():void + build():void + makeDirectionPane():BoarderPane - makeMenuBox():HBox + makeDisplacedThreshold():StackPane + makeIndicators1():StakePane + makeIndicators2():StakePane + makeSideViewMiddleDisplayBox():StakePane + makeTopViewMiddleDisplayBox():StakePane + makeSimultaneousMiddleDisplayBox():StakePane + changeColourScheme():void + makeMusicSettingPage():void + makeColourSettingPage():void + changeColourSchemeTop():void + changeColourSchemeTopSim():void
SS	23	- left_box:VBox - right_box:VBox - menuBox:HBox - middleDisplayBox:StackPane
	24	+ SimultaneousScene(homeWindow:HomeWindow) + initialise():void + build():void - makeMenuBox():HBox + makeMiddleDisplayBox():StackPane
TVS	25	- left_box:VBox - right_box:VBox - menuBox:HBox - middleDisplayBox:StackPane

Class	ID	Content
TVS	26	+ TopViewScene(homeWindow:HomeWindow) + initialise():void + build():void - makeMenuBox():HBox + makeMiddleDisplayBox():StackPane
HomePane	27	- width: int - height: int - scalar: double = 1 - autoScale: boolean = true
	28	+ HomePane(width:int, height:int) # setScalar(scalar:double):void + layoutChildren():void
HW	29	- width: int - height: int - stage: Stage - currentScene: BaseScene - scene: Scene
	30	+ HomeWindow(stage:Stage, width:int, height:int) + startHome():void + startSideView():void + startTopView():void + startSimultaneousView():void + setupResources():void + setupStage():void + loadScene(newScene:BaseScene):void + setupDefaultScene():void + getScene():Scene + getWidth():int + getHeight():int
xmFL	31	no variables used
	32	+ loadAirports():ArrayList<Airport>
MusicPlayer	33	- mediaPlayer:MediaPlayer
	34	+ MusicPlayer(filePath:String) + play():void + stop():void + loop():void + pause():void + isPlaying():boolean

4.1 Scenarios

There is no change made to the scenarios, they are put here to support understanding the key design artifacts and also used as test cases for the scenario testings.

4.1.1 Qualified Calculator of Air Traffic Control, Ava

Ava is an experienced qualified calculator for the air traffic control. She wants to input the parameters of the runway for the tool to complete the calculations and obstacles

- Ava opens the runway re-declaration tool.
- The home page of the tool is shown.
- Ava logs into the tool with her credentials.
- The tool shows the side viewing page first, with her airport being remembered by the system.
  - If this is the first time logging in, she will have to manually select the airport that she wishes to view.
- Ava then has to select the specific runway she is calculating parameters for, this is done from a drop down selection.
- She wants to add an obstacle, so she selects the obsticle, inputs the height, width, distance from plane and centre-line.
  - These values inputted should be in metres.
  - If an invalid input (such as a negative number, letter or symbol) is inputted, an error message will come up telling the user to put in a numerical value instead.
- Ava then selects the "Recalculate" button and an overview of the calculations will be shown in the left side of the screen.
- Ava checks the recalculated values.
- Once he completes his work, Ava presses "File" on the tool bar, then "Exit".
- The application now closes.

4.1.2 Qualified Calculator of Air Traffic Control, Ava

Ava is a qualifier calculator and would like to import an XML file containing airports and obstacles.

- Ava opens the runway re-declaration tool, then logs in after the home page is shown.
- The tool shows the side on viewing page first.
- Ava navigates to the tool bar and presses "File".
- She then presses "Import XML file".
- This then shows a pop up prompting her to enter an XML file.
- She selects he XML file that she has saved and presses enter.
  - If the file she enters isn't a valid XML file, she will be told that it isn't the right format and be prompted to re-enter a file.
- She will be returned to the original file with her changes now being added.
  - If the XML file contains airports, the additional airports will now be shown in the selection box.
  - If the XML file contains obstacles, these will also now be shown.
- Ava then continues with her work with these new additions.

4.1.3 Junior Qualified Calculator Air Traffic Controller (Trainee), Alex

Alex is a junior calculator (ATC), and needs to reread the help document as she has encountered a new problem she has not encountered yet.

- Alex opens the runway re-declaration tool.
- The home page of the tool is shown.
- Alex logs into the tool with her credentials.
- The tool shows the side on viewing page first.
- Lisa navigates to the tool bar and pressed on "Help".
- This then opens a page containing a help document.
- Alex then scans through the help document, finding the solution to the specific problem he encountered.
- Alex exits from the help document and he continues his work now with the some of the gaps in his knowledge now filled.
- Once he completes his work, Alex presses "File" on the tool bar, then "Exit".
- The application now closes.

4.1.4 Regulatory Authorities(CAA), Lisa

Lisa is part of the Regulartory Authorities and would like to edit the input parameters.

- Lisa opens up the runway re-declaration tool.
- Lisa logs into the tool with her credentials, these credentials have certain permissions that other users do not have.
- Lisa is shown the side on viewing page.
- Lisa presses on "File", then a drop down menu appears.
- This menu will include "Revise runway parameters" (this is a unique options to only people that have permissions on their account), Lisa pressed this.
- A screen with all of the aspects of the parameters that are provided by the CAA, with this user now being able to edit these.
- Lisa edits the parameters that are required and presses "Finish and Save".
- A pop up comes up with all of the changes she's made, asking her "Are these the correct changes?".
- Lisa presses "Yes", and is returned to the first page, the simultaneous viewing page.
- Lisa then presses "File", then presses "Exit".
- The application now closes.

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

Maria is a Qualified Calculator (ATC) and is changing the colours the the tools to suit her disability.

- Maria opens the runway re-declaration tool
- The home page of the tool is shown
- Maria logs into the tool
- The tool shows the side on viewing page first.
- Maria navigates to the upper toolbar and presses the help button
- Maria clicks on accessibility
- A colour blind drop down menu appears with options for different types of colour blindness (Protanopia, Deuteranopia, Tritanopia, Monochrome)
- Maria clicks on Tritanopia
- Maria is asked if she would like to set this as her default setting
- Maria selects "Yes, set as default"
- The colour palette of the tool changes to better suit her needs
- This becomes her default setting for next time she logs onto the application

4.1.6 Pilot, James

James is a pilot, circling around the airport in the sky and is wanting to know whether he can land.

- James contacts ground control to ask whether he can land the plane
- Ground control informs ATC of James' intention to land the plan
- Ground control uses James's flight parameters, such as current weather conditions, runway length, wind speed, and aircraft weight.
- ATC checks with the tool:
  - ATC gathers the input parameters from their interfaces
  - ATC inputs necessary parameters for this specific pilots landing
  - The outputted calculations are returned to the ATC
  - ATC reviews the outputted calculations and uses it as an aid to decide whether performing the calculations with the official landing process is worthwhile
  - If the calculated parameters indicate safe landing conditions, ATC informs James that he is clear to land.
  - If the conditions are deemed unsafe, ATC may advise James to continue circling or divert to an alternate airport.
- James receives the ATCs information and takes it onboard while preparing to land
- James begins descending and lands the plane safely and smoothly

4.1.7 Qualified Calculator of Air Traffic Control, Ava

Ava would like to change her dashboard to view both the top down and side on view simultaneously

- Ava opens the runway re-declaration tool.
- The home page of the tool is shown.
- Ava logs into the tool with her credentials.
- The tool shows the side on viewing page first.
- Ava navigates to the top toolbar and clicks on the "view" button
- This opens a dropdown menu with three options "top", "side", "top & side"
- Ava presses "top & side"
- This changes the view of her software to display both the top down view and the side on the view simultaneously

5 Testing

For our application, we have included various error messages and error-handling methods. Different types of testing below describe our methods to ensure error messages are shown when necessary. When an error message for inputting values shows the user is told what is wrong and then can exit off of the message and return to the main program. Once they return to the program, the value or values that were incorrect have their text field cleared so that the user can re-enter valid values. The tests will be explained in more details in this section. More specifically, this section will reflect the software's error handling when invalid values are inputted.

5.1 Boundary Testing

5.1.1 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.

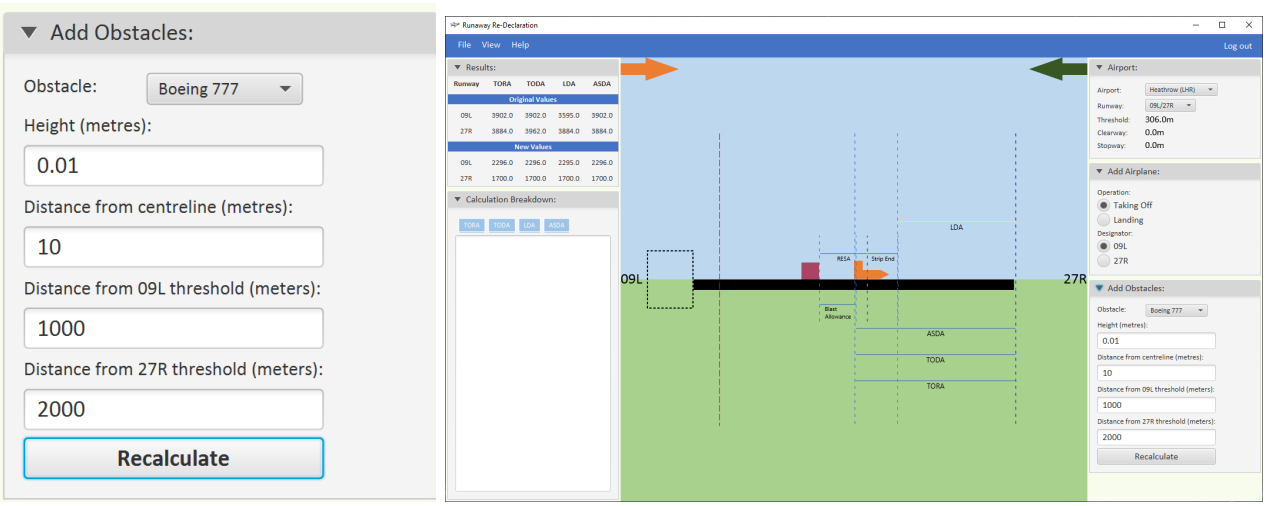


Figure 4: 0.01 meters is accepted for boundary testing of obstacle height

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

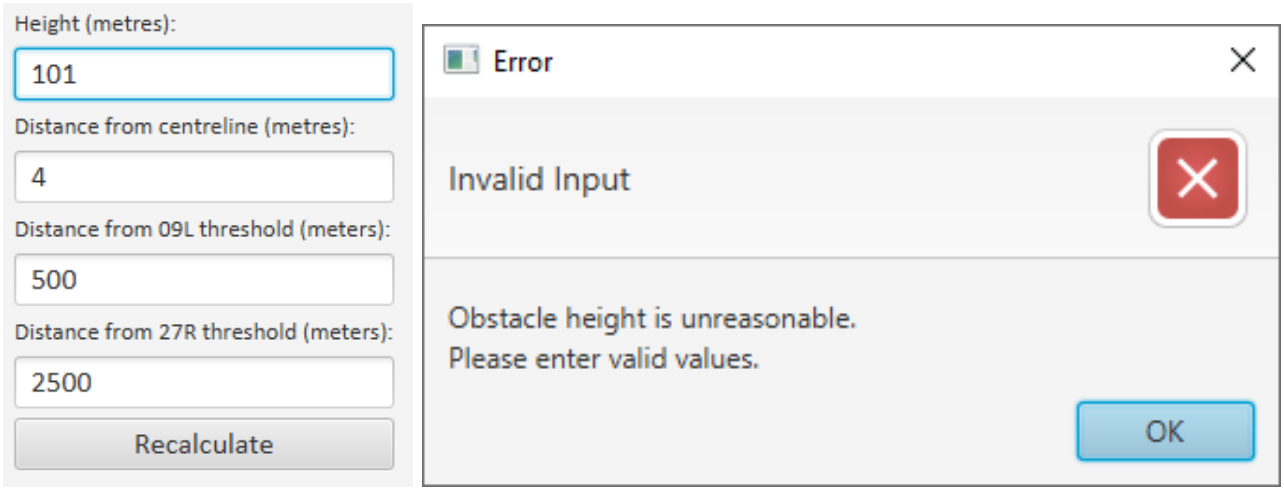


Figure 5: 101 meters is rejected for boundary testing of obstacle height

5.1.2 Distance from centreline

The inputted distance from centreline can be, at most, 75m away. An obstacle located 75m away from the centreline doesn't require recalculations of the indicator values. This means that a value of 75.1m or -75.1m shouldn't be accepted and shouldn't be shown on the display, and also a prompt should come up letting the user know why this is invalid.

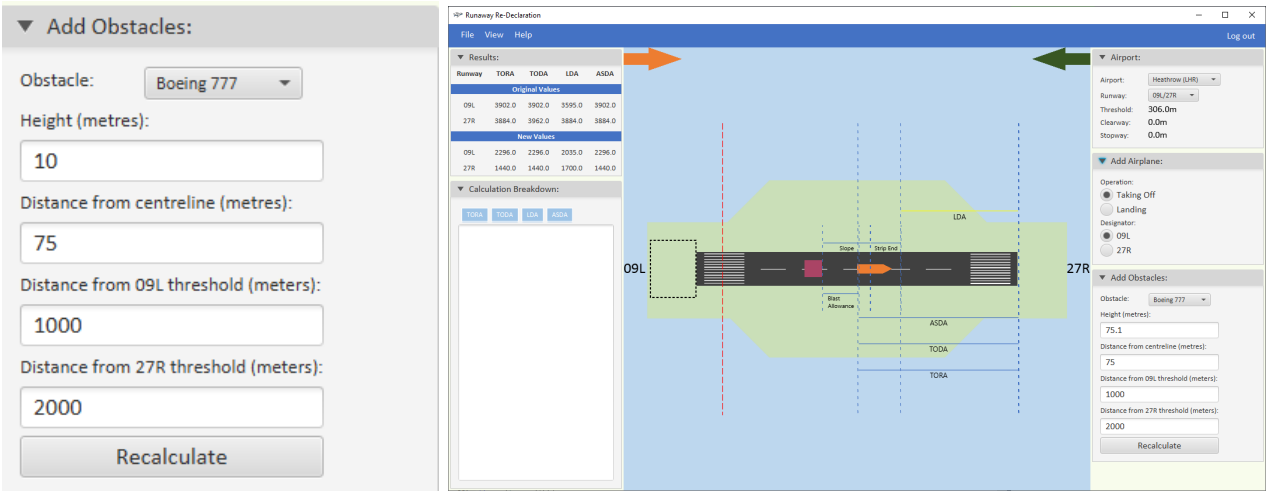


Figure 6: 75.1 meters is inputted for boundary testing of distance from centreline

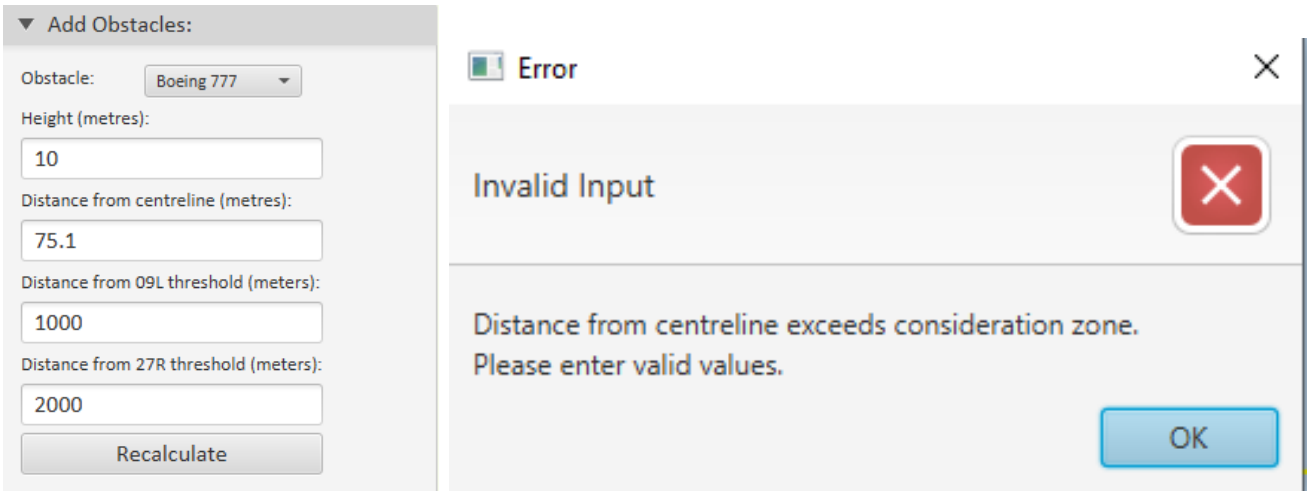


Figure 7: 75 meters is inputted for boundary testing of distance from centreline

5.1.3 Distances from threshold

If one obstacle is located more than 60 meters away from the strip end, then recalculation isn't required. Therefore, if a user inputs 60.1 meters or -60.1 meters for the distance from threshold (assuming the current runway doesn't have a displaced threshold) our program rejects such input and prompts an error message.

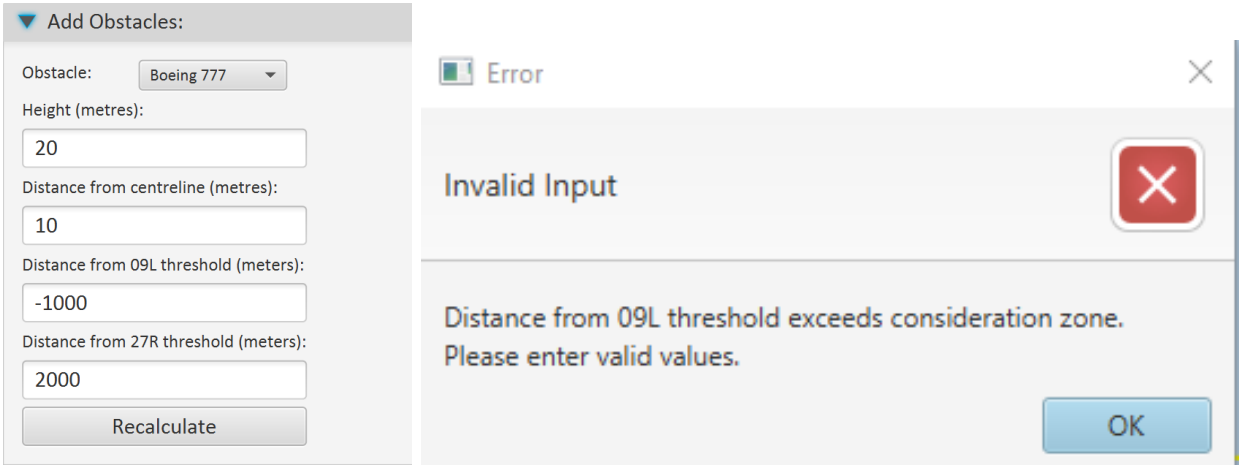


Figure 8: -1000m is used for boundary testing of distance from beginning of threshold where the threshold is 939 metres in such that it is over 60m before the threshold

5.2 Extreme Value Testing

5.2.1 Obstacle Height

Our program can handle extreme 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.

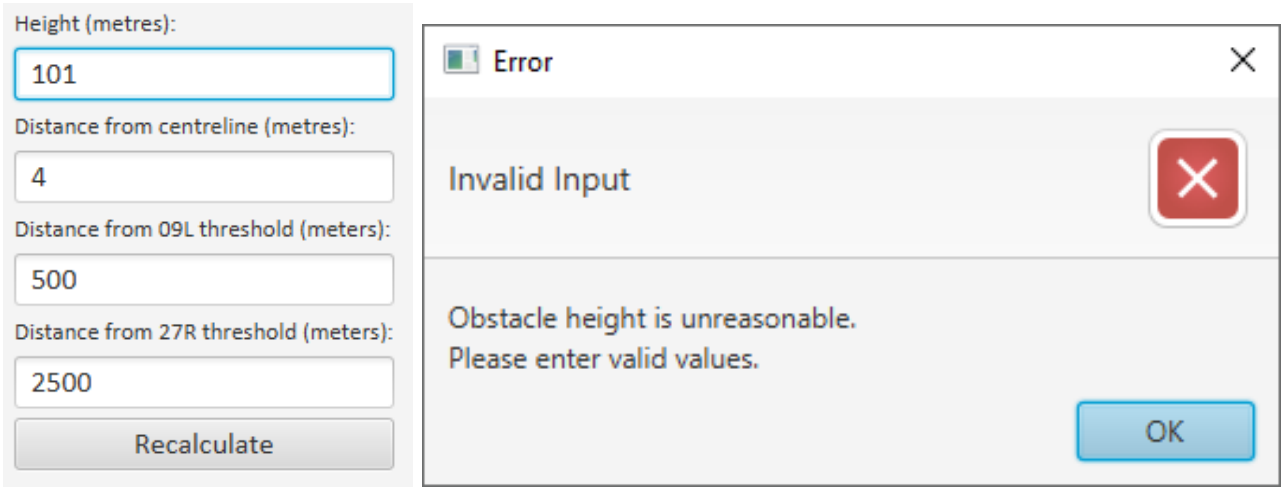


Figure 9: 1000 meters is rejected for boundary testing of obstacle height

5.2.2 Distance from centreline

As explained in the boundary testing, the distance from centreline shouldn't exceed 75 meters otherwise the obstacle doesn't worth recalculation.

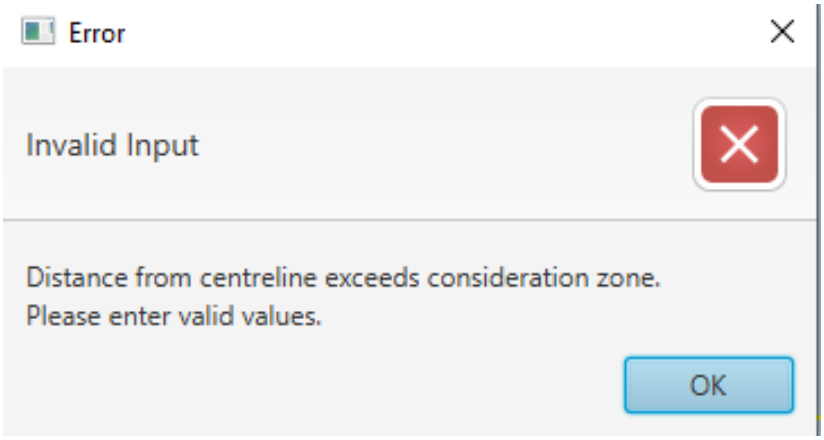


Figure 10: 1000 meters is inputted for boundary testing of distance from centreline

5.2.3 Distances from threshold

As explained in the boundary testing, the distance away from strip end shouldn't exceed 60 meters otherwise the obstacle doesn't worth recalculation. Furthermore, we believe if some inputs are unrealistic for the calculation then the software should reject them.

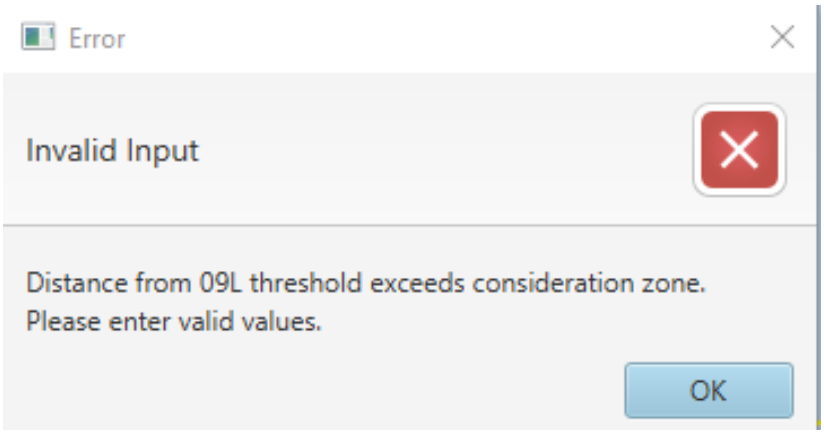


Figure 11: -10000 is used for the threshold and the error is correctly shown

5.3 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.

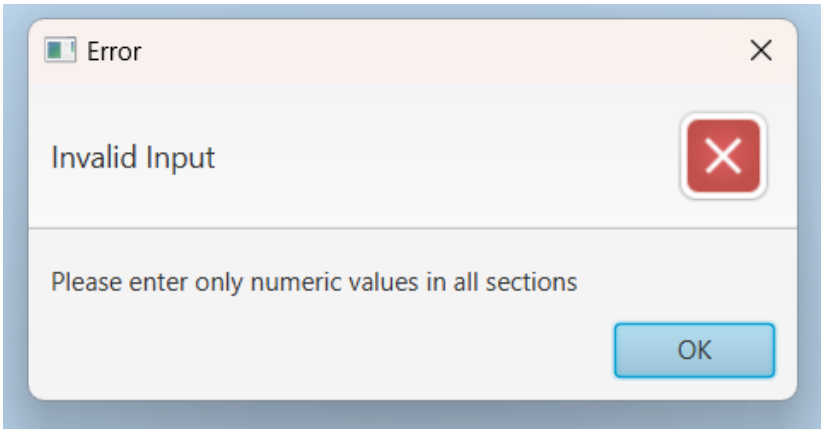


Figure 12: An invalid input type will provide an error

5.4 Unit Testing

Our program can handle correct values and output accurate results. This is evident in all of the Junit tests that we created. The parameters and results from the provided example calculations are used as the answer to be compared in the Junit testing files.

Figure 13: JUnit tests designed to test given calculation examples

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.

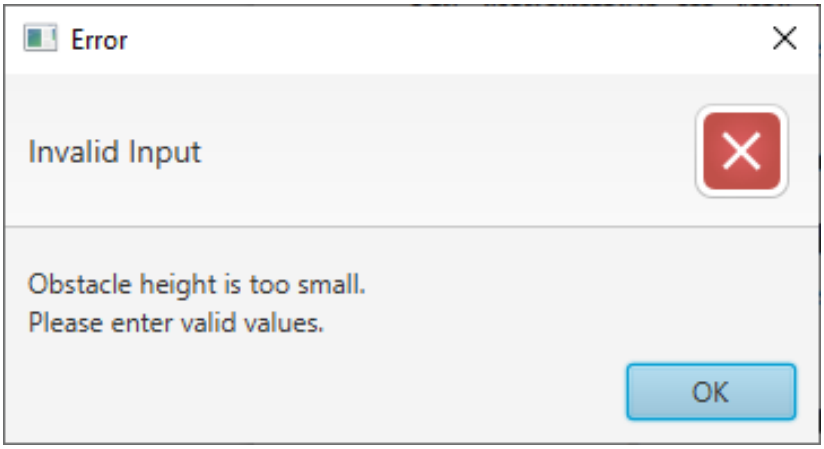


Figure 14: Result when an input of 0 is given for the height of the obstacle

5.5.2 Distance from centreline

As explained in the boundary testing for distance of obstacle from centreline, the distance from centreline will accept 0 meter in which it is still in the range of -75m and 75m.

5.5.3 Distance from threshold

As explained in the boundary testing for distance of obstacle from threshold, the distance from centreline will accept 0 meter in which it is still in the range of -60m and 60m.

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 textfield of the height, ready for the user to input valid inputs.

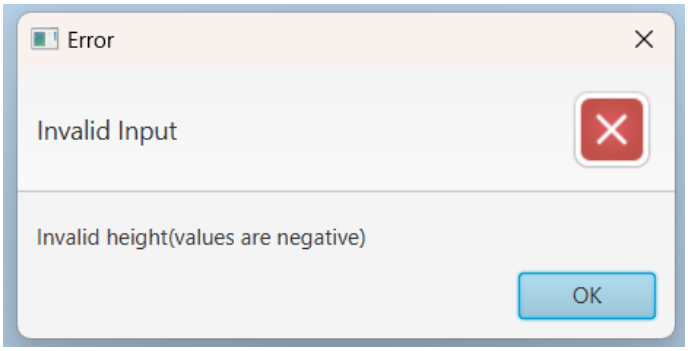


Figure 15: Negative number is inputted for the obstacle height.

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 behavior in areas that were previously functioning correctly. Regression testing typically involves running existing test cases against the updated version of the 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

5.8.1 Qualified Calculator of Air Traffic Control, Ava

This scenario describes a qualified calculator of the air traffic control undergoing standard calculations. An example of these calculations are evident in the tests and all aspects of our code follows the aspects of the scenario, apart from exitting the application where this part is to be completed in the next increment.

5.8.2 Qualified Calculator of Air Traffic Control, Ava

This scenario is describing importing and exporting XML files which is to be completed in the next increment so this testing isn't applicable.

5.8.3 Regulatory Authorities(CAA), Lisa

This scenario is describing logging in with specialised credentials and changing certain aspects which is to be completed in the next increment so this testing isn't applicable.

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

This scenario describes a user that wants to change the colour scheme on the application to accommodate for her colour blindness. Our application passes this text except for the saving preference aspect as this part is to be completed in the next increment. 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 colourblindess 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.

6 Sprint 2 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 ahead of essential ones due to their direct relevance to the tasks within their respective increments. This incremental plan, derived summarily from user stories, serves primarily as a consultative guide. It will be further refined and modified in subsequent phases. All user stories are classified to be **Must** tasks and all design artifact tasks are classified to be **Should** tasks.

Story ID	Task	Workforce	Estimated	Actual
5	Original & New Values Display	Eric, Sam H	7	5
6	Obstacle	Eric, Sam H	4	4
9	Recalculation	Eric, AJ	6	6
16	Calculation Breakdown	Eric, AJ	5	4
18	Threshold Indicators	Eric, Sam H	8	8
24	Displace Thresholds	Eric, Muhammad	8	7
25	Offset	Muhammad, Sam H	8	8
13	Alternative Colour Scheme	Eric, Muhammad, Sam H	4	4
28	Printing	AJ, Sam H	6	5
Design	UML	Muhammad	4	4
Design	Tests	Muhammad, Sam H	5	5
Report	Report	All	4	4
Report	Burndown Chart	All	2	2
Report	Report Sprint	All	1	1
N/A	**Total**	**All**	**72**	**67**

6.1 Description of the implementation of each story ID

6.1.1 Original New Values Displayed

These values are displayed in the top left corner of our program.

6.1.2 Obstacle

There are select pre-defined obstacles already present that the user can select from and also edit. When inputting the obstacle the user also has to input the distance from the centre line as well as the distance from either threshold. Once all of these are defined and after the user presses recalculate, the obstacle will appear on the centre display.

6.1.3 Recalculation

6.1.4 Calculation Breakdown

The breakdown is shown in the bottom left corner of the display. When each of the four options of the calculations are pressed the box will show the breakdown and only display the relevant markers on the screen.

6.1.5 Threshold Indicators

The threshold indicators (TORA, TODA, ASDA and LDA) are represented by blue dashed lines align the runway. The starting position and end position is adjusted to the correct relative position according to the results of the recalculation. Safety distances (RESA, strip end and blast protection) are also displayed if they are involved in the re-calculation.

6.1.6 Displaced Thresholds

The displaced thresholds are represented by red dashed lines if they they are defined in the runway. The distance from the threshold to the end of the runway is displayed in the top of the right side box after the user clicks which designator is currently focused.

6.1.7 Alternative Colour Schemes

Changing the colour schemes can be done in the help menu section at the top. The colour schemes were specifically chosen based on some of the most common colour blindness', however, because many other colour blindness are similar to one another we decided to change it to just describing the actual colour scheme instead. The following colour schemes are available:

- Default (Blue/Green)
- Blue/Yellow: this is accommodating for Deuteranopia.
- Magenta/Lime Green: this is accommodating for Tritanopia.
- Cyan/Deep Purple: this is accommodating for Protanopia.

6.1.8 Printing

Functionality was added for export report under the file menu. This ensures that the report can be immediately exported to to the desired location using the file explorer. First, it checks for any errors and recalculates with current values inside of the boxes. Then if there are no issues a pop up appears asking the user where they would like to export to and allows them to rename and save it.

6.1.9 Additional Task: Music

We managed to finished all planned tasks earlier so we started additional functionalities such as music.
Music can be located under the help section of the menu. When you press on it you are prompted whether or not you'd like to play music or not. At the moment the music being played is just as a placeholder as we aim to properly finish his off next increment.

7 Sprint 2 Burndown Chart

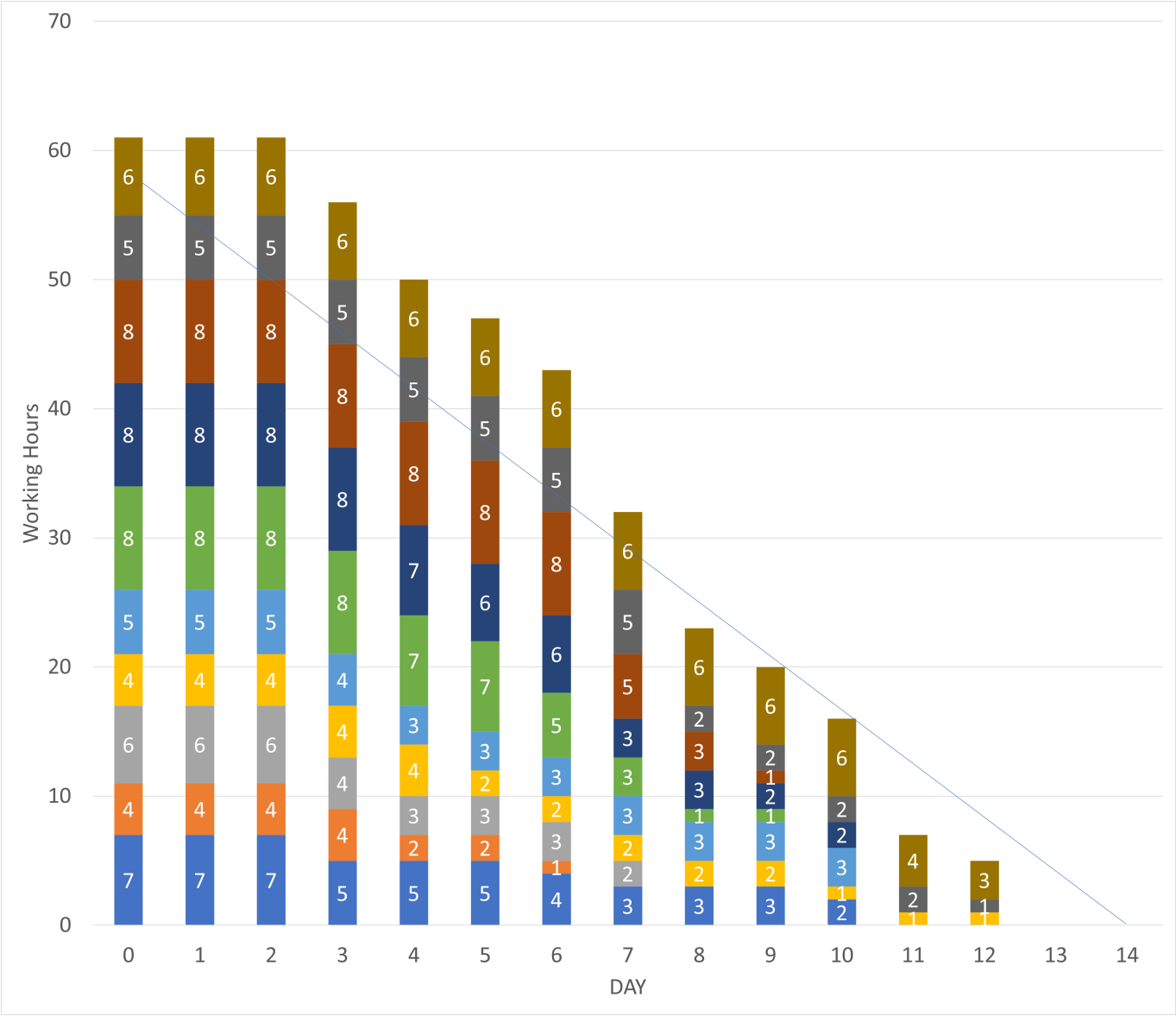


Figure 16: Burndown chart for sprint 2

This is the burndown chart for all the planed critical user stories in this sprint. All tasks are finished on time.

8 Sprint 3 Sprint Plan

This is the sprint plane for the next sprint. The tasks are coloured to follow the MoSCoW prioritisation:

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

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

9 Sprint 3 Burndown Chart

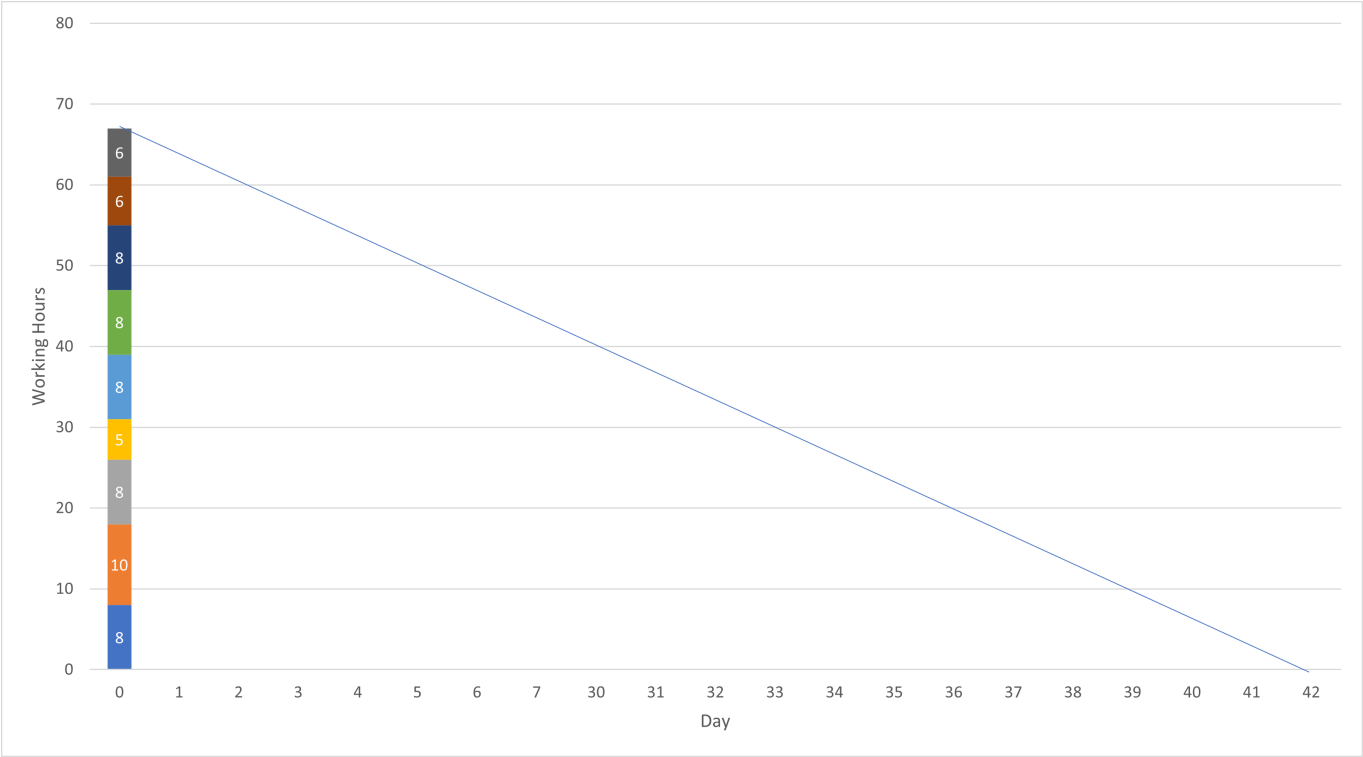


Figure 17: Sprint 3 initial burndown chart

This is the burndown chart for the next increment. The period of time between day 8 and day 29 is hidden due to the Easter break and the burndown chart will be continued after day 29.