University of Southampton

COMP2211: Software Engineering Group Project

Deliverable 3:
Increment 2

Group 30: Benjamin Lellouch (brl1u18), Daniel Best (db5n17),
Deniz Matur (dm1n17), Iman Dzulhadi (ids1g18) Rajpal Dhillon (rsd1n18)

September 28, 2020

Contents

1 Design

1.1 Artifacts

In our last sprint review, our supervisors have suggested that we refactor our controller class as it was getting lengthy and was managing multiple different views at the same time, making the code less maintainable. We have therefore refactored the Controller by splitting it up in multiple, more readable and maintainable controller classes. This change in design has been reflected in the class and sequence diagram below.

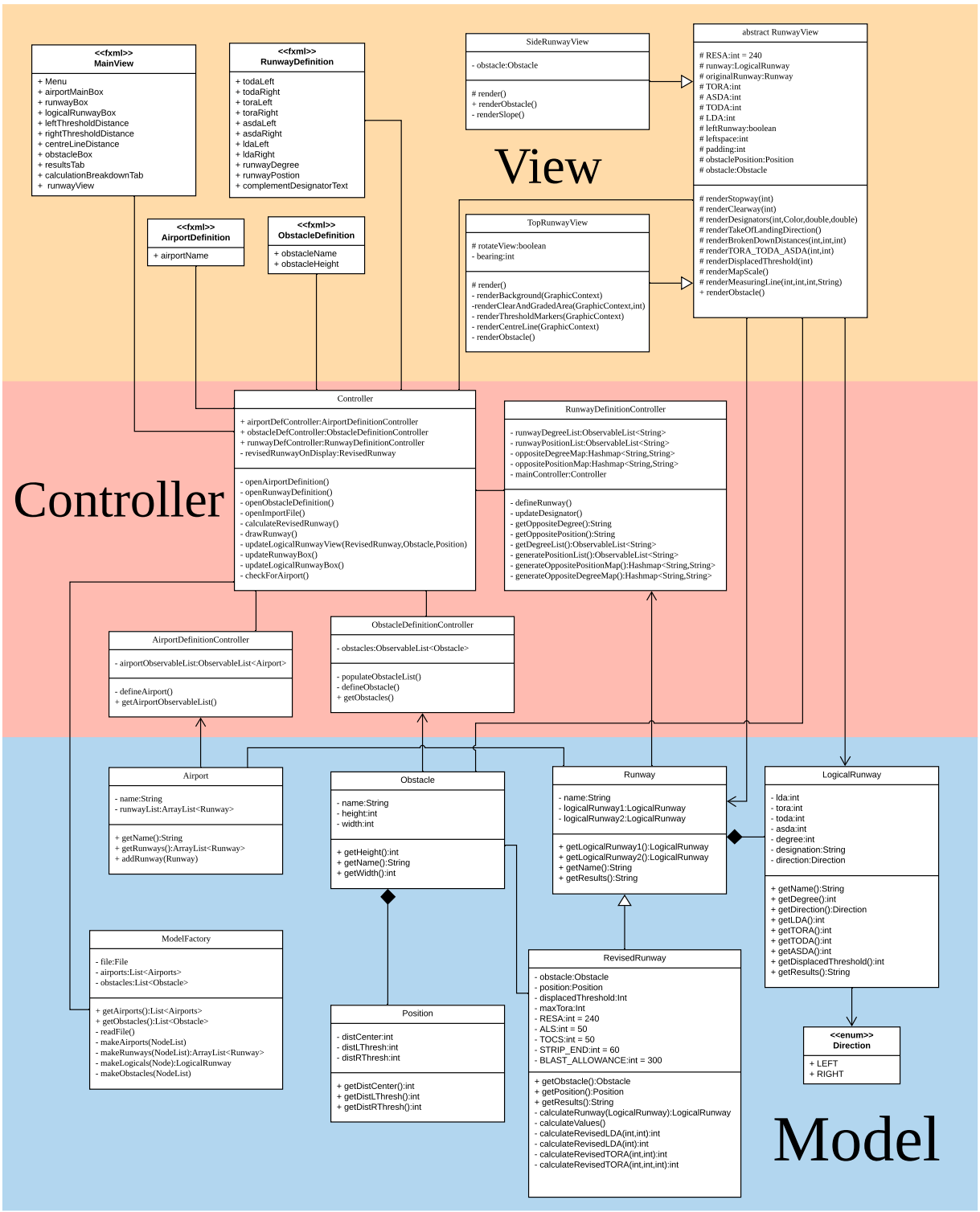


Figure 1: Class Diagram

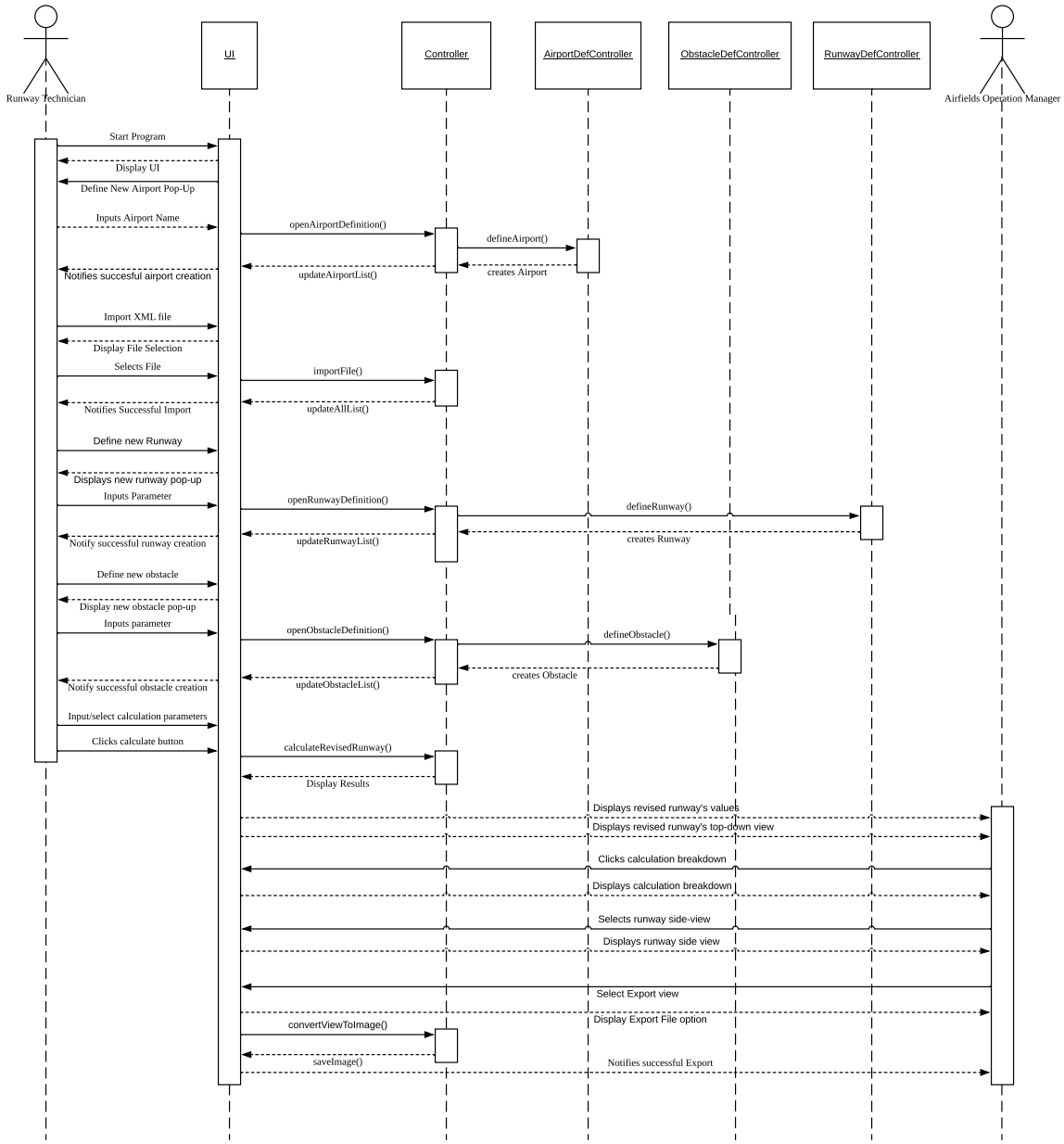
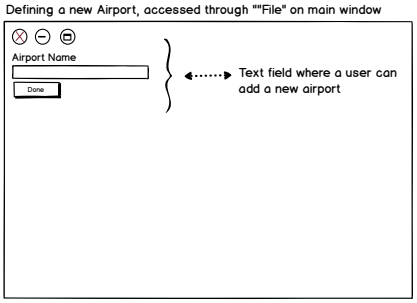
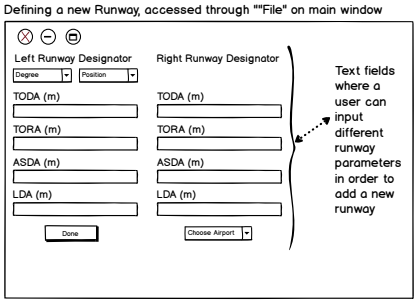


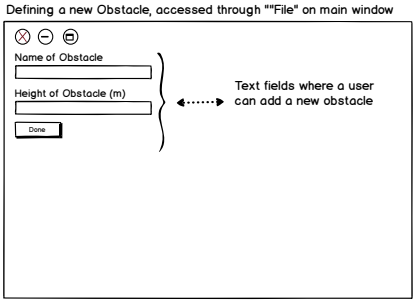
Figure 2: Sequence Diagram

1.2 Storyboards

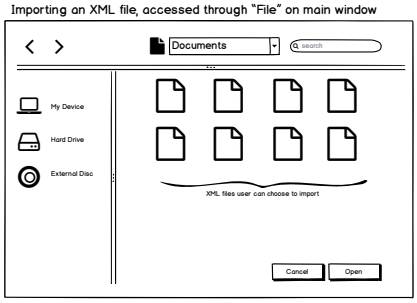
We have also revised our storyboards so that they reflect our current UI more accurately.



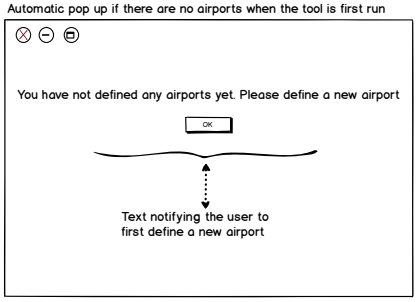




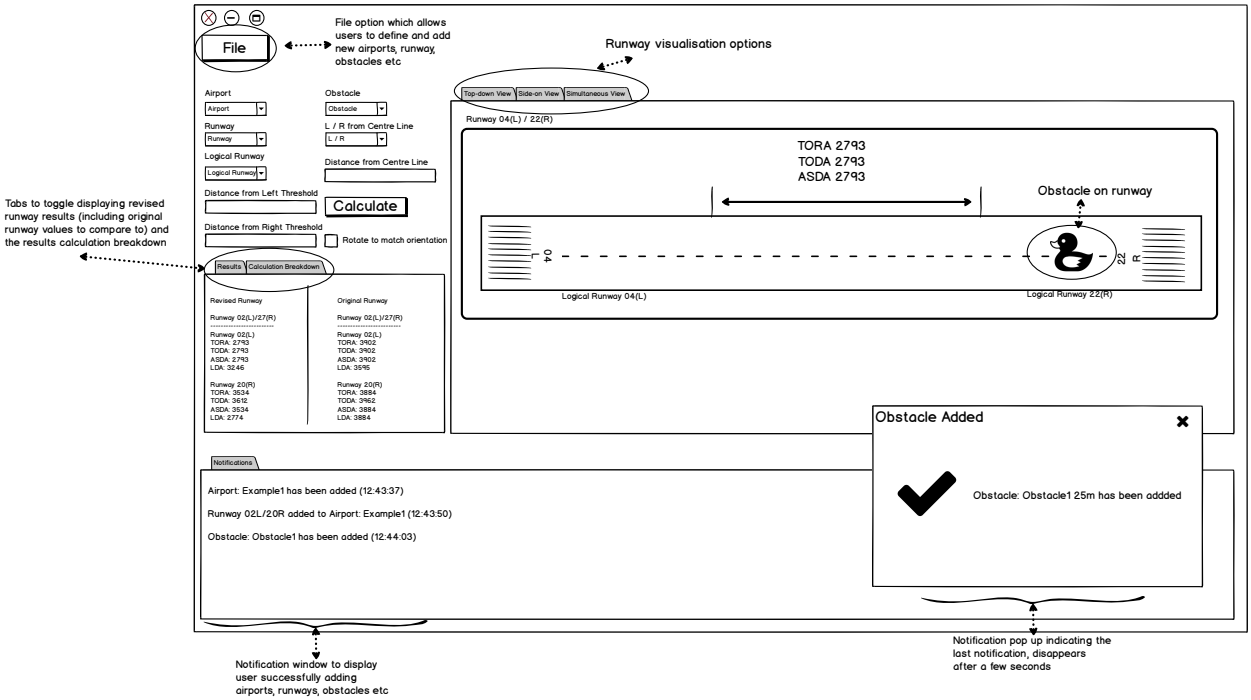
Importing an XML file, accessed through "File" on main window



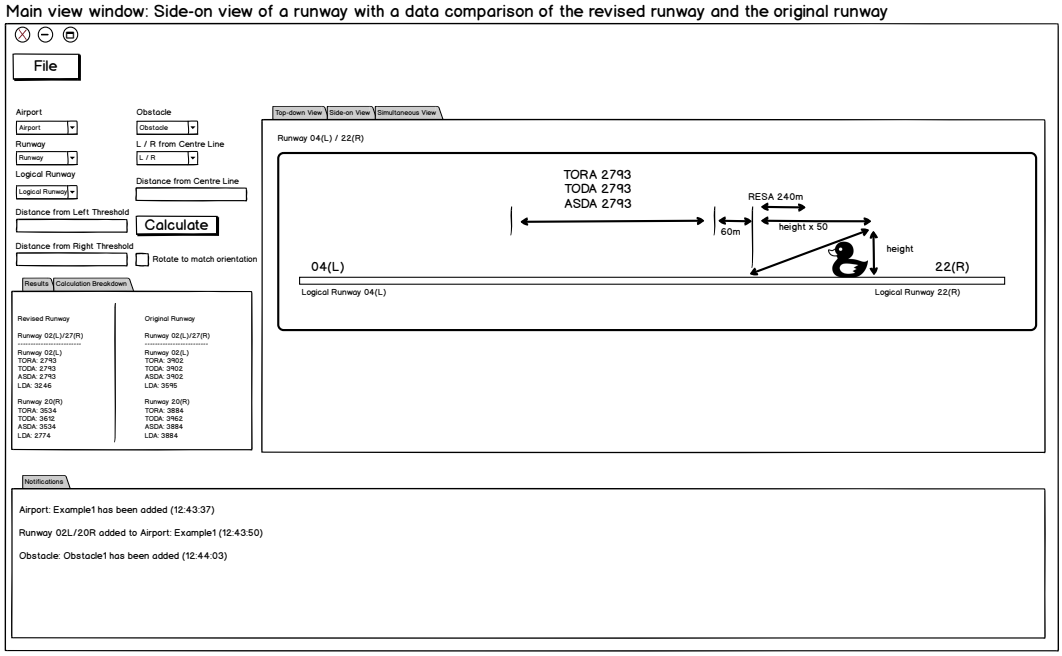
Automatic pop up if there are no airports when the tool is first run



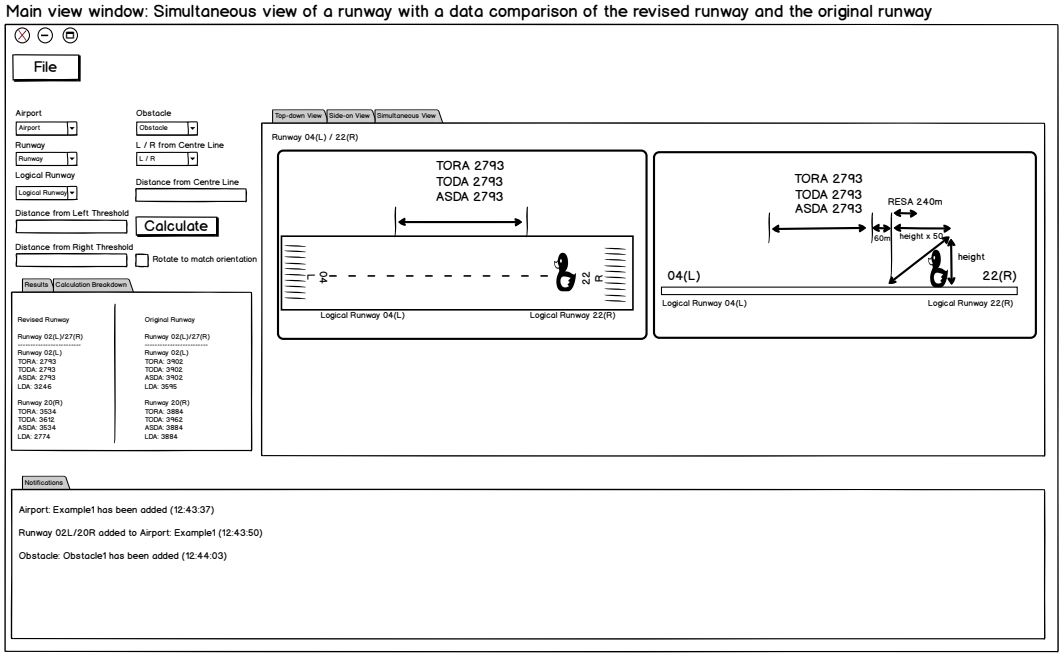
Main view window: Top-down view of a runway with a notification pop up and a data comparison of the revised runway and the original runway



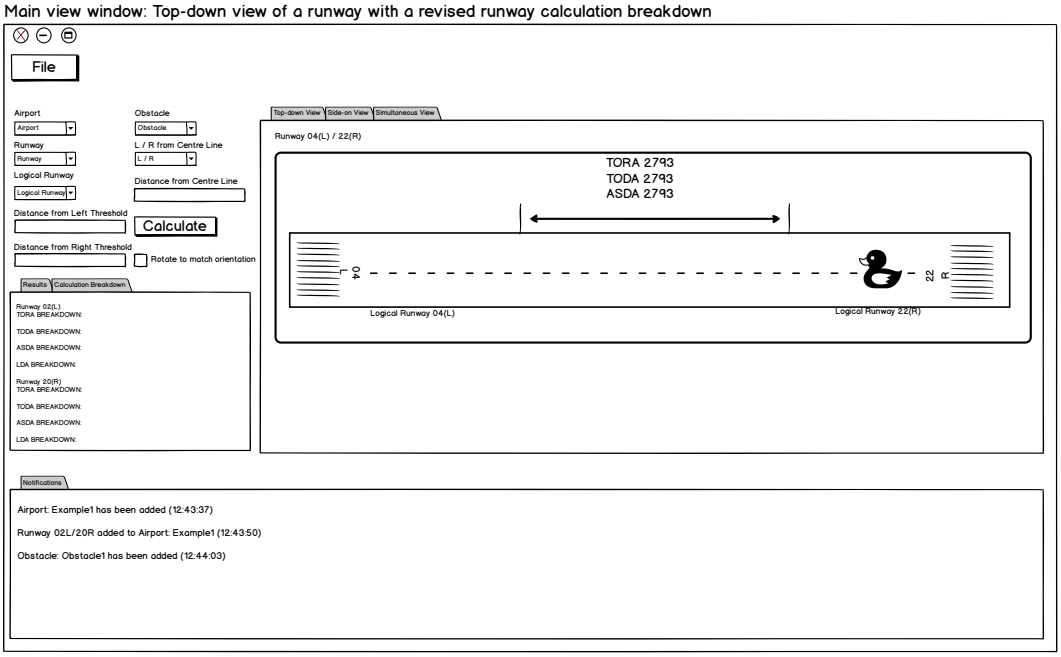
Main view window: Side-on view of a runway with a data comparison of the revised runway and the original runway



Main view window: Simultaneous view of a runway with a data comparison of the revised runway and the original runway



Main view window: Top-down view of a runway with a revised runway calculation breakdown



1.3 Architecture

As suggested by our supervisors at the last sprint review, we have refactored our Controller to be a collection of smaller and more maintainable classes. First, we have our main controller which handles interactions with the main window and we then have secondary controllers which control the creation and storage of airports, runways and obstacles. These controllers follow the singleton design pattern as we want to make sure that only one instance of each controller is present at any single time.

2 Testing

2.1 Unit Testing and Scenario Testing

We have a total of 25 tests which cover different test cases and scenarios. For example, we have tests that focus solely on the calculations of the revised runway based on the given calculation examples.

```
@Test
public void testScenario1()
{
    Obstacle obstacle = new Obstacle( name: "test",  height: 12, width: 1);
    Position position = new Position( distCenter: 0, distLThresh: -50, distRThresh: 3646);
    RevisedRunway revisedRunway = new RevisedRunway(runway09L27R, obstacle,position);

    assertEquals( expected: 3346,revisedRunway.getLogicalRunway1().getTora());
    assertEquals( expected: 3346,revisedRunway.getLogicalRunway1().getToda());
    assertEquals( expected: 3346,revisedRunway.getLogicalRunway1().getAsda());
    assertEquals( expected: 2986,revisedRunway.getLogicalRunway1().getLda());

    assertEquals( expected: 2986,revisedRunway.getLogicalRunway2().getTora());
    assertEquals( expected: 2986,revisedRunway.getLogicalRunway2().getToda());
    assertEquals( expected: 2986,revisedRunway.getLogicalRunway2().getAsda());
    assertEquals( expected: 3346,revisedRunway.getLogicalRunway2().getLda());

}
```

Figure 3: Calculation test for scenario 1

We also have designed tests which checks that erroneous input from the user is handled properly.

```
@Test
public void fail_emptyInputAirport(){
    clickOn( query: "File");
    clickOn( query: "Define New Airport");
    clickOn( query: "#airportDoneButton");
    alert_dialog_has_header_and_content( expectedHeader: "Message", expectedContent: "Please fill all input fields");
}

@Test
public void fail_airportAlreadyExist(){
    clickOn( query: "File");
    clickOn( query: "Define New Airport");
    clickOn( query: "#airportName").write("Heathrow");
    clickOn( query: "#airportDoneButton");
    alert_dialog_has_header_and_content( expectedHeader: "Message", expectedContent: "Airport already exists");
}
```

Figure 4: Tests giving undesired outputs

2.2 Boundary and Partition Testing

We have incorporated boundary tests into our main test suite as our application is calculation intensive with variables that represent physical quantities (height of an obstacle, length of a runway etc...). We used these tests to reduce an otherwise larger number of test cases to a manageable number of tests. For example, the obstacle height parameter has been split into three partitions: [INT_MIN,0], [1, 100] and [101, INT_MAX] with [1,100] being the only valid partition. We have chosen 100 meters to be the maximum value of our valid partition as a 100 meter high obstacle would generate a take-off/landing slope of around 5 kilometers which is longer that any active commercial runway in the UK thus rendering any runway totally unusable.

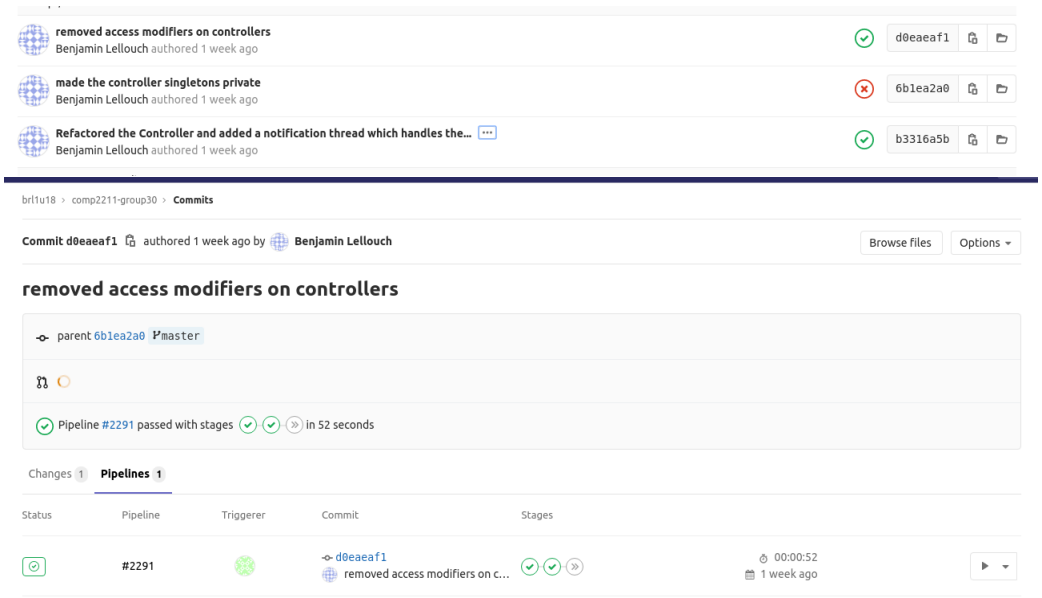
```
// Boundary Tests ////////////////////////////////////////////////////////////////////////

@Test
public void boundaryTest_Obstacle_Min()
{
    clickOn( query: "File").clickOn( query: "Define New Obstacle");
    clickOn( query: "#obstacleName").write("Boundary Obstacle");
    clickOn( query: "#obstacleHeight").write("1");
    clickOn( query: "#obstacleDoneButton");
    FxAssert.verifyThat( nodeQuery: "#obstacleBox", (ComboBox<Obstacle> o) -> {
        String val = o.getValue().getName();
        return val.equals("Boundary Obstacle") && o.getValue().getHeight() == 1;
    });
}
```

Figure 5: Minimum valid value test

2.3 Regression Testing

In order to test for regressions, we have decided to use the CI/CD (Continuous Integration, Continuous Development) feature that Gitlab offers. We have set up runners which, at every commit on any branch, will build the application and will run our test suite on that build when the commit was made to the master branch. This helps check that our application builds in the first place but it also helps us test for regressions.



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running MainTest
Prism pipeline init order: sw
Using native-based Pisces rasterizer
Using dirty region optimizations
Not using texture mask for primitives
Not forcing power of 2 sizes for textures
Using hardware CLAMP_TO_ZERO mode
Opting in for HiDPI pixel scaling
*** Fallback to Prism SW pipeline
Prism pipeline name = com.sun.prism.sw.SWPipeline
(X) Got class = class com.sun.prism.sw.SWPipeline
Initialized prism pipeline: com.sun.prism.sw.SWPipeline
 vsync: true vpipe: false
Loading Prism common native library ...
      succeeded.
Tests run: 15, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 154.854 sec
QuantumRenderer: shutdown
Results :
Tests run: 15, Failures: 0, Errors: 0, Skipped: 0
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  02:38 min
[INFO] Finished at: 2020-03-27T05:08:12Z
[INFO] ------------------------------------------------------------------------
Version:      12.8.0
Version:      12.8.0
Job succeeded
```

2.4 Scenarios

We also implemented automated tests for some of the scenarios that we defined in the envisioning process to show that the application correctly functions in the way we originally anticipated. Due to the fact that the FileChooser dialog is not itself a JavaFX dialog (and instead makes use of a form directly from the OS), we were only able to write automated TestFX tests for scenarios 1 and 2. Scenarios 3 and 4 were instead ran manually to ensure the expected behaviour was exhibited.

```java
// Scenario 1: Lauren (Runway Technician)
@Test
public void scenario1()
{
    clickOn( query: "File").clickOn( query: "Define New Airport");
    write("Bristol").clickOn( query: "#airportDoneButton");
    clickOn( query: "#airportMainBox").clickOn( query: "Bristol");
    clickOn( query: "File").clickOn( query: "Define New Runway");
    clickOn( query: "#runwayDegree").clickOn( query: "09");
    clickOn( query: "#runwayPosition").clickOn( query: "C");

    clickOn( query: "#todaLeft").write("7500");
    clickOn( query: "#toraLeft").write("7500");
    clickOn( query: "#asdaLeft").write("7500");
    clickOn( query: "#ldaLeft").write("5000;");

    clickOn( query: "#todaRight").write("7500");
    clickOn( query: "#toraRight").write("7500");
    clickOn( query: "#asdaRight").write("-7500");
    clickOn( query: "#ldaRight").write("5000");

    clickOn( query: "#airports").clickOn( query: "Bristol");
    clickOn( query: "#runwayDoneButton");

    alert_dialog_has_header_and_content( expectedHeader: "Message", expectedContent: "Please ensure only positive values are used for measurements");

    clickOn( query: "#asdaRight");
    clearTextField( size: 5);
    write("7500");

    clickOn( query: "#runwayDoneButton");

    alert_dialog_has_header_and_content( expectedHeader: "Message", expectedContent: "Please ensure only numbers are used as inputs for measurements");
```

3 Planning

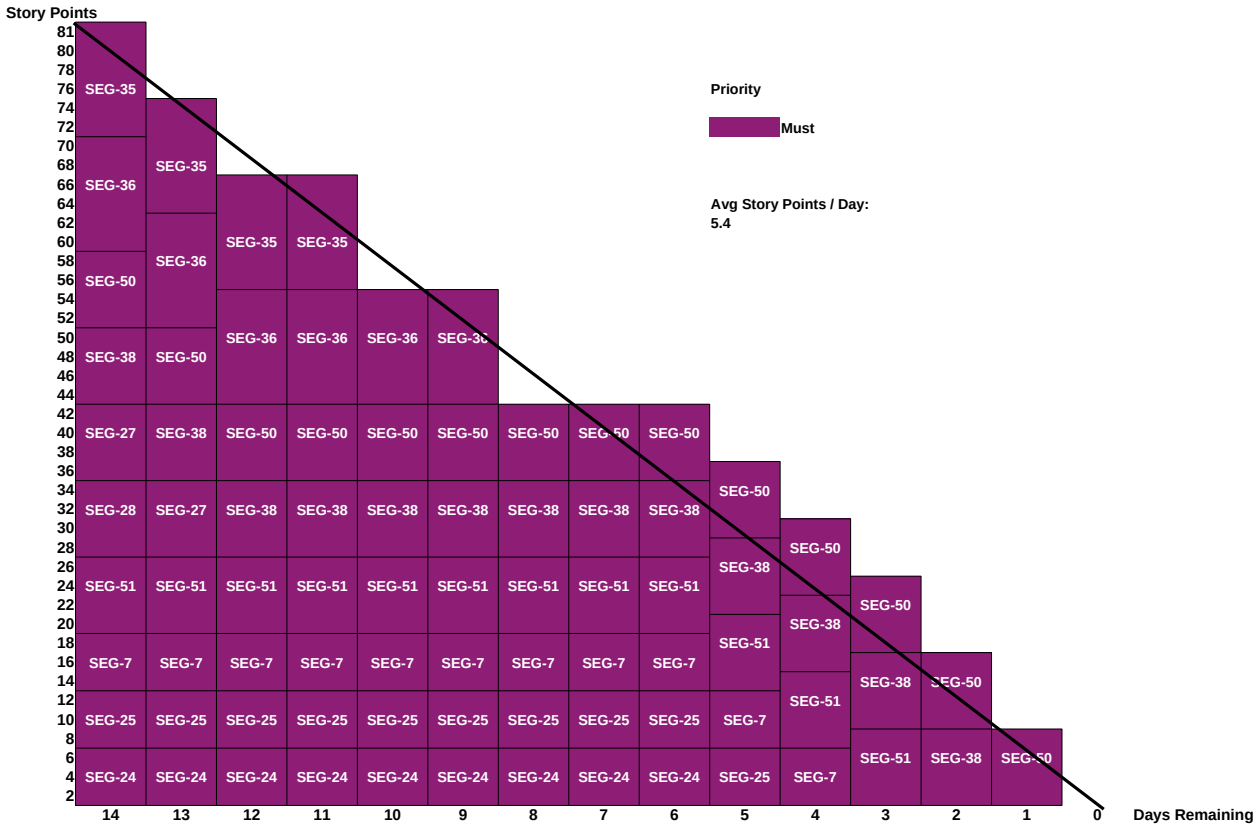
3.1 Sprint 2: Review

3.1.1 Completed Stories

We were able to complete all the user stories that we had set out to do. You can find these completed stories below.

ID	Name	Description	Priority
SEG-35	Runway Sideways	As an Airfield Operations Manager I want to be able to visualise the runway with the obstacle from a sideways perspective so that I can decide whether official calculations are necessary or the runway should be closed.	MUST
SEG-36	Runway Bird's-eye	As an Airfield Operations Manager I want to be able to visualise the runway with the obstacle from a bird's-eye perspective so that I can decide whether official calculations are necessary or the runway should be closed.	MUST
SEG-50	Simultaneous View Runway	As an Airfield Operations Manager I want to be able to visualise the runway from both a sideways and bird's-eye view simultaneously so that I can compare both perspectives to help determine whether to close the runway or re-declare it.	MUST
SEG-38	Runway Rotation	As an Airfield Operations Manager I want the system to automatically rotate the top-down view to the appropriate angle based on the compass heading so that I am able to easily visualise the runway re-declaration.	MUST
SEG-51	Clear and Graded Area	As an Airfield Operations Manager I want to be able to see the clear and graded area on the bird's-eye view of the runway so that I can determine if official calculations are required dependent on if the obstacle is located in that area.	MUST
SEG-24	Obstacle Notification	As a Runway Technician I want to receive a notification from the system so that I know that I have added an obstacle.	MUST
SEG-7	Successful Runway Revision Notification	As a Runway Technician I want to receive a notification from the system so that I know that I have successfully a revised a runway.	MUST
SEG-25	Runway Update Notification	As a Runway Technician I want to receive a notification from the system so that I know that runway values have changed.	MUST
SEG-27	XML Airport Import	As a Runway Technician I want to be able to import details of the airport via an XML file so that I do not have to manually define it every time I use the system.	MUST
SEG-28	XML Obstacle Import	As a Runway Technician I want to be able to import obstacles via an XML file so that I do not have to manually define them every time I use the system.	MUST

3.1.2 Burndown



3.2 Sprint 3: Plan

3.2.1 Backlog

- Benjamin Lellouch (B)
- Rajpal Dhillon (R)
- Iman Dzulhadi (I)
- Deniz Matur (DM)
- Daniel Best (DB)

The number at the beginning of each task represents the number of story points of that task.

Story: JPEG Runway [SEG-43]

Tasks:

- (2)(R) Code middle-tier which saves the runway to an image in a JPEG format
- (2)(DM) Implement UI which enables the user to save the runway visualisation
- (1)(DM) Testing & debugging

Total story points: 5

Story: PNG Runway [SEG-44]

Tasks:

- (2)(DM) Code middle-tier which saves the runway to an image in a PNG format
- (2)(R) Implement UI which enables the user to save the runway visualisation
- (1)(DM) Testing & debugging

Total story points: 5

Story: GIF Runway [SEG-45]

Tasks:

- (2)(R) Code middle-tier which saves the runway to an image in a GIF format
- (2)(DM) Implement UI which enables the user to save the runway visualisation
- (1)(R) Testing & debugging

Total story points: 5

Story: Runway Colour Scheme [SEG-46]

Tasks:

- (4)(B) Reformat back-end view to allow for color changing
- (3)(B) Implement UI which enables the user to change colors of runway
- (1)(DM) Testing & debugging

Total story points: 8

Story: Screen Reader [SEG-47]

Tasks:

- (8)(I) Code middle-tier which enables reading of UI elements
- (3)(R) Implement UI which enables the user to switch screen reader on
- (2)(DM) Testing & debugging

Total story points: 13

Story: Extra Visual Control [SEG-42]

Tasks:

- (8)(B) Code back-end to change scale of view
- (4)(B) Implement UI which enables the user to save the runway visualisation
- (1)(R) Testing & debugging

Total story points: 13

Story: XML Data Import [SEG-29]

Tasks:

- (3)(DM)Create XML parser for other data in the back-end
- (2)(DM)Add XML parser instance to middle-tier
- (2)(DM)Implement UI which allows to select XML file to import
- (1)(R)Testing & debugging

Total story points: 8

Story: XML Obstacle Export [SEG-30]

Tasks:

- (3)(I)Code back-end which prints Obstacle XML to a file
- (2)(R)Add XML writer instance to middle-tier
- (2)(DM)Implement UI which allows to select obstacles to export
- (1)(I)Testing & debugging

Total story points: 8

Story: XML Airport Export [SEG-31]

Tasks:

- (3)(I)Code back-end which prints Airport XML to a file
- (2)(R)Add XML writer instance to middle-tier
- (2)(DM)Implement UI which allows to select airports to export
- (1)(I)Testing & debugging

Total story points: 8

Story: XML Data Export [SEG-32]

Tasks:

- (3)(I)Code back-end which prints Data XML to a file
- (2)(R)Add XML writer instance to middle-tier
- (2)(DM)Implement UI which allows to select other data to export
- (1)(I)Testing & debugging

Total story points: 8

Story: 3D Runway View [SEG-39]

Tasks:

- (10)(DB)Create 3d viewer in back-end
- (2)(DB)Implement UI which allows to switch between 2D and 3D
- (1)()Testing & debugging

Total story points: 13

Story: Print Visual Representation [SEG-40]

Tasks:

- (5)(B)Add printer support in middle-tier
- (2)(DM)Implement UI which allows to print visual representation
- (1)(DM)Testing & debugging

Total story points:

8

Story: Real-World Overlay [SEG-41]

Tasks:

- (8)(DB)Add backend to support images as background in Views
- (3)(R)Implement UI which allows to swich between virtual and read-word visualisation
- (2)(R)Testing & debugging

Total story points:

13

Story:Print Result [SEG-43]

Tasks:

- (5)(DB)Add printer support in middle-tier
- (2)(I)Implement UI which allows to print results
- (1)(I)Testing & debugging

Total story points:

8

3.2.2 Burndown