### **School of Electronics and Computer Science**

### **University of Southampton**

### 

### **Runway Redeclaration Tool - Increment 1**

### **COMP2211 Software Engineering Group Project**

Team 10: Andrew John Evans (aje3g16)

Lyubomir Tsvetanov Hristov (lth1u17)

Suyash Datt Dubey (sdd1n17)

Kemal Yesildagli (ky1g17)

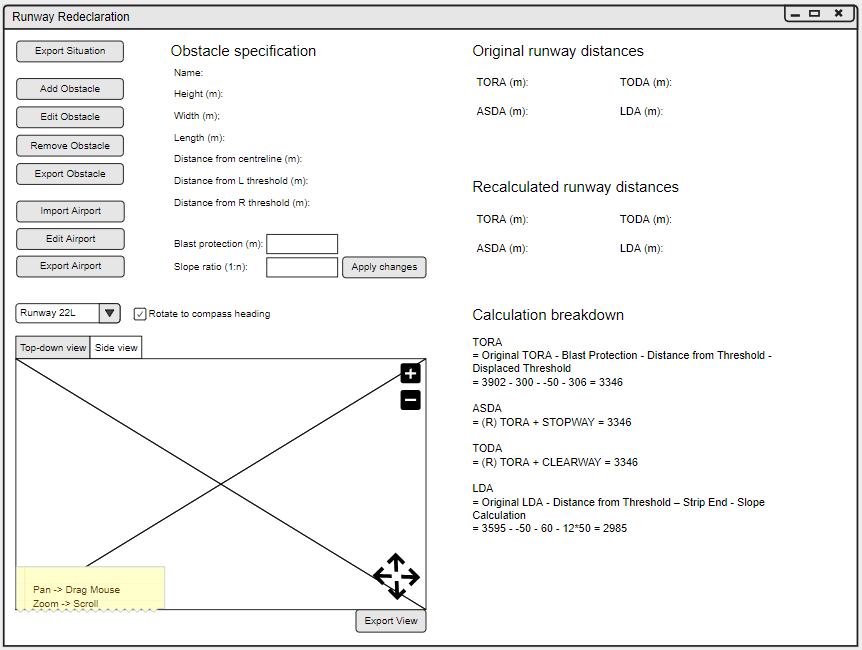
Denis Kovachev (dk1g17)

**Discussion of key design choices / decisions**

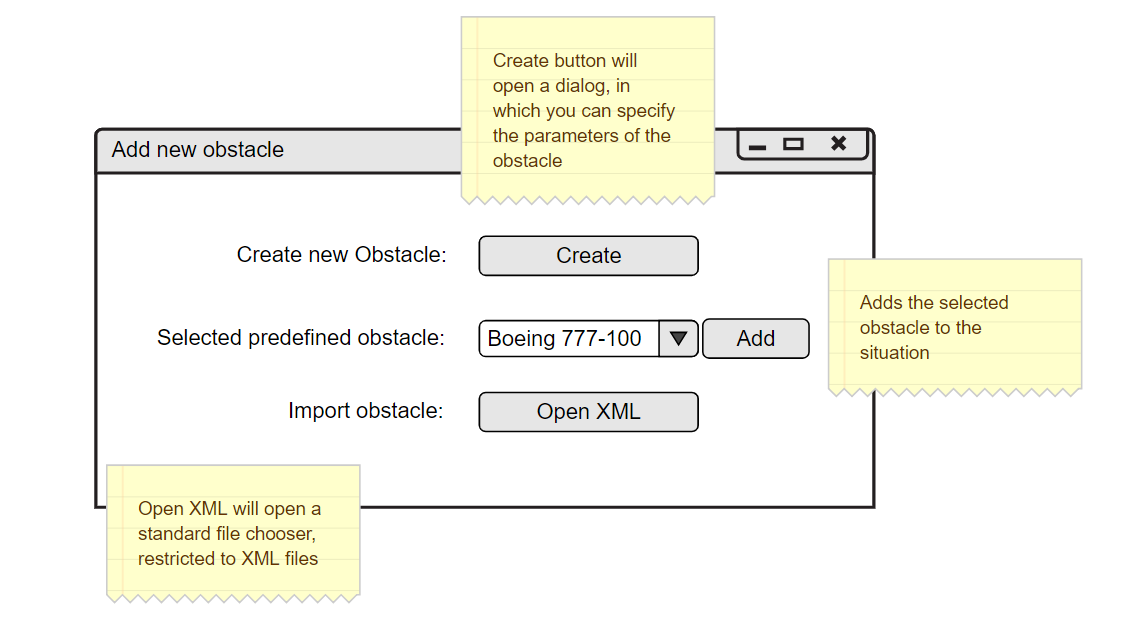
* Universal UK compatibility
  + We have decided to make a comprehensive model of a UK airport, so that our application can be used at any airport. For that purpose we have created an OOP model of a UK airport.
    - Airport class containing all of an airport’s runways
    - Runway class containing all of the parameters of a runway
    - Obstacle class containing all of the parameters of an obstacle
  + This model allows for airports using the application to have a custom profile.
* Calculation Functionality
  + The Runway class contains a calculate method which recalculates the new TORA, TODA, ASDA and LDA values when an obstacle is present on the runway. We decided that the method should be part of the Runway class as it uses/modifies most of the parameters of a Runway Object.
* Manual Input/Output
  + As you can see from the mockups, options are provided for manual input of data into the model.
* Predefined obstacles
  + In our ObstacleManager class, we have predefined 5 planes as obstacles and 5 more obstacles so that an obstacle can be quickly added.
  + We allowed the parameters of these obstacles to be edited, if necessary, after adding them to the runway - e.g altering position.
  + We have decided to follow the Singleton pattern for the ObstacleManager class, which allows us to use the same instance anywhere we need.
* Visualisation of data and recalculation
* Print out results to text file
  + The WriteFile class has the functionality to write to a text file.
  + It outputs original and recalculated runway distances, obstacle specification, airport data and more data related to the runway in a human readable format.
* MVC Model
  + We chose to use the MVC design pattern for our project, to allow for reuse of components in another context with minimal modifications. To do so, we began with a UML diagram showing the different classes we intended to start from (this set of classes grew over the course of increment 1).
  + We separated the classes such that the view and the model communicated only through the controller, giving us a great deal of modularity. This variation of the MVC pattern is sometimes referred to as Model-View-Presenter.
* View design decisions
  + With the design of the view, we aimed for visibility of a lot of the information on the main frame, as we wanted to make sure that anyone qualified could see all the necessary information at once.
  + At the same time, we didn’t want to overcrowd the main UI, and as such, created a variety of dialogs and other menus for functionality that would overcrowd the UI otherwise.
  + We aimed for an intuitive design, and as such, chose the components such that their purpose is obvious. As we develop this further, we will continue to test the UI against scenarios and welcome any feedback, to make the UI easier and clearer to use.

**Key storyboards**

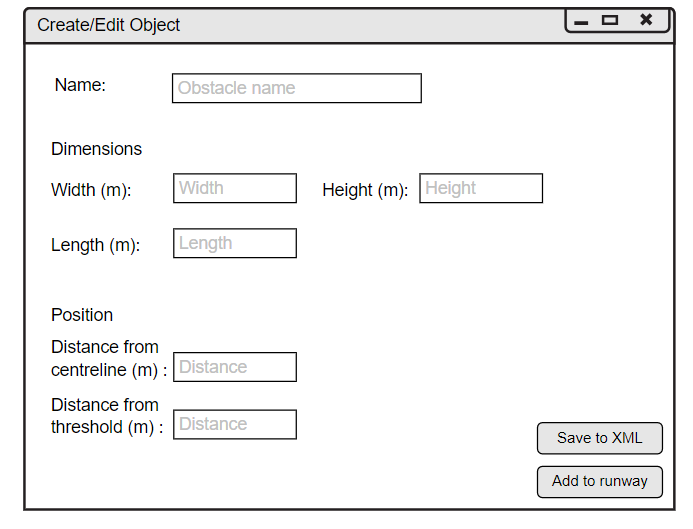
**Main Menu**

****

**Add new obstacle to situation**

****

**Creating/Editing an obstacle**

****

**Scenarios**

Scenario 1:

* Teo opens the application and selects a Runway from the comboBox above the views.
* He then navigates to the “Add Obstacle” button on the left.
* A window opens and Teo navigates to the “Create obstacle” button
* A form opens and Teo fills it with the obstacle information, after which he presses “Apply Changes”.
  + If any of the Input is different from what is required typewise, an error message pops up saying “Invalid parameters”.
  + If the form is correctly filled in, a message pops up telling you “you were successful”
* After hitting apply changes, the recalculation is displayed on the main screen.

Scenario 2:

* Stefi opens the application and selects a Runway from the comboBox above the views.
* She then navigates to the “Add Obstacle” button on the left.
* A window opens and Stefi navigates to the Predefined Obstacles comboBox and picks an obstacle from the list.
* After picking, she presses the button “Add”, adjacent to the comboBox, and after that the calculation is displayed on the main screen.

**New tools introduced**

As part of the design process, we created storyboards for the main parts of the user interface. To facilitate this, we used a online wireframe tool called **moqups.com**. Alternatives exist to this, such as **Adobe XD** and **Balsamiq** - just to name a few. We chose Moqups as it is free to use, making it much more accessible to all team members. Also, it is cloud based and allows for sharing of the storyboard among the team. Moreover, the UI design of the website itself was incredible intuitive and easy to pick up.

**Responses to feedback**

1. **Redeclaration calculator is too general - use job title instead**
   1. Since the specification does not mention a specific job title, we named the operator of the product as “Redeclaration Calculator”.
2. **Our team is a developer as a stakeholder**
   1. We agree since as developers, we are actively involved in the project, affected by its outcome, and we can influence is outcome.
3. **Overall mistakes and semantic issues fixed.**
4. **Product Backlog - We can also put the size of the product backlog to improve our report**
   1. Moving forward with sprint plans, will incorporate the size of user stories.
5. **Severity for developer made mistakes is low**
   1. For risk analysis, we deduced a risk factor of ⅖ for developer made mistakes, when it should have been ⅘ or more, since any mistake not caught by our testing practices could be very dangerous for all stakeholders, risking the lives of the passengers and the reputation of all other stakeholders.
6. **Tools used - Compare with competitors tools and think why you chose the ones we chose**
   1. For communication between team members, we chose Slack since it had the cleanest interface and it was free. Slack is secure and the use of two factor authentication reinforces that fact. Even though it’s secure, we mostly arrange face-to-face meetings to plan and discuss, so that our work or our client’s data is not leaked under any circumstance.
   2. For task management, we chose Trello over other competitors since its free, simple, has an intuitive card system, and provides a clean interface for our workflow. We could have used another tool if the team and the project was a lot bigger.
   3. For version control, we chose GitHub since it is the most popular and UoS gives us the chance to obtain a premium account for free, giving us the ability to create and maintain a private repository.
   4. For IDE, we thought about using IntelliJ or Eclipse. Since IntelliJ was the one we were all familiar with, and students get access to the Ultimate edition for free, we went with that choice.
   5. For documentation, we chose Google Docs over competitors like Microsoft Online since our documents are easily saved to our team’s Google Drive, and we are confident in its security.

**Sprint 1 burndown chart (Key)**

**1:** Universal UK compatibility

**2:** Calculation Functionality

**4:** Manual Input/Output

**3:** Predefined obstacles

**5:** Visualisation of data and recalculation

**19:** Print out results to text file

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tasks | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 |  | |  | | | |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|
|
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |  |  | |
|
|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 |
|
|
| 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 1 |
|
|
| Day | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**Increment 2 Plan**

|  |  |  |
| --- | --- | --- |
| S | 6 | Recalculation breakdown |
| M | 11 | Runway and Threshold Selection |
| XL | 14 | Side-on visualisation |
| M | 1 | UK Compatibility Integration |
| L | 7 | XML Input/Output |
| M | 17 | Export displays in different formats |

**Pushed to increment 3**

|  |  |  |
| --- | --- | --- |
| S | 12 | GUI Notifications |

* Recalculation breakdown
  + Add a method doing the recalculation breakdown (possibly as Strings to begin with) that show the different parameters and how they were calculated.
  + Update GUI Panel to reflect the changes of the obstacle
  + Comparison of breakdown output to calculation breakdowns of Heathrow examples.
  + Additional testing of output
* Runway and threshold selection
  + Add more sample runways/thresholds and provide the ability to select them
  + Update the calculations when a different runway/threshold is selected
  + Acceptance testing
* Side-on visualisation
  + Create a panel that provides a side view of the runway, the obstacle on it and the relative values of the runway parameters
  + Show recalculated TORA, TODA, ASDA, LDA etc on side view
  + Testing of correctness of output (acceptance testing)
* XML Input/Output
  + Create an XML structure with sample data about obstacles
  + Create a class, which provides the ability to read the XML file and adapt this method to the currently working system
  + Add error handling capabilities
  + Thorough testing of reading and writing files (Unit testing)
* Integrating the Model with the GUI fully.
  + Create a GUI component for creating and selecting an Airport profile.
  + Add validation on the inputs, so no erroneous data enters the model.
  + Acceptance testing
* Export displays in different formats
  + Identify the most common image output formats for images
  + Create methods to take the view and export it in various file formats
  + Add functionality for choosing location and file type of the exported picture.
  + Acceptance testing

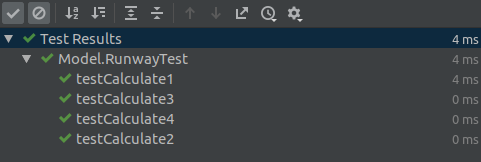
**Pushed to increment 3**

* GUI Notifications
  + During sprint 1, we determined that we underestimated the amount of work implementing a GUI for editing airport would be. As such, we have pushed the GUI creation (for UK compatibility) to Increment 2 and to balance our workload, we have pushed the implementation of GUI notifications to increment 3. The nature of the way we structured our sprints meant that we can ensure the “Must” user stories can be pushed back slightly, and still get done before the final deliverable.

**Increment 2 - Day 0 burndown chart**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tasks | 17 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Day | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**Calculation tests**

****

**UML Class diagrams**

