CS261

SOFTWARE PROJECT

FINAL REPORT

March 13, 2023

Group 24

Contents

1 Introduction

The problem we set out to solve consisted of creating a project management aid software which could evaluate the riskiness of a project, highlight key areas of the project which were causing the most risk, suggest improvements, and learn from past projects in order to be able to evaluate risk more accurately. We set out to do this through a non-linear optimisation model hosted on a web-front for convenient access.

Live (for a short time after submission) at: `https://storied-frangollo-58861d.netlify.app/`

2 Overview of the system

2.1 General Overview

Our product is split into a front-end and back-end respectively. They communicate through the means of an AWS API-Gateway as the back-end is stored in lambdas and the front-end is hosted on Netlify. This allowed us to keep the code bases for the front and back end separate and yet still have them interact seamlessly. The front-end fulfils a larger percentage of the customer requirements as it is what the customer will interact with while the back-end provides the database and the functionality of evaluating risk and adapting to past projects, therefore satisfying a few of the most crucial requirements.

2.2 Front-end

The front-end has a handful of features which all help satisfy customer requirements. The home page has an overview of all the user's projects with a few simple tracking metrics displayed for a convenient way to view all projects at once. These metrics are things such as the budget, money spent so far, overall risk etc. A project can then be expanded to display more detailed information, individual areas of risk are displayed with a short message entailing what changes should be made to lower risk and why. Maintaining a list of metrics allows the system to satisfy the 2nd requirement of the initial specification 'Maintain a list of metrics, looking at both the code base and 'softer' aspects of team management'

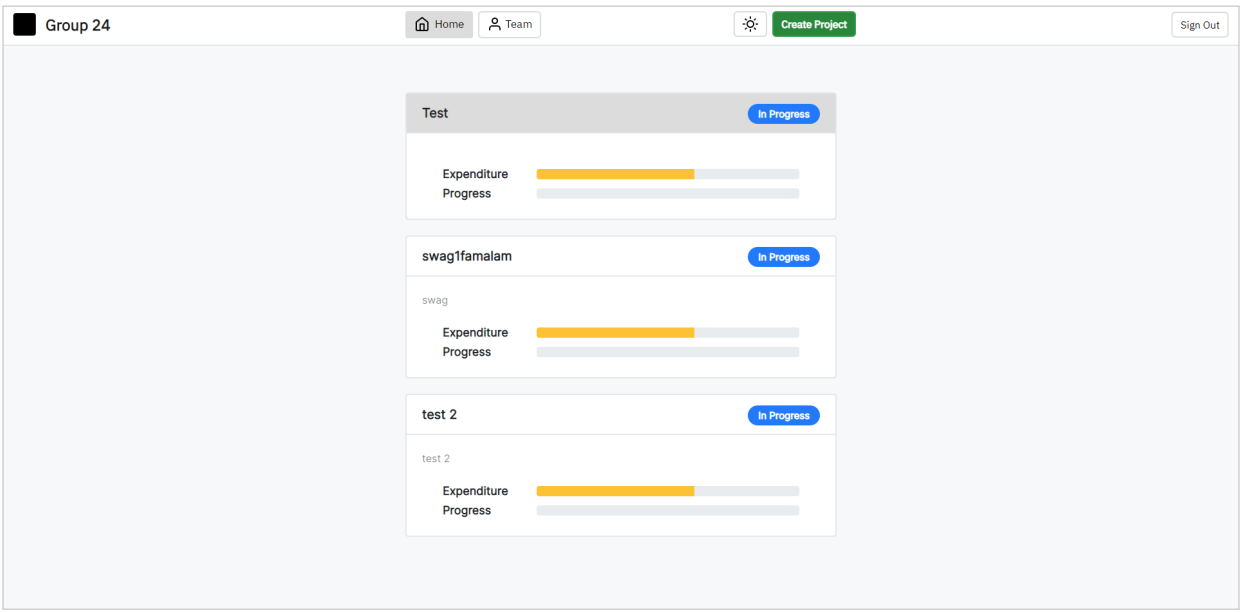

Figure 1: Homepage: Light

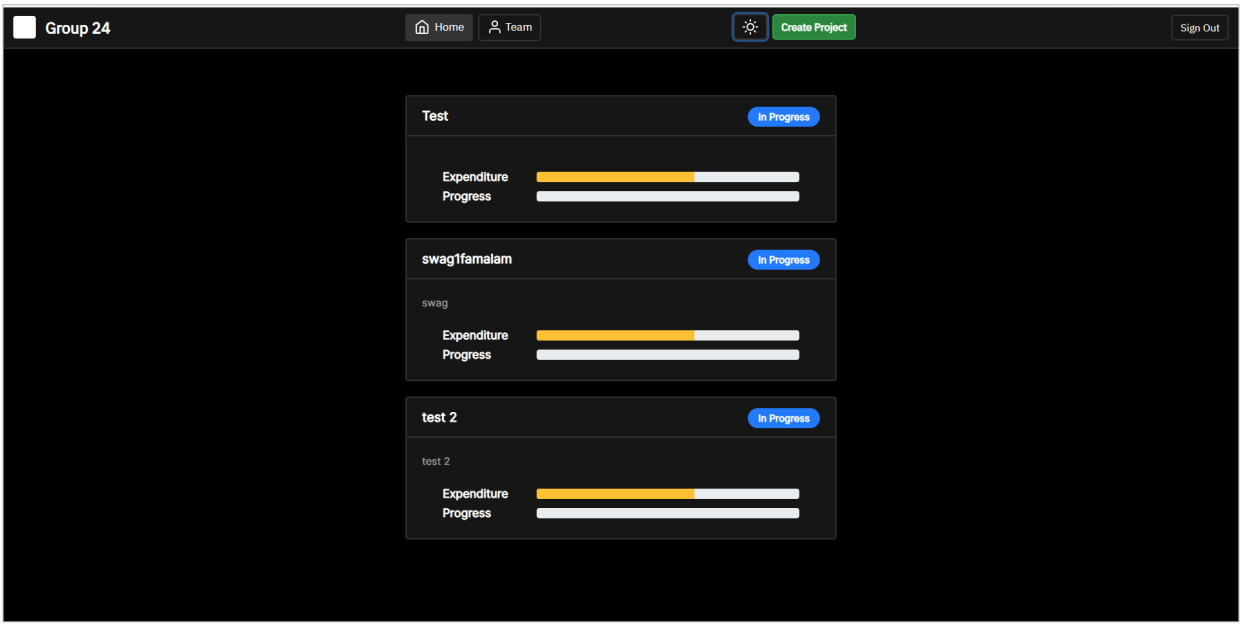

Figure 2: Homepage: Dark

Users can also easily enter new projects on a different page with easy to understand sliders and input boxes, all boxes must be filled in order for our model to work and the information we are requesting is all easily known or able to be estimated reasonably accurately by someone in a managerial position. All metrics can also be updated at any time and the risk is re-evaluated whenever an update occurs. The update takes only a few seconds so metrics can easily be updated and changed whenever the user requires. There are validation checks that are applied to the metrics input such as a deadline cannot be before the current date and budget cannot be below 0. As well as ensuring that inputs make logical sense this protects against divide by 0 errors and negative values which could flip the constraint meanings in our risk evaluation model

Further, for each team member they can easily add team members with the level of their skill(from 1- 10) when they enter the team page, here they can also check the members of the team

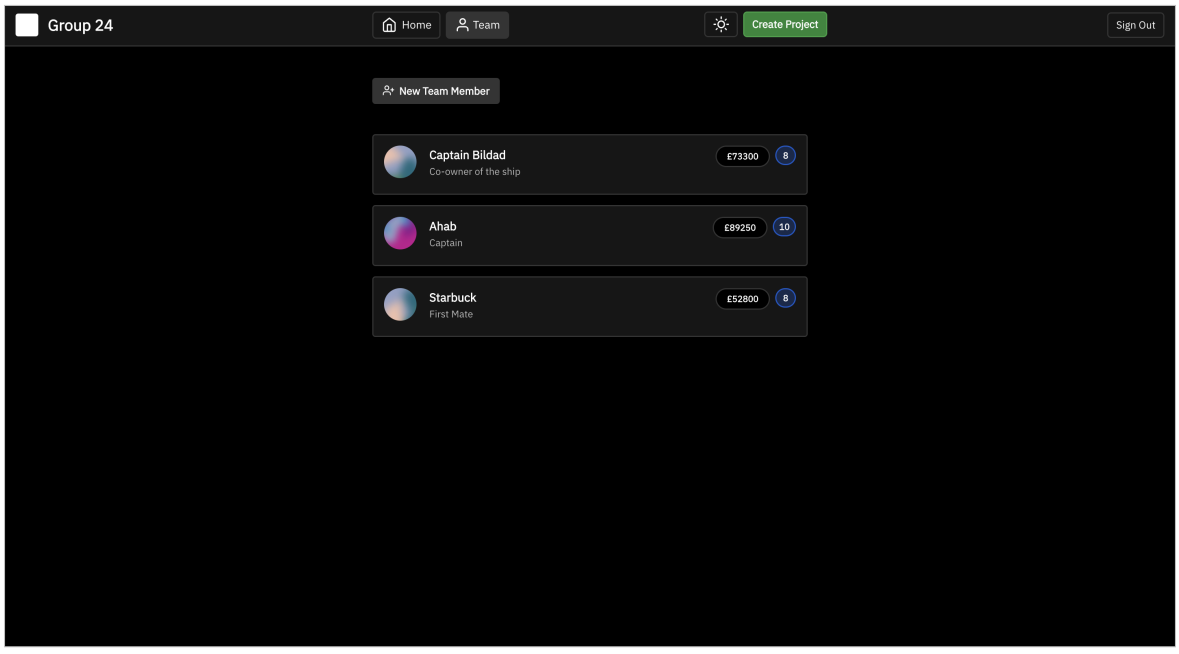

Figure 3: Team Member page

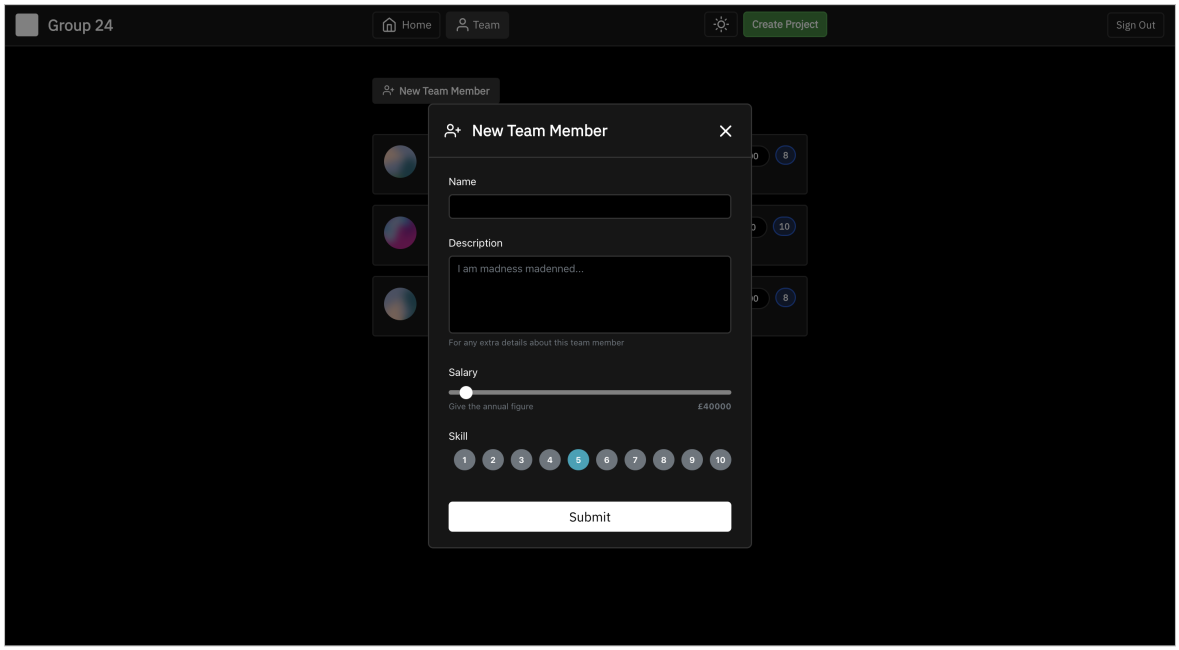

Figure 4: Team Member page

When creating tasks for a project, users can add task name and descriptions of projects. This is done on creation of a project alongside adding team members of the project, values for the metrics of the project, such as Budget, Expenditure and End time; all tasks about the project can be added on this page or later through the edit feature.

In the project page, users can check an overview of their project, such as, the expenditure and budget. Also on this page, the risk evaluation and the history of evaluations are shown with suggestions to this project. Milestones of this project can also be added the modify in this page.

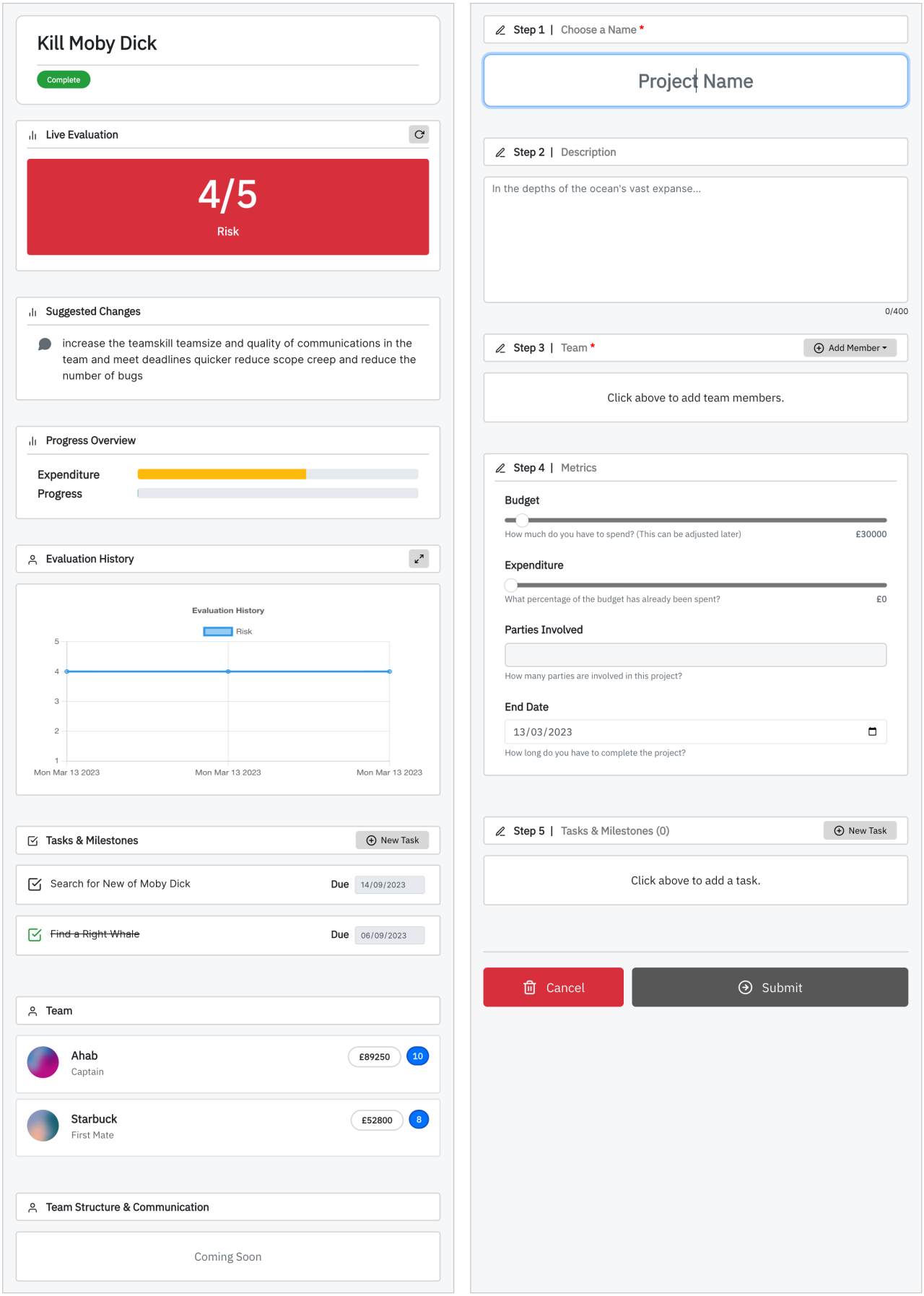

Figure 5: Project page & Create Project page

When a project is finished, whether successful or not, it can be marked as so and this will trigger the learning functionality of the model to adjust coefficient values to create a more accurate feasible region so that the riskiness of future projects can be evaluated more accurately. This is discussed in more detail in Section 2.3.3

There is a basic login and sign up feature that is robust enough to detect duplicate accounts and it also stores the passwords using a salted hash in order to ensure user's passwords are stored securely. Once logged in, users can only see projects related to their account and all data is stored in an AWS database with in-built features to keep the user's data safe and secure, in the case of the hashed passwords this provides an extra level of security to the salted hash.

There is a to-do list page where tasks can be entered alongside a deadline, priority and description. From this to-do list, we extract the estimated time it would take to complete each task (this is the difference between the deadline and date of entry); the number of tasks that have been ticked off; and the cumulative lateness which is calculated as the sum of the differences between when each task was ticked off and its deadline. These 3 pieces of information are used in conjunction with other metrics as input for our model. Additionally, the to-do list also allows the user to keep track of the project in a centralised location, near to where information about other project metrics are stored therefore providing a broad overview of the project. Once again, the tasks can be added and removed whenever the user likes keeping the software as easy to use as possible. Overall the front end allows the system to satisfy the 1st requirement from the initial specification 'Allow a user to track a software development project that they manage.'

2.3 Back-end Mathematical Model

2.3.1 How the model calculates risk

The ability to calculate the riskiness of a project is at the core of the system as the 3rd requirement in the initial specification states 'Evaluate the initial riskiness of a project, and periodically update this evaluation as the project develops'. Our system combines optimization and human intuition to create a model that can analyse the relationships between metrics in order to assign a dynamically updating risk rating to a given project. In particular, 'the model takes advantage of the emergent behaviour that arises from applying many constraints' in order to calculate the riskiness of a project. Each constraint is some inequality that takes a subset of the metrics input by the user as variables, in some cases these metrics are multiplied by some coefficient. We refer to the region in which every inequality is satisfied as the feasible region and this is deemed to be where an extremely 'safe project' and therefore the further away a project is from this region the higher the risk rating assigned to it. After testing our model on a data set consisting of fake projects that we deemed to be in particular risk levels we decided on a distance to risk mapping which can be found at Section 4.1.1. We take 13 metrics as input from the user and therefore we can define a projects location as a coordinate in some 13-dimensional space and similarly the 'feasible region' is some 13-dimensional subspace in this 13 dimensional space.

Each constraint we have chosen encapsulates some sort of human intuition about the characteristics that should hold true for a project that is likely to succeed. For example, the constraint

$$\text{ProjectProgression} \leq \frac{\text{timeSpent}}{\text{totalTimeToCompleteProject}}$$

encapsulates how 'on track' a project is in terms of the likelihood of the project being complete before the deadline. For example, this inequality implies that if 9 out of 10 available weeks have passed a safe project would be at least 90 percent done. Some constraints capture more complex relationships such as $K \cdot Teamskill \cdot Teamsize \cdot Communications \leq A \cdot CumulativeLateness + B \cdot ScopeCreep + C \cdot bugs$. Here the RHS of the inequality represents how 'behind schedule' a project is and the LHS of the inequality represents how competent the team working on the project is. Therefore how far behind a safe project can fall is bounded by the competency of the team working on it. This is because a large, highly skilled team with excellent communication is more likely to be able to get a project back on track than a small low-skilled team with poor communication. The variables on the LHS of this inequality are multiplied together as Teamskill is averaged out per person, effective communication will increase everyone's productivity and Teamsize is the number of people therefore all of these metrics compound each other with respect to measuring the competency of a team. The variables on the RHS are added together as they all correlate to some amount of time that will be required to fix them and therefore the sum of them is correlated with the total amount of time it will take to get the project back on track. The coefficients K, A, B and C are values that we will be able to learn once we have more data regarding projects

that have failed or been successful. Based on some mock test data we have initial values of 0.01, 5, 5 and 5 respectively.

2.3.2 How the model recommends changes

The ability to recommend changes was necessary in order to meet the 2nd requirement of the initial specification 'Highlight areas that are potentially putting the project at risk and possibly make suggestions on how to address these problems'. This was also necessary to satisfy requirement FM5 'Users must be able to see which metrics pose the greatest risk to the project.' We have a function to recommend which metrics to change and how to change them in order to most effectively reduce the riskiness of a project. This function evaluates which constraint is being violated by the 'largest degree'. Each constraint is of the form $RHS \geq LHS$ where the degree to which it is violated $= LHS - RHS$ where the largest degree is the largest negative number we get after evaluating $LHS - RHS$ for each constraint. Each constraint is a key in a recommendation hashmap which then suggests decreasing values of metrics on the LHS and increasing the value of metrics on the RHS where possible. For example, if the constraint $projectProgression \geq \frac{moneySpent}{Budget}$ is causing the most risk we would recommend the user decreases the amount of money they spend per week for the rest of the project or get additional funding as they are on track to run out of money before they complete their project. We do not suggest that the user increases the project progression metrics as every user will be aware that getting more of a project complete will reduce the likelihood of it failing and therefore this suggestion would not be of any value for the user.

2.3.3 How the model learns

The ability to learn from past data satisfies the 5th requirement from the original specification 'Make use of both research and past projects in order to inform its future decisions and frameworks' and allows our system to more accurately assign risk ratings. In order to learn better values for coefficients used in the inequalities, where better implies values that provide more accurate risk assignments, a machine learning approach is used. Whenever a complete project is passed to the system it's location is calculated, if it is in the safe region and was completed successfully then we do nothing. Similarly if it failed but is outside of the safe region we do nothing. However, if it was successful and was outside the safe region we adjust the safe region to move it slightly closer to this project. We do this by iterating over each coefficient and increasing each coefficient by some small value and seeing if the distance between the safe region and the project, which can be thought of as our 'cost function', decreases. If the distance decreases we update the coefficient to this new value, if not we decrease the old value of the coefficient by some small value and test again. If neither increasing or decreasing the coefficient decreases this distance the coefficient is not changed. The same thing is done if an unsuccessful project is located inside of the safe region with the conceptual difference being that a successful project outside of the safe region results in the safe region increasing in size in the direction of the project, where as an unsuccessful project inside the safe region results in the safe region decreasing in size in the direction of the project.

3 Changes To Our Original Requirements and Design

3.1 Front-end Design

Since we reduced some of the functionality of our product,(measuring communication & assigning jobs), There are minor differences in the user interface we implemented and our initial user interface design. In the previous design, there was an area for the manager to check the group structure and assign jobs, but now the assign jobs area has been removed and there is only check group structure exist in the area. Also, the area for communication is removed from the precious design. Other parts remain the same. The reason why we removed these is: the product is used for risk evaluation, team discussion and manager assignment of work do not need to be completed through this product.

3.2 Back-end Model

We did not make significant changes to our Back-end Model, we still use the non-linear optimization in order to calculate the distance between the coordinate defined by the values of the projects metrics and the 'feasible region' and then check what band of risk this distance falls in. The changes we did make were in relation to the metrics we take as input and when handling some edge cases when converting the distance to a risk assignment.

3.2.1 Project Metrics

We made significant change to the metrics we take as inputs. Firstly, we no longer take in Git Commits as a metric. We decided the amount of git commits does not contribute significantly enough to the probability of success of a project, this can be for a number of reasons. Note that the number of git commits can be very easily forged by employees, by simply committing sections of code which add no significant value to the project. Another reason was due to the fact the number of git commits does not correlate to the quality of the commits. For instance consider these two cases: case 1 may have 1 commit which has the working functionality of many key aspects to the project and case 2 which has many commits which produce only one working component and results in many other bugs. It is clear that case 1 is better with respect to increasing the likelihood of the project succeeding, however if we took number of Git commits as a metric our model may say case 2 is better off, which is incorrect.

Secondly, we edited our Scope Creep metric. This initially was a user-inputted value which was on a scale of 1-10 to how likely it was that their project would be impacted by scope creep; 10 representing at least double the initial workload added. This was changed to the amount of days added tasks would take rather than a likelihood rating. This is because although scope creep can occur, it is only impactful when significant time and resources are needed in order to complete that additional tasks.This results in us having a more precise representation of the severity of scope creep a project may be affected by.

Thirdly, We also added an extra metric, salary, which is the cumulative amount of all the money that is intended to be spent from the budget to pay for all the employees working on the project. This is particularly important as the workers could still be poorly paid despite the budget being extremely high. This fact in conjunction with the fact that we designed a useful constraint involving salary which we believed was relevant to the motivation of workers and therefore relevant to the riskiness of the project, is why we decided to include this metric. The constraint is highlighted in our constraint table Section 4.1.1

Finally, we also edited the representation of Git Bugs to the same representation discussed about Scope Creep; the estimated amount of time needed to resolve the bugs. This is in part due to use not having enough time to implement the GitHub API feature - a member of our group not participating contributed heavily to this- and similar reasoning to why we deduced Git Commits was a redundant metric. Instead of having a raw number of bugs, it is much more beneficial to track the severity of bugs. Since 1 bug may prove much more difficult and complex than a series of many other smaller bugs. We came up with a way to measure the severity of particular bugs, $SeverityOfBugs \propto TimeAdded$ and resulted in a useful metric we can directly input to our optimisation model.

3.2.2 Calculating Risk

A small addition was made to the calculating risk section of our code. We added two functions which separately calculated risk alongside the risk assessment result from our non-linear optimisation model. The 2 additional functions each produced their own risk rating value based on constraints 2 and 3 respectively. We then took the max risk band value of all three methods to give us our final risk assessment value. This was due to a problem that arose when testing the risk from the non-linear optimisation model. The problem was due to the distances made by substituting extreme values of project progression and days passed (0.01 and 99 out of 100 days) did not yield a high distance value from the feasible solution that it should (< 1 unit). We concluded that this was as a result of the constraints being represented as percentages that had no constant multiplied to it to increase the distance a large enough amount respectively. Unfortunately we could not add in this constant as our constraints would no longer make logical sense and so an easy fix were to treat these separately as we also decided that they had a relatively large impact to project success.

4 Development Process

4.1 Back-End Mathematical Model and Risk Assessment

4.1.1 Calculating Risk

To calculate our assessment of risk we model our optimisation problem with respect to the following constraints:

Number	Constraint	Justification
1	$K \cdot TeamSkill \cdot TeamSize \cdot Communnications \geq A \cdot CumulativeLateness + B \cdot ScopeCreep + C \cdot Bugs$	If the quality of the team in conjunction with the team size is larger, we believe that the project can still be successful despite the fact that additional time can be added on as the team is skilled enough to have a higher chance of getting the project back on track.
2	$ProjectProgression \geq \frac{DaysPassed}{TotalDays}$	The project should progress the same amount as the time progresses to ensure the product is made before the deadline.
3	$ProjectProgression \geq \frac{MoneySpent}{Budget}$	If a large amount of the budget is spent relative to the amount the project progresses it is likely the project budget will finish before the project is completed.
4	$\frac{Salary}{TeamSize} \geq K \cdot TeamSkill$	A higher salary with respect to the skill level of the team could motivate the team to work accordingly; if members of the team are being underpaid with respect to their skill this could result in these members slacking in rebuttal.
5	$K \cdot TeamSize \cdot TeamSkill \cdot \frac{TotalDays - DaysPassed}{TotalDays} \geq Bugs$	If there are bugs as the deadline approaches there is a higher chance that the project will not finish on time. Additionally, a skilled team may be more experienced when dealing with bugs and so the deadline can be closer.
6	$TeamSkill \cdot Communication \geq K \cdot PartiesInvolved$	Working with other parties requires the team to have enough experience, alongside good communication. If this is not present, there is a fair chance the project will be more risky.
7	$\frac{TeamSkill \cdot Communication}{TeamSize} \geq K \cdot PartiesInvolved$	Similar reasoning to 6, however larger team size means communication is harder with other parties also, through research, it is shown that the optimal team size when using an agile methodology is 7 - 12 people, "the conclusion is that the optimal team size for the 300 FPs new development project on the PC platform using 3GL languages would be model Team = 7 with three developers as the optimal team size" [1], going higher than this could add unnecessary confusion to the project in turn resulting in possible delays and problems.
8	$\frac{Budget}{TotalDays} \geq \frac{MoneySpent}{DaysPassed}$	Spending too much money as the project goes on could result in the budget running out before the end product is made.
9	$(Budget - MoneySpent) \cdot \frac{TotalDays - DaysPassed}{TotalDays} \geq K \cdot ScopeCreep$	Scope creep results in additional tasks which must be added, these tasks may require money to implement and so there must be enough money to sustain the implementation of all scope crept tasks.
10	$\frac{Budget - MoneySpent}{(TotalDays - DaysPassed) + CumulativeLateness} \geq \frac{MoneySpent}{DaysPassed}$	There needs to be enough money to sustain extra time past the deadline if the project is falling behind.
11	$ProjectProgression \cdot TeamSkill \geq CumulativeLateness \cdot \frac{TotalDays - DaysPassed}{TotalDays}$	Same reasoning as 5.

These were all modelled using the Scipy.optimise Python library. We would then plot our project in a 13 dimensional plane and calculate the closest distance to the feasible region generated by our constraints. Depending on the distance we calculate they are placed into respective band levels representing their risk level:

Band Number	Distance Interval	Band Definition
1	$0 \leq x < 5$	Not Risky
2	$5 \leq x < 10$	Somewhat Risky
3	$10 \leq x < 20$	Risky
4	$20 \leq x < 40$	Very Risky

5	$40+$	Extremely Risky

After the distance is calculated we use the two additional functions discussed in Section 3.3.2 to calculate which output Band Numbers are unique to constraints 2 and 3 respectively. We then take the max band level returned by the non-linear optimisation function and these two additional functions, to leave us with our resulting risk estimation of the project.
The following tables show what Band Numbers are returned based on the values inputted into constraints 2 and 3 respectively:

Band Number	$LHS - RHS$	**Band Definition**
1	$0 < x$	Not Risky
2	$0 \geq x > -0.1$	Somewhat Risky
3	$-0.1 \geq x > -0.2$	Risky
4	$-0.2 \geq x > -0.35$	Very Risky
5	≤ -0.35	Extremely Risky

Band Number	$LHS - RHS$	**Band Definition**
1	$0 < x$	Not Risky
2	$0 \geq x > -0.15$	Somewhat Risky
3	$-0.1 \geq x > -0.3$	Risky
4	$-0.2 \geq x > -0.45$	Very Risky
5	≤ -0.45	Extremely Risky

Challenges: Many challenges were faced whilst developing the risk calculation; majority of these challenges were discovered through testing our product. A significant problem we faced was to do with the distance calculator in the Scipy.Optimise library itself. We used the SLSQP algorithm in order to calculate our minimum distance (Sequential Least SQuares Programming), which entails using an initial guess for the solution, however if this guess is too far away from the solution it would result in frequently unexpected values and occasionally the process would terminate early due to a "Positive directional derivative for linesearch". Another reason to why this may have been occurring is due to our functions being poorly scaled. However after many uses of our program, our product would 'learn', refine, better coefficient values which would in turn scale our functions better. More can be found at `https://en.wikipedia.org/wiki/Sequential_quadratic_programming`.

Another challenge lay in the weighting of our constraints, this is particularly to do with the fine tuning of our initial constraint coefficient values. We saw that changing the coefficients slightly would dramatically affect the hierarchy, whilst we aimed for every constraint to be weighted equally with respect to the impact on the resultant distance values, it proved a great challenge to accomplish this. To ensure some constraints had an affect on our risk assessment, particularly constraints 2 and 3, we had separate functions for them as outlined in Section 3.3.2. To obtain our preliminary coefficient values we procured artificial projects which we deemed fell in each of the risk bands and altered our coefficients accordingly to ensure the analysis done by our product would place these projects in their correct respective band. This led our product to correctly analyse projects inline with what our human intuition(and research) decided that the risk of projects were.

This section covers the third functional requirement in our project specification, "Evaluate the initial riskiness of a project, and periodically update this evaluation as the project develops". As our constraints consider both, initial metrics (team skill, budget, parties involved , etc...) and metrics which change throughout the project, (bugs, money spent, Lateness, etc...) which all have an effect on the risk assessment our product provides. Alongside this we also cover our third non-functional requirement, "The system should be intuitive and require little to no training" as all of our constraints are made via human intuition alongside the minimal training done to obtain preliminary coefficient values for our constraints.

4.1.2 Suggestions to Improve Projects

In order to satisfy our 'FM5' and 'FM4' requirements of the project specification, "Highlight areas that are potentially putting the project at risk and possibly make suggestions on how to address this problems." We developed a function called 'recommendChange' which is able to analyse the project with respect to each constraint and see which constraint the project was furthest away from satisfying. This is so that we could see which changes would

be most significant in getting the project closer to the feasible region. Each constraint was a key in a suggestion dictionary which stored a string which would output to the user if that constraint was the furthest away. Initially, each string would tell the user what metrics to adjust in order to make the project less risky. For example, if the constraint causing the most risk was

$$\text{ProjectProgression} \leq \frac{\text{MoneySpent}}{\text{Budget}}$$

we would suggest to the user that they should 'Spend less money each week for the remainder of the project or increase your budget'. However, after carrying out Beta Testing by asking people who were not part of the team to test our product we repeatedly got feedback asking why a particular change was recommended. Consequently, we took this into account and redeveloped our suggestion dictionary such that each constraint mapped to a string which would recommended a change and explained why that change was necessary. The string in the example above was then changed to 'Spend less money each week for the remainder of the project or increase your budget as you are on track to run out of money before you complete the project'.

4.1.3 Machine Learning AI and Past Project Manipulation

In order to implement the ability to learn we required our model to be able to adjust the feasible region when given past project data if the project was successful and outside our safe region or unsuccessful and inside our feasible region. In order to implement the first case we adjusted each coefficient of the model such that the feasible region was moved closer to the successful project that was not in the feasible region. However in the second case, the unsuccessful project being inside the feasible region resulted in our distance calculation function to return 0 and therefore we could not tell if our adjustments to the feasible region were resulting in it shrinking towards the direction of the unsuccessful project or not. We essentially needed to invert the feasible region so that we could then use the function that we used for case 1. This caused a significant delay in development. Then as the back-end developers were not sure how to proceed a team meeting was called to discuss how to solve this issue. Through collective deliberation and brainstorming, we realised we could reformat our model such that it takes in an additional 'flip' parameter that takes a value of 1 or -1. If the flip parameter is set to -1 every inequality is multiplied by -1 and this inverts the feasible region and consequently allows case 2 to be successfully implemented.

Challenges: No particularly large challenges arose when implementing this section of the program. Our greatest difficulty stemmed from the uncertainty of our group as a collective on how to implement the learning feature of our program, initially we desired to stretch the entire feasible region in one step towards/away from the completed project dependant on its success outcome. We concluded that this was not possible to implement and so sought other ways for our product to learn. This led us to the current method we have implemented, which shifts the feasible region accordingly.

This clearly demonstrates the fulfillment of functional requirement 5 "Make use of both research and past projects in order to inform its future decisions and frameworks" As we utilise the information provided from past projects inputted in the system when they are marked as complete to alter the assessment we give to future projects, as our feasible region shifts as a consequence.

5 Evaluation of the Final Product

5.1 Additional Features and Improvements

Given additional time we would be able to fully implement the effective measuring of our communication metric using a network flow as discussed in the planning document. We would also make a mobile application which would be employee oriented allowing them to input communication metrics, i.e. communication which takes place between sub-leaders and their respective teams and communication that does not involve the manager/boss. This application would also allow employees to assess their manager on the quality of leadership [2], a metric we discussed was important in risk assessment which we have not currently included.

We would also implement a more effective means of calculating our risk band distances, as currently these are manually hand picked distances based on fake data and intuition. This can be done using ratios and the "success rate method" which involves drawing ellipses around our feasible region which represented the percentage of successful projects, for instance we could represent our first band, not risky, as the ellipse around our feasible

region which held 90% of successful projects instead of less than 5 units away from our feasible region. This would give a more accurate measure for the bands we have and in turn provide a more accurate risk assessment, by also using past data inside of our program.

Furthermore, with more time we would take more care with our constraints as some can be modelled much more accurately. While we do not think the constraints we have made are incorrect better equations can accurately represent them. For example, consider Constraint 4 based on research the average software engineer salary is approximately £58,000 however the lower end is £48,000 and the higher end is £100,000+ [3]. If we model the average software engineer to have an expected pay of the average salary, it is clear that our constraint should be modelled with respect to some exponential graph however this is not due to the difficulty of obtaining the equation of the graph.

Given additional money and resources, we would be able to procure a relatively extensive synthetic data set of projects or possibly finance one. This would allow us to do further testing to ensure our mathematical model produces accurate results. Furthermore, with a data set we can train our model using our learn feature as outlined in Section 4.1.3 if we so wish.

5.2 Testing

5.2.1 User Testing

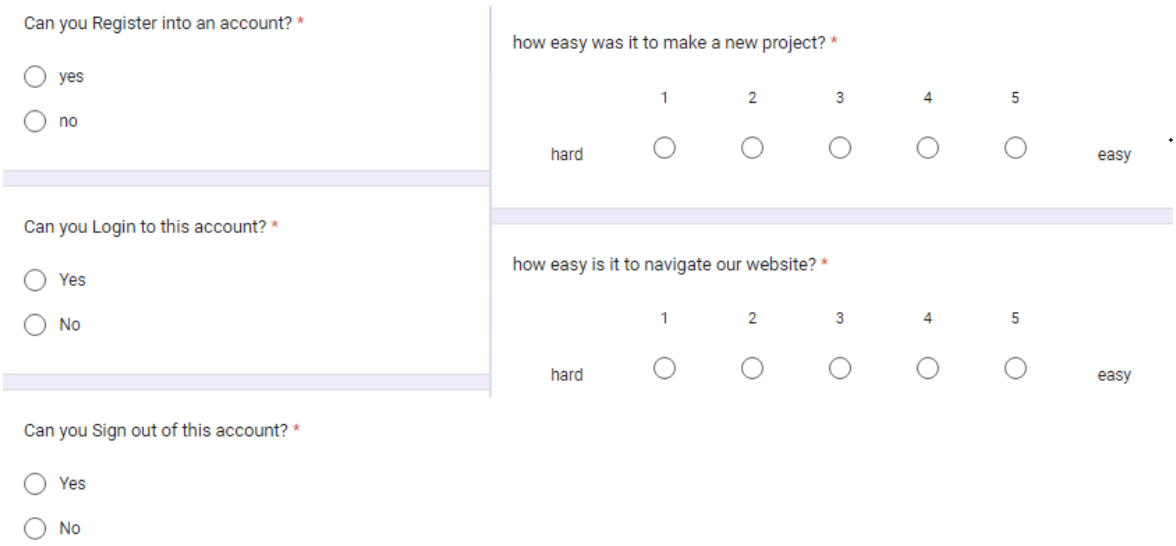

Figure 6: Product Test Google Form

Figure 4 shows a snippet from a google form we used for user testing, the user had to follow a list of tasks designed to test our website then fill out this form. The results were then visible to us and were able to gauge how easy different aspects of our website were to use. We had initially planned on having a physical form printed out that we would ask users to fill out in person whilst we allowed them to use the website from one of our laptops. However, as a result of getting an unpolished but functioning version the website deployed earlier than expected we were able to send users a link to a google form alongside a link to our website so that they could test remotely. This had many benefits. 1) This allowed us to have a much larger sample of users to get feedback from as increasing the number of users from which we gained feedback no longer correlated with the amount of time we had to spend to carry out the tests. For example, we could send the links to a groupchat consisting of 300 people in a matter of seconds, where as getting feedback from 300 people in person could take days. 2) There was no chance for members of the development team to give the users any advice on how to use the website as no team members were present whilst any of the users were testing the site. This allowed us to test if the website was intuitive to use which was a part of our requirements. 3) Google Forms automatically aggregates results. This saved us lots of time as manually calculating the average results of each question by hand would take lots of time and would have likely resulted in inaccuracies as a result of human error.

5.2.2 Sample of some unit tests we ran on the model

Test Data	Expected Output	Output
A project with metrics: Budget = 10000, Teamsize = 10, Project Progression = 50%, Teamskill = 10, ScopeCreep = 0, Communication = 10, Bugs = 0, Lateness = 0, NumberOfParties = 1, DaysPassed = 50, TotalDays = 100, MoneySpent = 5000, Salary = 4000	1/5 Not Risky with green background	1/5 Not Risky with green background
A project with metrics Budget = 10000, Teamsize = 10, Project Progression = 50%, Teamskill = 5, ScopeCreep = 4, Communication = 7, Bugs = 3, Lateness = 3, NumberOfParties = 2, DaysPassed = 50, TotalDays = 100, MoneySpent = 5500, Salary = 2000	2/5 Somewhat Risky with orange background	2/5 Somewhat Risky with orange background
A project with metrics Budget = 10000, Teamsize = 10, Project Progression = 50%, Teamskill = 2, ScopeCreep = 8, Communication = 5, Bugs = 5, Lateness = 7, NumberOfParties = 4, DaysPassed = 50, TotalDays = 100, MoneySpent = 5000, Salary = 1900	3/5 Risky with orange background	3/5 Risky with orange background

A project with metrics Budget = 10000, Teamsize = 10, Project Progression = 50%, Teamskill = 2, ScopeCreep = 25, Communication = 5, Bugs = 25, Lateness = 7, NumberOfParties = 4, DaysPassed = 50, TotalDays = 100, MoneySpent = 5000, Salary = 1900	4/5 Very Risky with red background	4/5 Very Risky with red background
A project with metrics Budget = 10000, Teamsize = 10, Project Progression = 50%, Teamskill = 1, ScopeCreep = 35, Communication = 5, Bugs = 40, Lateness = 10, NumberOfParties = 6, DaysPassed = 50, TotalDays = 100, MoneySpent = 5000, Salary = 1900	5/5 Extremely Risky with red background	5/5 Extremely Risky with red background

5.3 Verdict

Whilst we mitigated the frequent incorrect distances as outlined in Section 4.1.1 problems do still arise when extreme values are inputted, which garner unexpected and incorrect distance results. This raises questions about the robustness of our risk evaluation. However, we believe that as the coefficient values for our constraints become more accurate over time as a consequence of the program learning, these unexpected results will become less frequent and eventually no longer existent. In addition to this, some constraints are not properly fine-tuned and as a result also currently produce undesirable results when certain metrics are inputted. Despite this, after testing we discovered that for the majority of projects the evaluations were consistent with what we expected. With both correct and incorrect evaluations of our distances, our recommendChange and Learn function works perfectly as intended and we have no robustness issues regarding this. With more time we would have finished implementing the ability to calculate scope creep and bugs.

The product is capable of meeting all 5 requirements from the client's initial specification and therefore can be considered a successful product.

5.4 Testing against initial requirements

Key: **F** = Functional Req, **NF** = Non-Functional Req, **M** = Must, **S** = Should, **C** = Could, **D** = Don't

Requirement Number	Evaluation of how well we met the requirement
FM1	Successfully met as users are able to login to their account and create one if it does not exist. We were able to test this by having a person outside of the development team create an account and then login to that account without anyone telling them how. We also tested that no duplicate emails would be accepted and that passwords were stored securely.

FM2	Users are able to create projects linked to their specific account, the boxes and sliders have limited value ranges and didn't allow the users who tested the product to enter values that would have caused errors. The data is then submitted to a database through API gateway and as a part of testing we had people who weren't on the project team enter some mock values without knowing anything about the UI prior.
FM3	The website allows for individual metrics to be updated by resubmitting the entire metrics form, this can be changed in the future to not require entering all values again. Unexpected values and incorrect sequencing was tested for database integrity and it passed all tests.
FM4	Users can see the most current risk rating at anytime during a project and can recalculate the models evaluation of risk at anytime too, alongside the model automatically re-evaluating the risk on insertion or update of values. There is also a graph viewable for every project which displays the risk ratings overtime for a project, as the project goes on this graph just gets bigger.
FM5	This requirement has been successfully met as we have made a recommend change function which outputs a message to the user explaining what metrics should be changed and why. This has been discussed in Section 4.1.2
FM6	This requirement has not been met as during development the team member responsible for implementing the GitHub API became unresponsive. With more time this would have been implemented to avoid having to update values manually.
FM7	This requirement has been fully met as we have a bar chart display as part of the user interface that shows what percentage of the budget has been used so far
FM8	This requirement has been fully met as we had 2 users create 2 different projects and not assign each other as part of the team involved with the project. We then asked both users to view their projects and both users could see the project they created but not the project that the other user created
FM9	This requirement has been successfully met as we asked a user to input a priority ordered to-do list. They could see the list and to ensure that the list was stored and fetched correctly from the database we asked the user to logout and log back in and the to-do list was still visible
FM10	This requirement has been successfully met as we had a user input a project and then that user could successfully see a progress bar which showed how much of the budget had been spent, a to-do list etc.
FS1	This requirement was not met as we decided that this requirement was less important than we initially thought. This is because seeing how similar a current project is to a past project does not provide a user with any useful insight that providing a risk rating and change recommendation would not. This perspective was brought to light when we asked many users to review the requirements that had not been met a week before the deadline so that we could make sure that we were prioritising the most useful features.
FS2	This requirement was not met due to time constraints combined with the team member in charge of interacing with the Github API becoming non-responsive. With more time we would implement a function that analyses what percentage of tasks on the To-do list as well as what percentage of Github bugs have been completed. This could then be combined and displayed to the user as an overall progress bar for project completion.
FS3	Unfortunately due to time constraints and a member being unable to work on the project this requirement got delayed and was never completed, though the metrics of communication felt so vital that it is currently implemented as a raw value input and would be one of the first features we would implement given more time.
FS4	This requirement was revised to become more in line with requirment FM5 which was was met. Instead of asking the user to input which metrics contributed the most to the failure of a past project we are able to detect that automatically by calculating how far away a project is from the feasible region as discussed extensively in Section 2.3.2

FS5	We have successfully implemented a risk scale as talked about in Section 4.1.1 which can be further refined as more project data is gathered. The risk is updated over time and its evaluation is the most prominent feature of our model
FC1	Not implemented but alongside communication and getting GitHub API to work this would be a key feature to be worked on in the future. Currently the website is only built for project managers.
FC2	Not implemented but would be a key feature when implementing the mobile type software to allow team members to easily join and contribute to inputting data. It would in itself better communication between the team which in a sense would lower risk. This is highlighted in Section 5.2
FD1	We have successfully avoided this happening, though there is definitely room for improvement with the UI making it even less like a spreadsheet than it currently is. Due to time and resources we didn't have the luxury of hiding everything behind fancy UI so when entering projects and editing metrics it is still quite spreadsheet style, but as per the requirement, when viewing projects there are enough graphicals to justify having met this requirement. To test we had a user use the program and asked if they felt projects were displayed in a useful and digestible manner to which they agreed.

Requirement Number	Evaluation of how well we met the requirement
NFM1	The website was tested by several users with no exposure to software development or computer science in an academic sense and they were able to navigate to every part of the website we asked them to go and could set up a mock project in well under 20 minutes. The design is very minimalistic, as seen in Section 2.2, and the current features are all highly intuitive. Though as being software designed for people with a background in software development the level of intuition wasn't a huge priority.
NFM2	As hosted on AWS the servers can multiple users easily, this was tested by having a few of the team use the website at once, making sure to see that the response time for running the model didn't go up. AWS advertises a 99.99% up-time and there's no reason to doubt this given that it has been up the entire time we have been using it so far.
NFM3	The website is able to be accessed on any device that supports HTML and can be accessed on multiple work stations at the same time if that is ever wanted.
NFM4	Button sizes are large where possible, symbols are used frequently such a house as home page and a sun to change between light and dark mode. Black and white are used in excess to avoid colour blindness issues and WCAG2 were kept in mind when designing anything.
NFM5	Using AWS we were able to have hard limits on users accessing sites, given this number is excessively high. It also allowed us to control how much CPU power each lambda had access to so the part which calculates risk and evaluates past projects have significantly more access to the servers CPU's allowing for response times well under the projected 10 seconds. This was tested by spamming requests to the server and adjusting values for CPU usage until the desired times were reached.
NFS1	Not fulfilled currently as would require continuing on development past this deadline. However the project is currently in a functioning state.

6 Evaluation of the Software Development Process

6.1 Using AWS

The database being scalable has been perfect for this project, it will be able to handle a huge amount of projects and this demo version of the software can be easily extended into a full product later on. By storing both the code and data on AWS it was really easy to integrate the different parts of the project through a library called 'boto3', it had API for 'GET' and 'POST' equivalents and will once again allow for easy integration of any further features. The database can easily be edited through an easy to use UI which allowed for values to be changed for

testing purposes easily and quickly.

AWS lambdas hosted all the code for the back-end, and there exists a built in testing feature that allowed for inputting of data into the database and general simple. It also allowed for quick and dirty testing to be performed as development occurred in a handful of minutes rather than having to compile and run huge amounts of code. This testing feature also allowed some intuitive insight into how the lambdas received and outputted parameters which made creating the API for the front-end much easier. The lambdas took and outputted parameters as 'json', which was desirable as the front-end could easily interpret this being written in Type Script(Java Script).

Using AWS meant the database had to be non-relational which did cause our initial mock ups of the database to require change. This change was welcome though as the tables remained mostly the same and the scalability and speed it provided was favoured. This does however mean there can't be any foreign keys meaning more of the chained updating has to be processed in code rather than a relational language such as SQL.

AWS meant that our code was able to be edited and committed in a similar way to the system on GitHub. Change logs are available for every lambda and tests are shareable so we could collectively work on and edit any piece of code with functioning version history, all proven to be very useful tools. These features meant that we abandoned using GitHub for the back-end as planned as almost all features relevant to us were identical to ones on AWS. There were also preinstalled ways to enable the required 'Scipy' and 'Numpy' packages for our code making the process of moving code into AWS very smooth.

Regarding uploading our codebase for the database, there isn't any code that we had to write to set it up, so although the database exists and we use it, it hasn't got any code attatched to it that we could upload. As it is fully non-relational, essentially a huge hash table with a few additional rules, no structure needs to be defined other than which attribute is the key for a table.

6.2 Methodology & Communication

The team was able to work coherently and efficiently due to the scrum structure alongside weekly meetings, allowing for the team to be up to date on each other's progress at all times. The members working on the back-end used pair-programming to quickly debug code and due to the project manager keeping lines of communication with members working on both sides of the project, any potential integration problems were able to be resolved and spotted quickly. One such example would be that the project manager wrote most of the API code as they had an understanding of both the front-end and back-end, making the implementation process - particularly the integration of the front and back end - extremely smooth.

Discord proved to be a highly effective means of communication for the group, users being able to both talk to individuals while allowing the rest of the group to read what's going on in their own time, as well as talk and code in calls, allowing for steady progress to be made while working from remote locations. We had different text-channels to split up conversation and images relevant to each part of the project. This helped keep conversation in these channels highly professional and allowed us to easily keep track of progress reports in meetings as we could work through each part of the project having a different channel showing the relevant information for each part.

The weekly meetings were vital to keeping members honest and responsible to what they had done in the week and allowed a level of transparency that avoided any serious problems. There is also a level of understanding that can be portrayed much more easily in person regarding complicated topics relating to the model that would've been harder to communicate without in-person contact. Scrum allowed a simple way to assign specific tasks to individuals and split the work load up. Using Jira allowed a single access point to keep track of tasks and although it only allowed us to "officially" assign one member to each part it was still helpful in making sure every task was being attempted by someone for that week. Sub-sectioning the work load into weeks worked really well, given we all have other modules to complete the timing of a week between sprints allowed members to do their work when they had time and anything shorter than a week would've probably start clashing with other things people have to do. We ended up scrapping the Gantt chart as it was feeling identical to the tasks and timeline we had on Jira and simply wasn't being used by members of the team or adding anything of value to our planning process. Moreover, we were flexible in our meeting format for example when one of our team members was unable to make it as a result of train strikes or illness they were able to join the meeting via a discord call and then were put on loudspeaker.

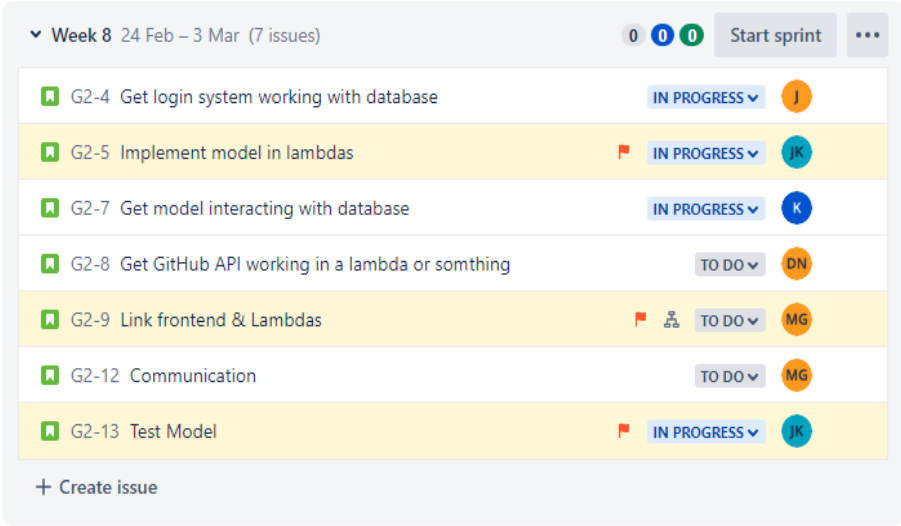

Figure 7: Example of our Jira sprint for week 8

Overall the project manager was able to ensure that we met our final deadline with a fully working product. This was as a result of ensuring that we completed all the smaller tasks that we outlined in each sprint on time by holding everyone accountable to the tasks they were assigned and checking in on progress before the weekly deadline. Additionally, the project manager effectively facilitated group decisions during meetings allowing us to come to a consensus on big decisions such as not completing particular requirements, instead of taking complete control himself. Possible improvements would have been being able to keep in touch with the group member that became non-responsive or ensuring that we completed the final code more than 1 day before the final deadline.

6.3 Did our process affect the quality of our final product?

Our project progressed smoothly, as we imagined, and there have been no issues with most team members; except one who was unable to work on the project due unknown reasons. We believe that the quality of the final project has met the required standard outlined in our requirements. After submitting the Requirement-Analysis and Design documents, we allocated jobs within our team and planned the project schedule well: we held a meeting every Friday to talk about the progress of the assigned work from the previous week and to plan for the week ahead. In addition there were few delays, where present they lasted for at most a week allowing us to quickly get back on track when extra efforts and manpower were allocated to the specific part which was causing delays.

The initial planning stage proved extremely useful as we didn't stray far from the requirements that were laid out. Having two meetings a week early on to discuss exactly what general approach and methodology we were going to use significantly benefited project progression. The group structure splitting back-end and front-end let members of the group specialise from the very beginning, giving our maths model and web design a good level of depth and adequate time was given to developing the needed skills. This did however put a fair bit of importance on our weekly meetings to get everyone up to speed which subsequently made the meetings overrun and take a few hours quite frequently.

References

[1] M. Heričko, A. Živkovič, and I. Rozman, "An approach to optimizing software development team size," *Information Processing Letters*, vol. 108, no. 3, pp. 101–106, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020019008001130

[2] L. Wallace, "The Development of an Instrument to Measure Software Project Risk," Ph.D. dissertation. [Online]. Available: https://www.proquest.com/docview/304527643?fromopenview=true&pq-origsite=gscholar

[3] "Average Software Engineer salary in the U." [Online]. Available: https://www.reed.co.uk/average-salary/average-software-engineer-salary