1 Introduction

Our prototype will aim to fill the need of highlighting risk in projects during development and relay this information in a concise and readable way for managers of projects. It will hopefully be a tool that helps in reducing waste of resources, time and money. The prototype will be a web app allowing data to be inputted into a mathematical model and the derived conclusions from that model being displayed back to the users in a helpful format.

2 Specification Interpretation

We will gear our prototype to be manager-use focused and not accommodate functionality for project members. In the actual product we would create some way to allow project members to input optional data that can aid the mathematical model such as an app or a spreadsheet that automatically feeds data into the website. A project manager will input all data for the prototype. In our requirements the use of "Users" refers to the indicated person to use the program i.e. managers of software engineering projects.

The prototype will be preventative-measure focused, focusing on looking to find areas of most risk and provide warning, rather than be a program that directly provides advice for success. The manager will be able to track projects on a website by viewing metrics defined in the requirements more thoroughly. The prototype will have measures of aspects such as communication between team members and progress towards tasks with upcoming deadlines.

For initial evaluation of risk we will provide a generalised level-based metric e.g. very risky, risky e.t.c. This will be obtained from the relevant information gathered from the inputs the manager has provided. Then we will use a mathematical model that will consider constantly changing values inputted by the team members over time and automatically read data from GitHub API to update this evaluation as the project goes on. This means the evaluation should get more accurate after a few weeks have passed as more data is available for the model to process.

The prototype will highlight areas that it deems under risk to the manager. This should then be interpreted by the manager into what needs to be changed/ fixed to continue having a successful project. Periodically, the prototype will compare similarity between the current project and past projects, highlighting areas of potential risk based on this similarity. This will take quite a few projects before useful feedback can be achieved.

3 Group Structure

3.1 Methodology

We will have a leader based group structure as the given task can be easily split into sub-problems, but due to the large scope and small team size we will operate in a largely informal manner. This will constitute roles implying that the person is responsible for overseeing that section of work but never doing it alone. The main decisions will also be made by group consensus but the PM can have final say on more trivial matters.

Discord will be the main stream of our communication alongside WhatsApp as a more casual channel. We will have meetings at a time all members of the team are happy with twice a week for the planning phase, then once the requirement analysis and planning document are done, we will move to one meeting per week to decide what is next to be completed and any recent complications. This allows us to work on large tasks over a whole week at a time while also keeping ourselves flexible in the planning stage.

Since our development team is unchanging in size and members, an agile approach will work best as working flexibly and together we can avoid any islands of knowledge and progress quickly. We will work to keep the PM informed on all aspects of the project so at least one member has a complete overview of the product.

An adaptation of Scrum was the decided agile methodology due to the factors which benefit it: our customer desires a single product and not smaller release cycles, our scope is large but can be split into sub-problems, deliverables and budget (no budget) are all relatively clear and the majority of the members in our group have project experience. We will have sprints with length of a week allowing for steady progress while maintaining regular meetings to discuss what comes next at every step. We will use Jira to manage sprints and the PM is the Scrum manager.

3.2 Roles

We have decided to take a fairly standard approach to roles with the slight variation of integrating testing into the system architect and the designer for the front end and back end respectively. This is because with such a small team testing can't practically be fully separated from the members working on the product.

Since the business analyst role seems slightly lacking in content from the other roles, we will assign the business analyst to someone with the intention that they have another more primary role as well.

As we are a small team of only 6 members and the scope is quite large we will adopt a 'largely informal structured group'. This means we will have roles assigned so that sections of the project have one member responsible for overseeing them generally, but every member will be part of either the front-end or back-end coding too, as otherwise we won't have enough manpower for the scope.

Role	Assignee	Justification
Project Manager	Jordan	Good communication skills, confidence to step up when needed.
Business Analyst	Denys	Has a business qualification.
Designer	Musab	Past experience with relevant dev tools.
System Architect	Yanlin	Past mathematical experience geared towards our chosen model.
Backend Developers	Joban, Krishi	Both CS students and have adept knowledge of the chosen languages.
Frontend Developers	Denys	Past experience with web-development.

3.3 Planning & Communication Tools

Google Drive: For informally writing up documentation and planning.
Overleaf: For formally writing up documentation and planning.
Discord: Main stream of team communication and organisation.
Whatsapp: Sub stream of more casual communication.
Jira: For managing scrum sprints during the development phase.

4 Requirements

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

Number	Customer Requirements	Developer Requirements
FM1	Users must be able to login to their account and create an account if one does not already exist.	Developers must make a login system which correctly authenticates users, using a username, password system. Don't allow duplicate users by checking if the email address is unique.
FM2	Users must be able to input metrics used to monitor the project.	The program will provide an easy UI to input data, this will be in the form of either the following : input boxes, check boxes, number scrolls.This data will then be stored in a connected database.
FM3	Users must be able to update metrics that are input.	When a customer clicks on a project there will be an option in the form of a button to allow a user to edit metrics of their projects. The changes will then overwrite the existing data for that project in the database. Database integrity must be maintained at all times. (leniency for prototype)
FM4	Users will be able to see the change in the perceived riskiness of a project which changes over time relative to how the project progresses.	After every change to a project metric, the program will recalculate the risk and display a change to the riskiness rating.
FM5	Users must be able to see which metrics pose the greatest risk to the project.	Must calculate areas which are contributing factors to greatest risk from the mathematical model.

FM6	Users should be able to see a section showing the number of bugs for different parts of the project	Interface with GitHub to automatically detect the number of bugs and relate them to the active to-do list.
FM7	Users must be able to see the total amount of money spent so far and total budget through a user-friendly graphical representation.	This is done by querying the database for the respective values and representing the values as bare minimum text , but should be visible as one of the following: pie chart, bar chart, etc.
FM8	Users must only be able to see information about the projects that they are involved with.	The program should keep a store of all the projects and users can be added and removed from the project by the manager.
FM9	Users must be able to input a priority ordered todo list for a specific project they are working on.	The product should have buttons and input fields that can be easily used to create, edit and delete project to-do lists, including a priority scale.
FM10	Users must be able to see basic information about projects such as total budget and money spent so far e.t.c.	Display these values directly from the database as some form of progress bar.
FS1	Users should be able to see how similar a project is to past projects with the metrics causing most similarity highlighted.	Use the model to compare current projects to past ones and calculate which metrics are causing the similarity the most.
FS2	Users should be able to see the overall progress of code relative to well defined tasks and deadlines in the to-do list.	Must be able to interface with GitHub to automatically track the number of commits and link back to the to-do list.
FS3	Users should be able to see how well the team working on a particular project is communicating	Should interpret how “good” streams of communication are using some simple weighted edge graphs, then convert this into a quantifiable metric.
FS4	Users should be able to input if their project was a success or failure, highlighting metrics that contributed more than others that the model may have missed.	Should allow an option to classify if the project was a success or failure, this information can then be used to alter our mathematical model, to give a more accurate analysis of future projects.
FS5	Users should be able to see a riskiness rating on a simple scale.	Will calculate the initial ‘perceived riskiness’ of individual projects and give them a specific rating using a mathematical model.
FC1	Could allow different user account levels which have access to different features e.g. a project manager could access all the data whereas other team members could only input data.	Possibly introduce an application version of the system, or introduce a developer login that has different permissions; such as not directly being able to view sensitive data values but can view overall trends across projects through looking at the models outputs.
FC2	Could allow direct communications from managers to team members via an alert system or some form of group chatting.	Implement a means of users sending update type messages consisting of only text to each other. Managers could alert everyone working on a given project.
FD1	The Customer should not view a project as a list of metrics, similar to that of a spreadsheet.	Developers should ensure only relevant information is portrayed to the user’s and not treat the program as an information dump.

Number	Customer Requirements	Developer Requirements
NFM1	Users must be able to be comfortable using the system within 20 minutes, with minimal to no training.	Make navigation between pages obvious and accessible to both technical and non-technical users, make sure anything non-intuitive has an explanation and keep design minimal to avoid confusion from too many features.

NFM2	The system should be available $\geq 99\%$ of the time.	Ensure the system is comprehensively tested, the system should also allow multiple users to use the system at once.
NFM3	Customers should be able to use the program from different workstations.	Ensure the system is portable (a website) and supports different sized device screens.
NFM4	Product must be accessible to users with a disability.	Ensure the system has good accessibility measures in place to allow people with disabilities to also use the program. Roughly adhere to WCAG 2 standards.
NFM5	Product should give low response times and not require the users to wait longer than 10 seconds per action.	Developers should ensure that the program does not take an extensive amount of time to run, and possibly limit the amount of users that can use the program at once to allow this; preventing DoS attacks.
NFS1	Product should be revisited periodically to add or remove metrics and make sure any bugs are fixed.	Product should be maintained to a functioning level at all times and periodically checked for any needed changes to fundamental features.

5 System Architecture

5.1 Overview

We plan to base our system design on a microservice architecture[1], with a unitary client front-end which accesses independent backend services through an API Gateway.

The client box is designed for the customer, which includes the UI system. The API provides a way for managers and developers to access the functionality of the system. The backend server contains the functions making up the prototype. Microservices are small and independent. Our single, small team of developers can write and maintain the service this way. Due to the independence of each function, an error in any specific part can be quickly found and fixed, not breaking down the entire system.

5.2 Developer Tools & System Modelling

GitHub: As a code respository.

Figma: For front-end UI design.

LucidChart & diagrams.net: For system modelling.

AWS

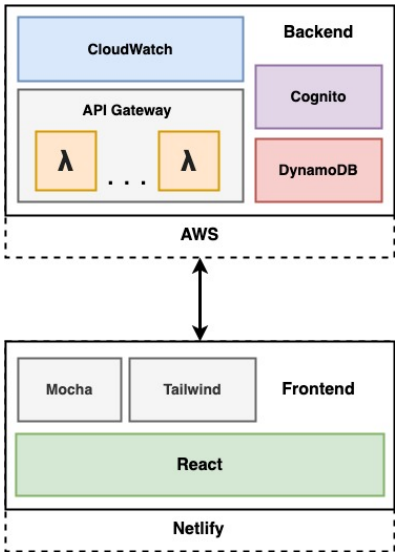
- DynamoDB: For our database to store the needed metrics.
- Cognito: For authentication of users.
- Lambdas
- API Gateway
- CloudWatch: Logs/monitoring for AWS services we use.

Python: To write the backend code (AWS lambdas).

TypeScript, React, Tailwind: To implement the front-end.

Mocha: For front-end unit testing.

Netlify: To host the website for both testing and presentation.



[1] As described in https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices