CS261: Planning & Design Report

Technical Description

Purpose of the Program

Our software will allow a user to track the software development project they are managing and also creates a risk assessment (initial and as the project progresses) which is evaluated using several metrics (hard and soft). Risky areas and suggestions on how to mitigate these risks for any software project will also be highlighted. This will be done through a website which can be accessed by the project leader. This design document illustrates the preliminary planning and design of our software. We will provide a description of processes to give an idea of how our software will work along with justification for our design choices. An outline of the user interface, subsystems interaction, testing plans and also risk assessments will be covered. Naturally, some specifics of the final product may differ from what is described but the general architecture and design should be similar.

Language and Libraries

Our choice of language was primarily motivated by our team's familiarity with Python and web technologies (Javascript, HTML/CSS), the compatibility between these languages, and appropriate libraries. Additionally, Python is a typical choice for ML applications, and we will be using SQLite for its interoperability with web libraries. The libraries are as follows:

- **Flask** - web development library made for Python 3.
- **WTForms** - created with compatibility with Flask in mind, this library makes creating web forms significantly easier if they include specific validators.
- **SQLAlchemy** - database object relational mapper, that is, databases can be expressed as objects in Python 3.
- **Werkzeug** - security library used for things such as password hashing.
- **PyGithub** - allows our project to retrieve GitHub repository data. Used as a metric.
- **sklearn** - Machine learning classification and regression toolkit library.
- **Pyfpdf** - PDF generation library. Will be primarily used to create the project risk and data graph output.
- **Matplotlib** - Library that is used to generate graphs to present data to the manager.

Description of Processes

Database Schema

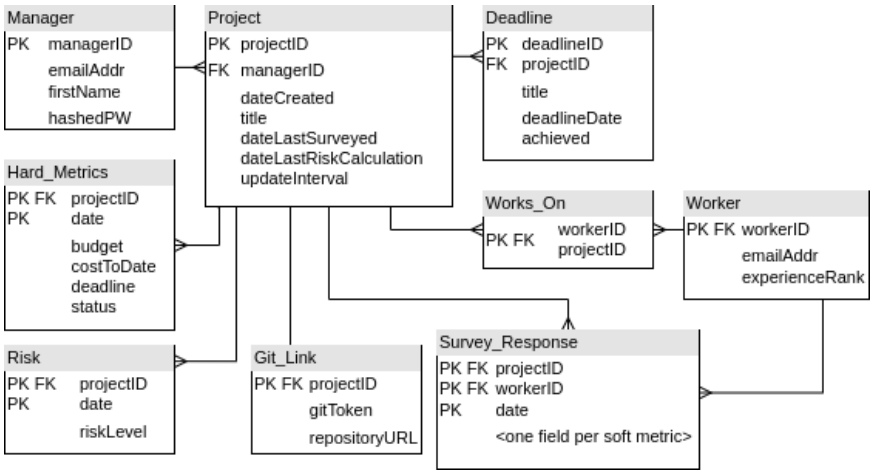


Figure 1: Database Schema

Manager Table The manager table is accountable for keeping manager accounts registered within our system. All passwords are hashed for security.

Project Table
This table is created if a manager creates a new project. Each project is associated with its creator, the manager, thus the foreign key is the managerID. Every project has an updateInterval that stores the frequency of our emailed surveys and the frequency of GitHub pushes within that interval. dateLastSurveyed and dateLastRiskCalculation are stored to enable periodic pushes of the survey and updates to risk calculation respectively.

Hard_Metric Table
The hard metric table stores all initial hard metrics after the creation of a new project. Creating a separate table for hard metrics allows our system to update and create new rows easier when implementing the form within our website. Furthermore, the separation makes it easy to implement new hard metrics in case of an update in the future.

Risk Table
The Risk table stores the calculated risk from our algorithm and associates it with a project, along with its date of calculation to track dateLastRiskCalculation in the Project table.

Git_Link Table
This table was chosen to be separate as gitToken, the GitHub token, can expire and be changed. Separating the table makes pushing the form easier. ProjectID was chosen to be the primary and foreign key to create a one-to-one relationship with Project table.

Deadline Table
This table stores the internal deadlines of a project that the manager creates. Each deadline is represented as a row.

Worker Table
This table stores all the software engineers that are associated with our system. Note that experience can vary from the same software engineer depending on their role. As such, the primary key is the workerID instead of the emailAddr, allowing them to change the experienceRank depending on the project.

Works_On Table
The Works_On table links the Worker Table with the Project Table.

Use-Case, Sequence Diagrams and Interaction

We have designed a Use-Case diagram to present all possible actions of key actors in our system, represented by figure 2 below. Note that the actions available depend not only on the role of the user (Manager/Developer) but also on whether they are signed-in or not.
The interaction of our system is as follows:

Log-in/Sign-up
A manager logs in or signs up using the login/signup form of our website. A signup would create a new row in the table Managers, whereas a login would check for existing credentials in the table. There is no interaction with the algorithm in this functionality.

Project creation
See figure 3.

Editing project data
Editing project data, such as its existing hard metrics, is similarly explained in figure 3. In particular, once hard metrics are POSTed through the website, a new row in the Hard_Metric table is inserted with data. This data is then fetched by the algorithm to re-calculate the risk. After the re-calculation, the new risk is stored in the Risk table and can be displayed in the website at will.

Survey
The survey interaction of our system is split into an if statement. If the software engineer fills the survey for the first time, the website prompts an initial survey. Once the initial survey is filled into the website and stored in the database, the website prompts back to the software engineer with the soft metric survey. Once the soft metric survey data is POSTed, this data is stored in the database under the Survey_Response table. This data is then fed into the algorithm, which re-calculates the risk, stores it in the database, which then can be viewed by the manager. If it is not the first time, the steps explained above repeat, except that they do not fill in the initial survey.

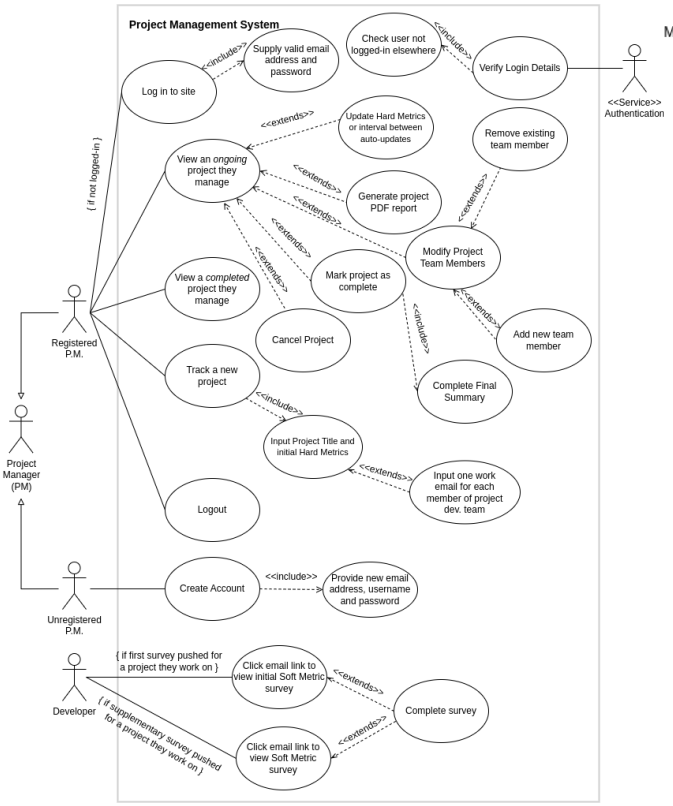


Figure 2: Use-Case diagram of our system

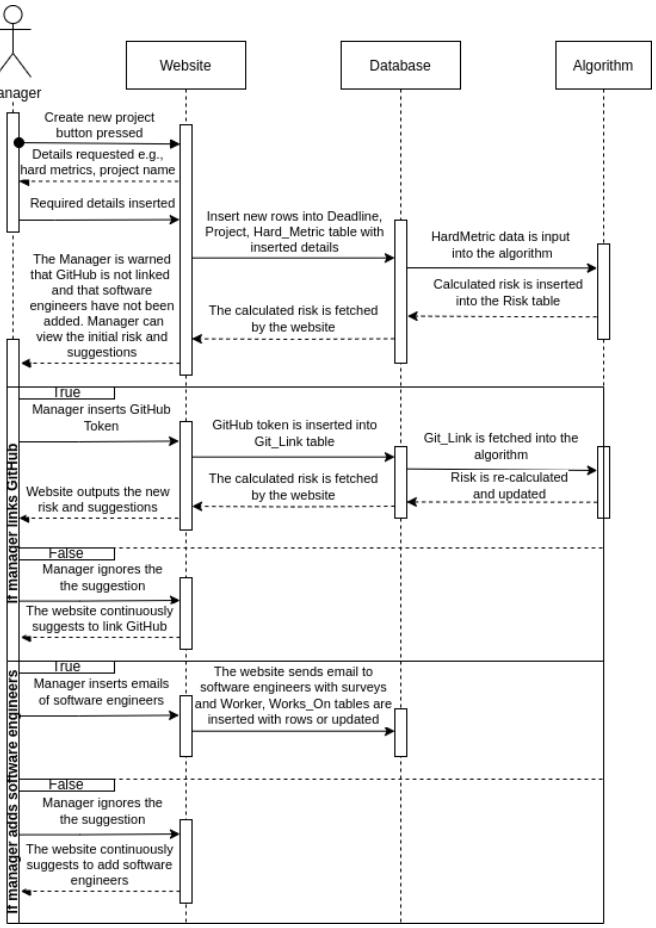


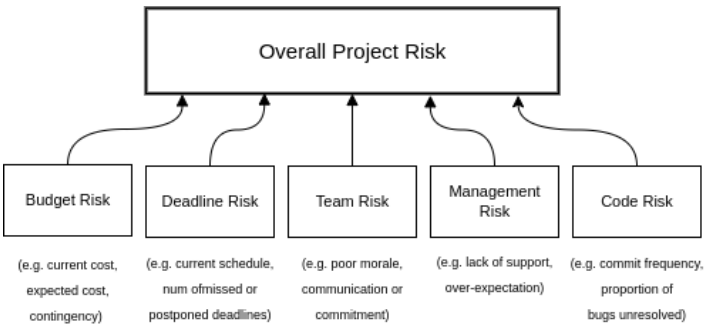
Figure 3: Sequence diagram of creating a project

Risk Assessment Model

The role of the Risk-Assessment Model is to receive a snapshot of the project metrics at a specific point in the development (the independent variables) and to return a value indicating whether the project is likely to be a success or a failure (the dependent variable). Given the complexity of such a relationship, we decided it would be most effective to use a form of Supervised Learning, where the model is trained on labelled data (I.e. a set of projects at various points in their development which have been labelled as ultimately being a success or failure). Furthermore, each project which is tracked to completion/failure using the software provides a new row of training data which can be fed-back into the learning model, to improve the accuracy over-time.

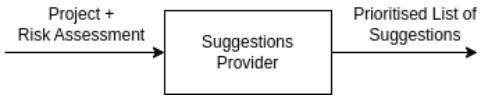
For this problem, there are two main types of Machine Learning model which are applicable: Classification and Regression. Initially, we planned to implement a Classifier such as Support Vector Machines or Decision-Trees in order to obtain a binary (Success/Failure) response, with the confidence of the model representing a measure of the likelihood of the predicted outcome. However, our research found that Logistic Regression models are more appropriate for this problem having previously been applied successfully for similar prediction tasks (Salem, Rekab, & Whittaker, 2004). Specifically, while traditional Linear Regression models produce a continuous output value, Logistic Regression models produce a discrete classification (e.g. 0 or 1, corresponding to success or failure) (Kanade, 2022) which is naturally more suited to the Risk-Assessment Model.

Additionally, to meet requirement **R.4.1**, our system must provide suggestions which can be taken to reduce the risk level, so it is important that the Risk Assessment can be decomposed into components. As such, the model will be divided into five sub-models, each of which provides a risk estimation for a different element of the project (Budget, Deadline, Team, Management and Code). This design offers greater granularity to the Risk-Assessment, so suggestions can be provided according to which component(s) are predicted to fail.



Suggestions

We have decided that suggestions will be provided by a dedicated SuggestionsProvider object, which takes a Project and its Risk Assessment as input and returns a list of suggestions. The actual implementation will be primitive as it will be hard-coded to return suggestions for each of the five categories which fails in the Risk Assessment. Our main reasoning is that because many projects suffer from similar issues, suggestions can be generic enough to apply to multiple projects.



User Interface

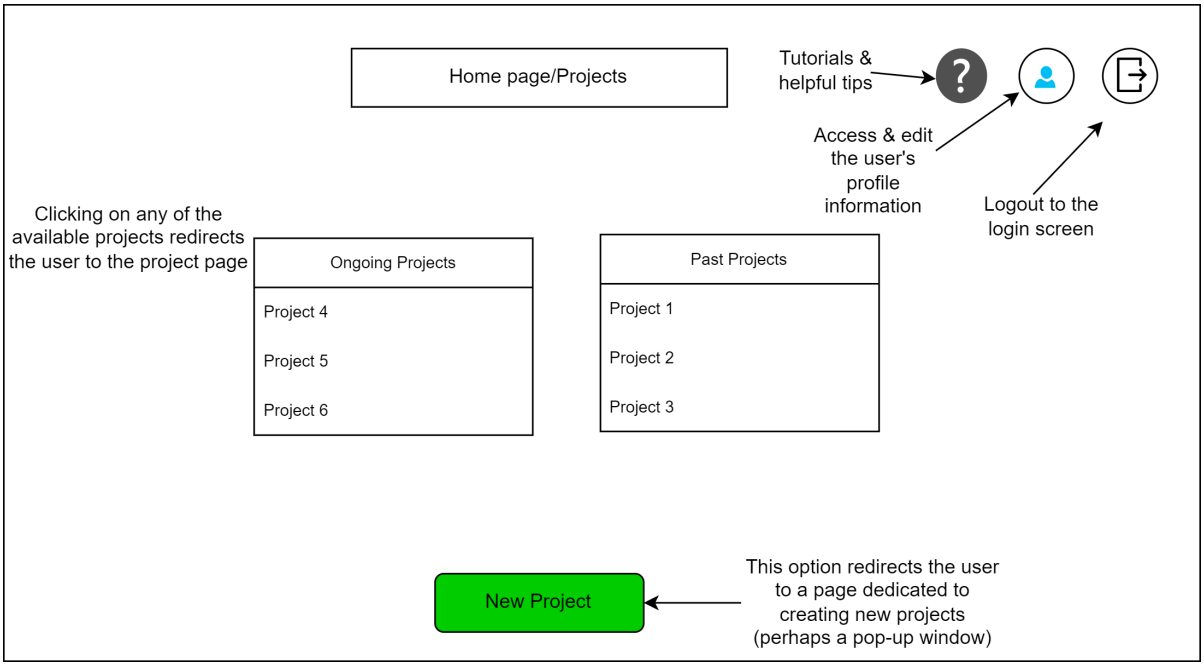


Figure 4: Project Management Interface

This is the design of the home page in which the user may select to view the details of current or past projects

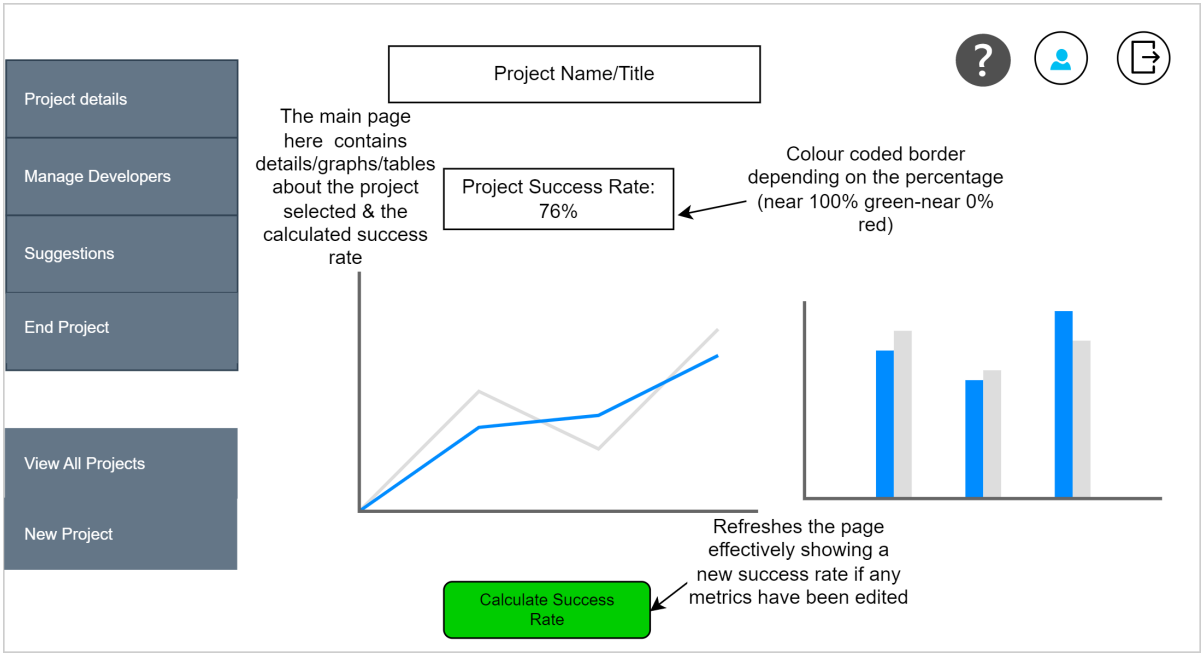


Figure 5: Project Risk and Statistics Page

Upon clicking on any project, the user is redirected to that project's main page.

The menu provides options through which the user may change the project metrics which should affect the output provided. The output is in the form of tables, graphs, success rate percentages and general suggestions. If any of the metrics have been changed (either from "Manage Developers" or "Project Details" sections), the website automatically refreshes showing updated results. The "Completed" projects will have these modification options greyed out and unclickable, so they cannot be interacted with.

UI Design Choices

The website and the user interface have a concise and minimalist design. It will consist of approximately nine different pages some of which will be represented as pop-up pages or lists. The UI is designed to be clear and have a high affordance to the user using tables, lists and menus wherever possible. Two icons (the account management and the log-out icon) can be found consistently on the top right of every page that supports said features. Additionally, at the bottom of each page there is a button of the same colour that indicates the "interact" button, providing a primary action for that page (e.g. to calculate the success rate, to create a new project or to continue to the next page when choosing which project to interact with). The affordance is provided by the simple icons located in any areas that the user may interact with. In case the user requires assistance, a third icon on the top right of any page that may need it, contains an "information/FAQ" clickable option. Lastly, the main use of the website which is to give suggestions and calculate a success rate of a software project is located in a clear and convenient place within the website so as to minimise user interaction when performing simple tasks.

GitHub

To provide an effective analysis of the hard code metrics (commit frequency and bug tracking), it was deemed appropriate to link the user's GitHub repository to our software to access this data. The python library PyGithub contains the necessary functionality to collect information from the user's specified GitHub repository. A user will need to provide a Personal Access Token, with at least full repo permissions and the URL of the repository in order to successfully sync and extract the required data. Tokens can also be made to expire, so whilst the user should provide a token that will last for the project's duration, deadlines may be altered, and a user will need to provide a token once again.

GitHub has a feature known as GitHub Issues where team members can report any bugs or improvements and mark them as completed when necessary. It also tracks user commits, so the frequency of commits can be extracted. Milestones are another feature of GitHub that let users set deadlines for the project; however since these are not mandated and may not be used by a project, it was deemed appropriate to instead collect this information from the user of our software for reliability.

Survey

Survey Questions

We have determined that measuring soft metrics in our software is necessary to provide an accurate estimation of the project failure risk. Since soft metrics in essence are not quantifiable data, we will collect this data via surveys which contain approximately 5 questions with a 1 to 5 scale (where 5 indicates a higher chance of success for all questions, thus standardising the data for the algorithm) to make the metric measurable. The survey will be emailed to all team members to ensure we receive input from all relevant stakeholders.

Initial Survey

The initial survey is unique, gathering data from before the software developers have started working on the project. We also require team members to let us know their expertise when they join a project; this is a hard metric, but we take advantage of a survey being sent out to gather this data.

Some examples of survey questions that may be asked (bold indicates the metric being assessed):

1. I agree with the initial project requirements $\&$ design planned out by the team - **Degree of project planning**
2. I fully understand the goals and objectives of the project that the customer is asking for - **Degree of project planning**
3. Choose the rank which best describes your level of experience as a software developer (**Hard metric - expertise**)
    - Rank 1 - junior software engineer, $1-3$ years, $2+$ languages
    - Rank 5 - senior software engineer, $5+$ years of experience, proficient in most modern languages as well as have other team management skills.

Periodic Survey

On the other hand, standard surveys will be sent at regular intervals or whenever the manager decides, to the development team to obtain metrics related to the current state of the team.

Some examples of the survey questions would be (bold indicates the metric being assessed):

1. I feel that the top management is committed to the success of this project and feel comfortable approaching them with any issues regarding the project - **Top-level management support**
2. I feel committed to the success of this project - **Team commitment**
3. I feel that the team communicates effectively - **Team communication**
4. I am happy and confident working on this project - **Team Morale**

Frequency of Risk Assessment

The risk assessment will be calculated after the initial metrics are provided by the manager and the developers. Throughout the duration of the project, the assessment will be recalculated after every change in the data for the project: such as a budget change, or the periodic surveys being completed. If a change occurs, this recalculation will only happen once at $00:00$ GMT. The user is also able to click a button on the website that recalculates the risk and provides new graphical representation and new suggestions (if a change in the risk is present). Alternatively, if no change occurs or no recalculation happens over a certain period of time, the system automatically updates as well.

The reason such limitations are in-place is to ensure the user does not flood the system with high numbers of requests, which would cause multiple database rows to be created for the same date as well as being demanding on our system.

Visualisation

When viewing an ongoing project, users will be presented with the overall risk of failure as well as several graphs presenting the trends for budget, risk and deadlines as mentioned in the requirements. The graphs will be created using the Python library matplotlib.

We decided that plotting every value would be infeasible for long projects which last over hundreds of days, so the system will instead divide the entire project time-frame (from start to deadline) into a fixed number of intervals. For each interval, the system will then plot the highest and lowest value for the risk over that period. This approach will ensure the overall trend is visible, while limiting the resolution of the graph to avoid overwhelming the user.

At the beginning of the project, the graph will be empty as there is no data to plot, but the time frames will already be decided. If a deadline is moved (which is highlighted in our requirements), then the time frames, and the respective data points will then be recalculated and replotted. This means that a graph that was previously fully-plotted may now be only half full depending on how much the deadline was extended by.

These data visualisation techniques increase the usability of our system and improve the chances that a user will be able to quickly and successfully extract important information about a project. A user is more likely to interact successfully with our website if it presents the risk in a graphical way rather than a text-based way and therefore more likely to come back and use our system again.

Testing

The test plan is laid out according to components of the software (database, user interface etc.) and will be carried out during each sprint cycle ensuring that all components completed during that cycle work as intended.

Initially, we will use preliminary static testing to ensure that the code meets the requirements and to catch any errors. These tests will be performed by other members of the team. Then, we will use Unit-Testing to validate the functionality of specific components and functions, as well as Integration-Testing for interoperability of components (for example, the website login and the database). Given that the majority of the system's back-end will be written in Python, we will be using the pytest library for unit-testing. Pytest is a well-documented (Krekel, 2023) test-suite for constructing and running automated tests and is preferable to unittest for the ease with which new test cases can be defined without needing to define a dedicated Test class.

Testing Plan

Database

Our system's functionality relies on user and project data being stored and retrieved correctly from the persistent storage (i.e. the SQLite database). We can presume, based on the wide usage of SQLite and its documented testing process (SQLite Contributors, 2022) that the database integrity will be maintained both between queries and when database operations are interrupted (for example, if power is lost). Therefore, we must focus our tests on ensuring our own SQL Functions, Procedures and Triggers are valid and robust.

For example, consider the function to retrieve the mean value of the soft metrics for a given project over a specific interval: $getMeanProjectResponses(projectID, startDate, endDate)$. We will Unit-Test this function by simulating the contents of the Survey_Response table:

Table 1: Example Survey_Response Test State

Project ID	Worker ID	Date	Metric 1	Metric 2	Metric 3	Metric 4	Metric 5
11	1	**2020/01/01**	4	3	1	1	1
11	2	2020/01/03	3	3	5	1	4
12	1	2020/01/03	1	4	1	5	2
11	1	**2020/01/08**	2	3	2	4	1

Then, we can assert that the expected result for a call with $(projectID = 11, startDate = 2020/01/01, endDate = 2020/01/07)$ will be a tuple of means: $(3.5, 3.0, 1.0, 3.0, 2.5)$, following the same order as the columns in the table above. Notice that to achieve this result, the function must calculate the mean of only the first two rows; essentially, this test verifies four correctness properties:

1. The mean of each column is calculated correctly
2. Row 1 is *included* (since its date falls on the boundary of the time period)
3. Row 3 is *ignored* (since it applies to Project 12, not 11)
4. Row 4 is *ignored* (since its date falls outside the time period)

Risk Assessment Model

The Machine-Learning model is difficult to test precisely, since it cannot be expected to produce the correct classification every single time. However, we can require that its performance meets a given accuracy threshold; in particular, we can require that the F-Score (Pedregosa et al., 2011), a measure of the overall precision and recall, is at least 80%. Similarly, sklearn provides methods to generate the Confusion Matrix of the model, which indicates the proportion of mis-classified observations for each pair of classes.

Suggestion System

In order to test the Suggestions Provider (SP), we will first need to have fully tested the Risk-Assessment Model (RAM) and validated that its output is appropriate. Then, to test that the Suggestions are selected correctly, we will write a series of Unit-Tests simulating different Risk-Assessment configurations (for example, one project where all categories are successful except for Budget; another project where all are successful except team). For each scenario, the development team will consider the list of suggestions and identify those which are deemed appropriate for the test-case.

Additionally, due to the relationship between the RAM and SP, we will perform Integration-Testing to ensure that both components interact correctly. This will be achieved by designing a series of sample projects and manually identifying appropriate feedback, before feeding each of them into the RAM-SP pipeline one-by-one, and verifying that the returned suggestions are appropriate for the given sample.

Website and UI

Testing the general functionality of the website is a very high level procedure. A team member other than the person who wrote the front end code will attempt to input erroneous data as well as navigate the website in "unorthodox" ways. Some such ways could be to log out of an account and then press the "back" button on the top left of the browser, attempting to log back in that way. Another detail of the website mentioned in requirement **R.1.2**, i.e., logging a user out from an older device when the same user attempts logging in from a new device, can be trivially tested.

Usability testing will also be carried out by non-technical users. These acceptance tests will highlight the ways in which different users may perceive web-page elements. Non-technical users will be asked to complete a series of tasks and later asked to review how they felt using the website. This ensures that the website is built in such a way that is usable and easy to understand for all users despite having different technical capabilities, fulfilling requirement **R.5.1**.

Static UI tests will be performed to ensure that all features work as it is intended. For example, the usage of the "question mark" icon will be tested to ensure that it is able to guide non-technical users to be able to use the website with minimal training and works as intended, ensuring that requirement **R.5.3** is met.

As mentioned in requirement **R.5.4**, we will mainly test the website on Google Chrome version 109.0.5414.119 on Windows 10. However, the website should be compatible with any other browser that supports Bootstrap 5 and its respective Javascript version. Testing this will be a matter of running the website on different browsers (for example, Firefox, Safari, Microsoft Edge) and using the features of the interface that use Bootstrap or interact with JS. Any observations will be well-documented.

We will also be testing how the website renders on different screen sizes such as mobile phones and laptops to check requirement **R.5.5**. Load testing will be carried out where different scenarios are stimulated to check the behaviour of the system under different throughput by the user to ensure requirement **R.5.6** is fulfilled.

GitHub Linkage

To ensure our software receives the correct information regarding the project's code, it is important to check if our software successfully obtains this data (**R.2.2**). By setting up a sample repository, we can perform unit tests between our software and the repository such as:

- After opening nine issues, each with one of the 9 different labels, run the $open\_issues\_count()$ function to check all issues are being detected
- Using $get\_issues(state =' open')$, iterate through all results and display them, comparing them against the issues open in the repository and check they are identical
- Set the repository to Private, and check the provided token still has access to the repository

Process Documentation and Risk Assessment

We have chosen an Agile methodology, inspired by Scrum. Due to the nature of our development, we do not have full access to the client and so we could not fully implement Scrum. We have chosen to enact weekly cycles, each following these steps: sprint planning, the sprint, and the sprint review. An initial product backlog, consisting of the components necessary to the software has been constructed. This will evolve as the project progresses as more information is gathered about how the team works best and any further functionality the software may need. During sprint planning, this backlog will have the most pressing items taken from the stack (for example, establishing the database is crucial as many future components will interact with it) and broken down into sub-tasks that can be feasibly completed in a cycle, such as the production of a schema, or the database triggers. Each task also requires testing so these will also be allocated in the cycle to be designed and implemented. Once the plan is agreed upon, the sprint can begin, in which each team member works to complete their allotted tasks. To ensure progress is made, daily scrum meetings will be held to track everyone's progress and ensure the tasks are completed by the end of the cycle. Once a cycle is finished, all completed tasks can be removed from the product backlog. Finally, the review is when all the team members can demonstrate their work to the rest of the team, and any comments and improvements can be added to the backlog to be addressed in the following cycle. In order to promote the smooth completion of cycles, we have elected a Scrum-Master who will be responsible for leading the sprint meetings (planning, daily and review).

The following is an outline of what each cycle hopes to address, but as each cycle progresses, these events will naturally be broken down further and completed at different rates as some subsystems are more complex than others:

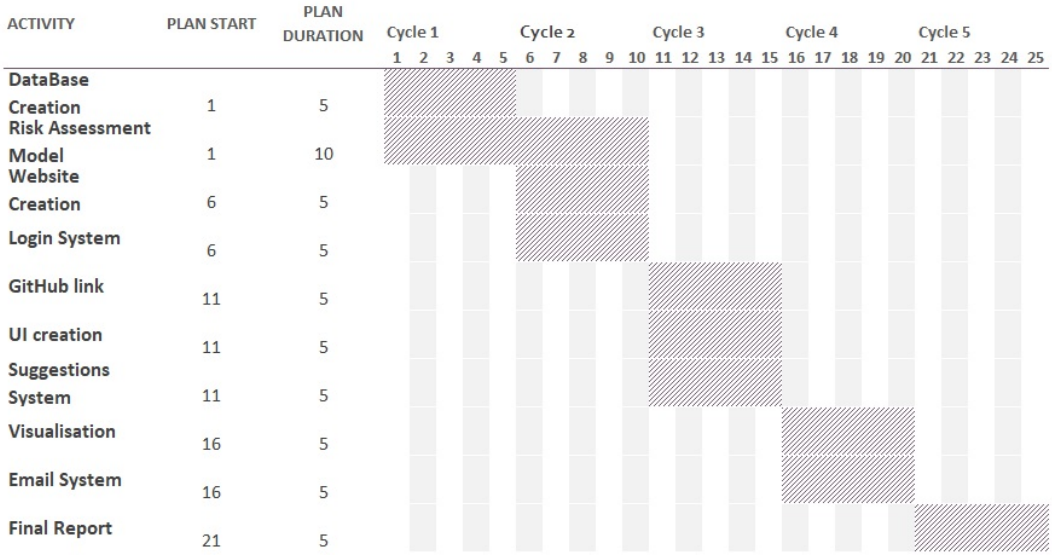


Figure 6: Cycle Gantt Chart

Risks

Risks have been ordered by decreasing potential impact to the development. The inherent risk refers to the threat-level posed when no effort is made to mitigate the danger whereas residual risk is the remaining threat after a treatment has been put into place. We also provide an explanation of how each risk applies to our team (Wilson, 2022).

Risk	Summary	Inherent Risk	Treatment	Residual Risk	Explanation
Size Underestimate or timescale not estimated correctly	Underestimating the complexity of the project and requirements which creates a risk of going over the deadline	High	Ensure a clear timeline is developed over time, with each task broken down into manageable sub sections that are understood by everyone	Medium	Due to our minimal experience with projects of this size, it is hard for us to reliably estimate the complexity and time-scale correctly. However, we have focused on designing a system which can be implemented with the knowledge and tools available to us
Not completing cycle(s) on time	If the end of the cycle(s) is reached without finishing all the tasks out-lined in the cycle timeline	High	Have a member of the team assigned to track the progress of each cycle and ensure all the developers are completing the tasks	Medium	As students, we are not developing the program full-time and therefore can fall behind on the plan we have set out
Tool Underperformance	Libraries or software we intend to use are not performing the way it was expected.	Medium	Perform appropriate re-search before intro-ducing new tools. This should happen in the design stage of the project.	Low	

Specification delays/ change in requirements	The client not presenting the specification in time or requirements are constantly changing.	Low	The agile methodology we have chosen allows us to assess new requirements at the beginning of each cycle and decide if implementing them is feasible.	Low	The client has already presented the specification to us. There is a chance we will change our requirements as we start development however this can be planned for at the beginning of each cycle.
Change in group members	Team members leaving the group and a new person joining who is unaware of the aims of the project.	Low	Re-allocate the roles such that at least two people are responsible for each component.	Low	If any team member leaves, for example in our case if someone drops the module or leaves the university; we will have redundancy in the form of at least one other member able to work on each feature and explain the requirements to a new member.

Overall, we believe that the risk of our project is manageable; we should be able to complete all the tasks, and meet the requirements if we plan for the risks and work towards mitigating them as outlined in the table. During each weekly meeting, we will continue to reassess the risks and ensure that any issues are managed correctly.

References

Kanade, V. (2022, Jun). *Linear regression vs. logistic regression: Understanding 13 key differences.* Retrieved from `https://www.spiceworks.com/tech/artificial-intelligence/articles/linear-regression-vs-logistic-regression/`

Krekel, H. (2023, Jan). *pytest documentation.* Retrieved from `https://buildmedia.readthedocs.org/media/pdf/pytest/latest/pytest.pdf`

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Salem, A. M., Rekab, K., & Whittaker, J. A. (2004). Prediction of software failures through logistic regression. *Inf. Softw. Technol.*, *46*, 1-5.

SQLite Contributors. (2022, May). *How sqlite is tested.* Retrieved from `https://www.sqlite.org/testing.html`

Wilson, F. (2022, Sep). *Inherent vs residual risk: Differences and examples explained - ntask.* Retrieved from `https://www.ntaskmanager.com/blog/inherent-vs-residual-risk/`