CS261 Planning and Design Document

Group 4: Param Bhatia, Matthew Jack-kee, Maddie Amza, Yasmine Mountaser, Rahul Kundi, Brindan Yasodaran

1. Introduction

This document outlines the design and development of a prototype system which closely monitors software projects, assesses potential risks, and provides early warnings should the project be deemed at risk of failure. It serves as a comprehensive guide, highlighting and justifying key technical decisions necessary for the successful creation of the system. This document also analyses the development life cycle of our system including project organisation, development methodologies, testing approaches, and User Interface (UI) design.

2. Technical Description:

This section discusses the technical decisions made to develop our solution to the user specification defined in the Requirements Analysis Report. We justify these decisions with a focus on maximum value for the customer, software development best practices, and timescale feasibility.

2.1. Scope:

It is essential that the project specification, goals, and deliverables are defined concretely to avoid confusion and scope creep during development. The deliverable we are creating is a prototype, rather than a minimum viable product (MVP) or proof of concept. The lines that define these concepts are somewhat blurred, and so for clarity, our goal is to create a working model that shows the feasibility of our chosen concepts and ideas in their ability to accurately fulfil the needs of the customer. The timescale of this prototype leads us to focus on the product's core functionality, with the omission of integration with external systems, and security concerns beyond user verification. Our prototype however is designed to allow for such features to be realistically added in an MVP or final product. The specification of the project is explicitly defined in our Requirements Analysis Report.

2.2. Technology Stack

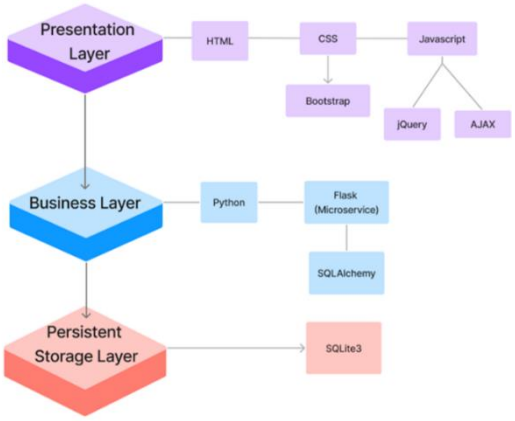

Figure 1: Web Application Architecture

Presentation Layer: Defines the web page format and is responsible for user interaction. This layer communicates with the business layer and displays the retrieved data, from the back-end to the user, via the User Interface. The main components of this layer in our web application are HTML, CSS, and JavaScript. To make the web pages more responsive, we will use Bootstrap, a CSS framework to design a responsive user interface, regardless of display size. This will also decrease the complexity of web design for front-end developers. The jQuery JavaScript library will be used to allow for simplified Document Object Model (DOM) traversal/manipulation and ease of cross-browser compatibility. AJAX is also included inside the presentation layer, as this allows asynchronous tasks to be carried out, resulting in a dynamic website that remains responsive while performing operations, such as calculating risk estimates, on the back-end.

Business Layer: This layer is responsible for accepting user requests from the presentation layer, processing the data, and determining the flow of data. The use of a business layer ensures that the application is secure, and it is easy to maintain the application. In our design, we use Python as it is a high-level language that provides a wide variety of libraries that will allow for complicated calculations to be carried out, such as the use of NumPy and SciPy libraries for Monte Carlo Simulations. Python is a widely used language, which ensures that there exists a plethora of documentation and guides for many use cases. We utilise Flask, a lightweight Python microservice that allows for quick deployment and has a large community. Many developers in our team are familiar with Flask, which allows us to begin development quickly and smoothly. Lastly, we use SQLAlchemy, which is an SQL toolkit, aimed to facilitate communication between Python applications and the SQL database. SQLAlchemy also provides Object-Relational Mapping (ORM), allowing to treat databases as Python objects, rather than through raw SQL queries.

Persistent Storage Layer: This centralised layer is responsible for receiving, storing, and accessing data. This layer is intricately linked to the business layer as the business layer facilitates the ORM ability, while this layer provides long-term data storage and access. This layer also allows for transactions, which is important to ensure data consistency and error handling. For this layer, we will use SQLite, which is suited to our software and needs. SQLite is a serverless database management system, which allows for quick and efficient calls, as well as cross-platform compatibility. Additionally, there is a minimal setup process for SQLite, allowing our developers to start the programming process smoothly and efficiently. SQLite is open source, which means it requires no license, and there is a large variety of documentation and guidance available if needed.

2.3. System Characteristics:

Our prototype is designed with an **Object-Oriented Programming (OOP)** approach. Our design emphasises well-defined **encapsulation** and creating functionality using distinct components. Similar objects and functions are grouped together, resulting in source code with high cohesion and low coupling. These characteristics are in line with **modular programming** techniques, allowing for individual components of the design (such as the "Developer" class or "Project" class) to be switched out or upgraded with minimal change to the remainder of the codebase. This approach also allows future requirements and features, that are developed from changing demand, to be added with little modification to the source code which characterises the **extensibility** of our project. Well-defined functions and classes also ensure **abstraction** of our code to allow the **reuse** of core concepts without having to repeat code, reducing complexity, and increasing efficiency within the design.

Unpredictable or invalid input from end users is of serious concern and risks the stability and security of the system. Front-end validation using HTML and JavaScript, along with back-end validation checks in Python ensure our system is **robust** to malformed requests and bad inputs. Our database will also utilise constraints and rollback methods to ensure the validity and integrity of the database regardless of input. These checks will ensure the **reliability** of the system and guarantee the continued execution of the program.

Our primary **security** concern for this prototype is user validation. This will be achieved via password hashing and ensuring that plain text passwords are never stored in the database. We have designed security using the werkzeug security extension as this provides important functionality such as password hashing and salting, as well as controlling the number of hashing iterations. These methods will significantly reduce the success of attacks on our system using common approaches such as brute force methods or rainbow tables.

To ensure maximum value from the prototype, the design must accurately fulfil the needs of the customer while retaining ease of use for users. This document in parallel with our Requirements Analysis Report ensures that the requirements of the customer are at the forefront of our design decisions. These initial deliverables ensure the **correctness** of the final prototype to the user requirements. Our user interface utilises the flexibility of web app design to display key information required by the user clearly and concisely. The interface is also intuitive to use, highlighting our solution's key features while ensuring the **usability** of our prototype.

We will provide an installer for our application, allowing fast and easy installation of our project across Linux devices, this will improve the usability and portability of our prototype. In addition, we leverage one of the key benefits of web application design, which ensures that users can access the user interface from their device, regardless of the operating system, using a browser. This again improves the ease of use of our design and defines the product's **portability**. It is important to note that our prototype relies on a local backend server, however, our design could easily be extended to utilise a remote host and so we consider these portability design choices as benefits of our prototype.

Finally, the SQLite database ensures ACID properties of transaction (IBM, 2023). The Durability property ensures that data persists even in the event of system failure which characterises the **Fault Tolerance** of our prototype.

2.4. Database Design

Our design uses an SQLite database to store the relevant information for the user. SQLite has been chosen due to its lightweight and portable nature, reducing complexity of use. In addition, SQLite has straightforward connection with Python (our chosen back-end language) which contains the sqlite3 library by default, allowing for easy interaction with the database.

There are two types of users in our design, Project Managers and Developers. The *Project_Managers* relation data fields contain information about the individual, as well as soft skills information which is used in back-end risk calculations. The relation utilises the *project_manager_id* primary key which ensures unique tuples within the relation. The *Developers* relation is similar, using identical fields to the *Project_Managers* and the *developer_id* primary key. Developers have an additional feature in a one-to-many mapping with the *Developer_Strength* relation. This allows any number of predefined strengths to be stored about a developer, the *strength_id* primary key and *developer_id* foreign key ensures unique tuples and the one-to-many relation between *Developers* and *Developer_strength*, respectively. A developer's strengths are compared against a project's requirements during risk calculation.

The Project relation stores information about the project, the tuples are kept unique with the *project_id* primary key. The Project relation has a one-to-many mapping with the *Project_Managers* relation, which represents how one manager can handle multiple projects. The Project relation stores the current status (complete, ongoing, etc.) about the project, and so the database stores information about both present and past projects. This is key information used to identify the cohesion factor between members of the team, and how their strengths align with the requirements of the project.

There is a one-to-one mapping between a Project and its respective *Project_Risk*. The *Project_Risk* relation stores calculated risk values concerning the related project and has a one-to-many mapping with both the *Time_Component* and *Cost_Component* relations. These one-to-many mappings (enabled by the *project_risk_id* foreign keys) allow the project manager to input any number of cost and time factors which culminate in the total cost of the project and allow this data to be dissected and factored into risk calculation.

The *Developer_Project* relation has a many-to-one relationship with both *Developers* and *Projects*. This relation stores the critical information about what projects a developer has worked on, as well as what developers are working on a particular project, in one relation. This information is used to keep track of current developer projects and allows analysis of how effective a developer has been on previous projects. The *developer_id* and *project_id* foreign keys in unison ensure unique tuples within the relation; the addition of a primary key would essentially be wasted memory.

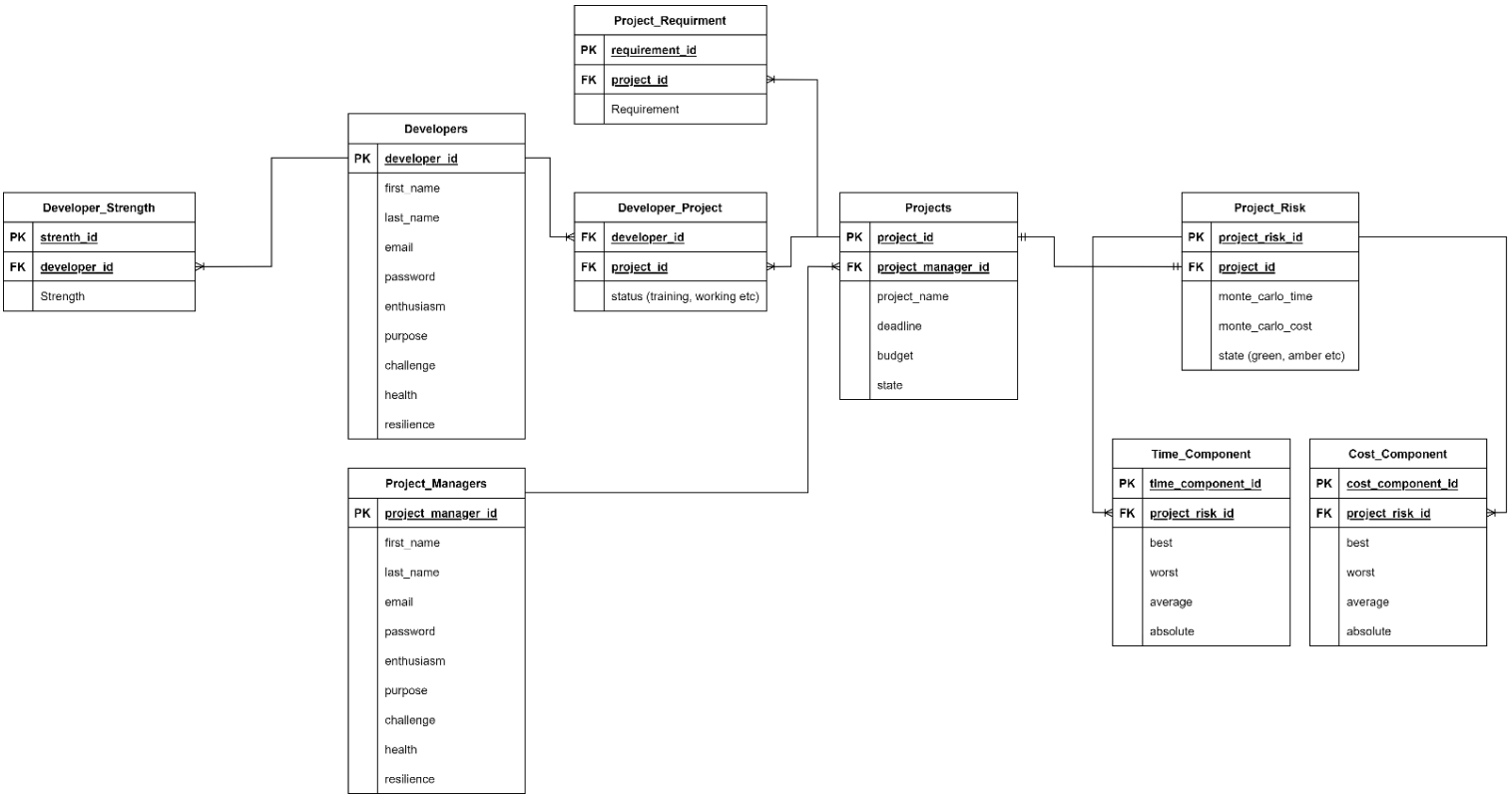

Figure 2: Entity Relation Diagram representation of our SQL Database

2.5. Object Structure

Our design utilises OOP to fulfil the user requirements while benefiting from the advantages discussed in Section 2.3. Encapsulation ensures modular programming which allows for sections of the code to be worked on independently and concurrently by different members of the team, making development as efficient as possible. Both the Developer and Project Manager classes extend the User abstract class. This use of inheritance ensures succinct code and allows the key User class attributes and methods to be required by both of its children. This same approach is taken for the project risk components, again ensuring the correct functionality of the children.

We utilise a Unified Modelling Language (UML) diagram (Fig. 3 and Fig. 4) to visualise these classes and use aggregation and composition to highlight key features, e.g. a Developer conceptually exists without a Project, or that a Developer's soft skills can only exist while the Developer object exists.

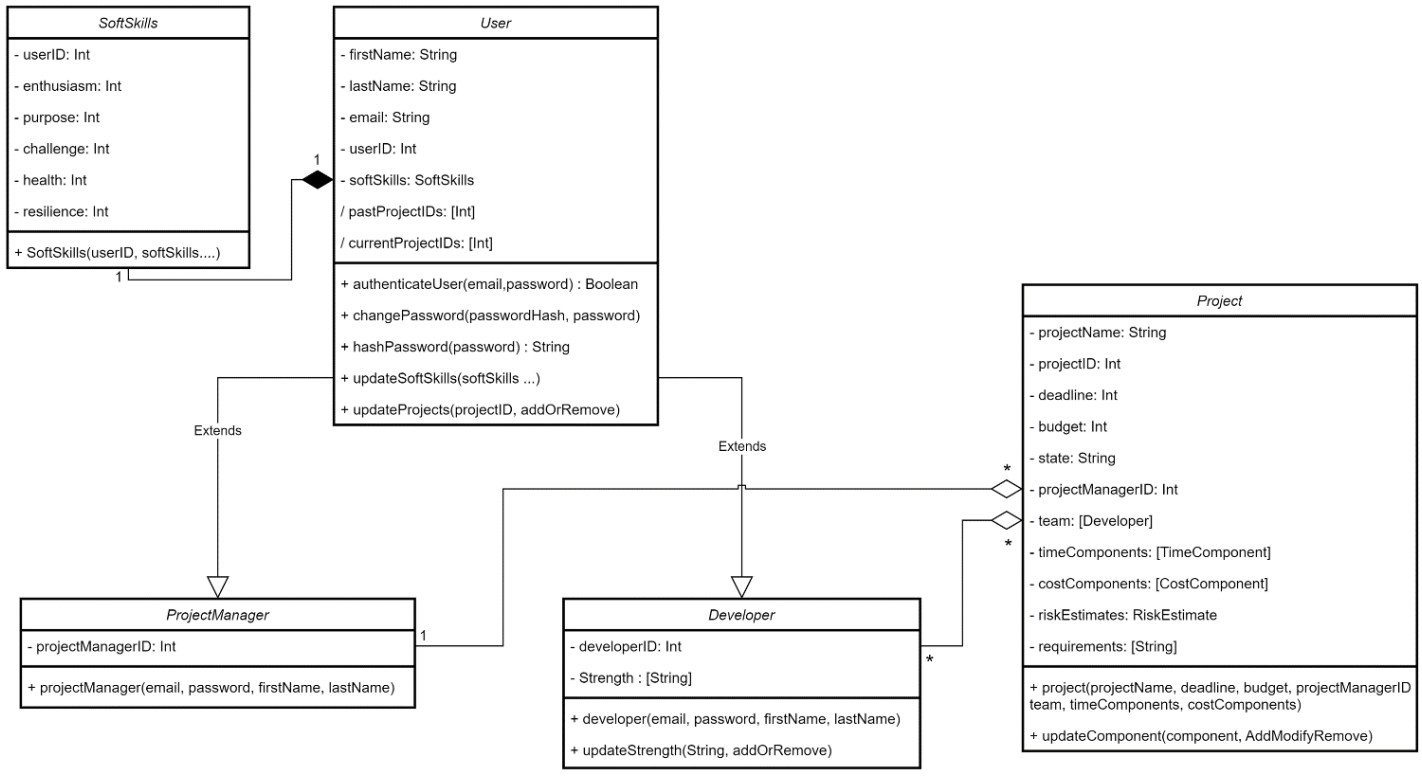

Figure 3: UML visualisation of the User classes used in our design

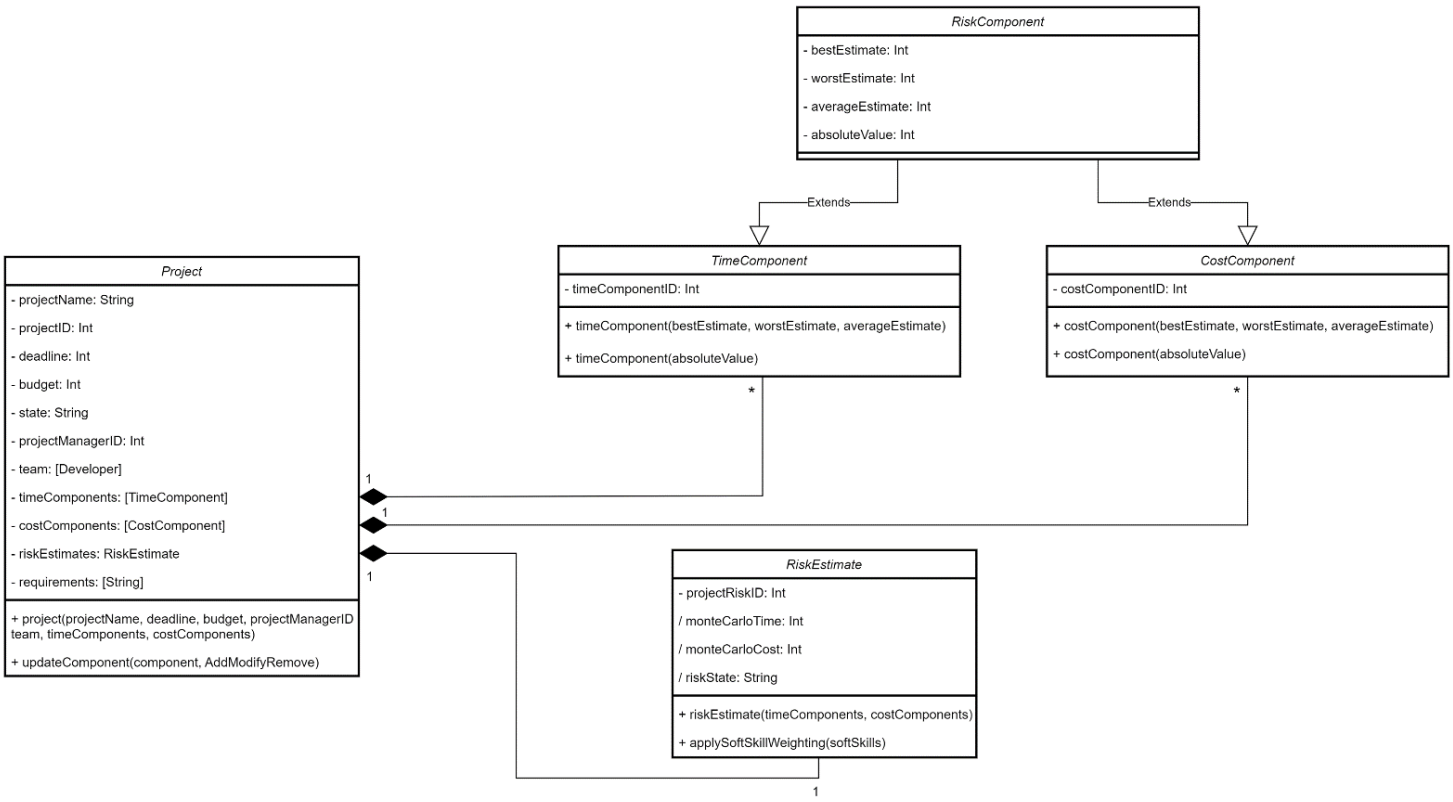

Figure 4: UML visualisation of the Risk classes used in our design (Continued from Figure 3)

2.6. Risk Computation Function

Figure 5: Flowchart depicting the calculation of risk

Fig. 5 displays the functionality of the main risk probability function. This function will conduct data analysis on many diverse sources, which will run concurrently. The first source of data is fetched by asking the user for their best-case, worst-case, and expected budget constraints, which will be used to run a Monte Carlo simulation.

The Monte Carlo simulations will utilise the NumPy and SciPy libraries and will return risk probabilities based on the estimates for budget and deadline constraints. These estimations will be used to create Gaussian Distributions for each component (time and cost) of the project and then generate random values from these distributions. The values chosen for a particular project are then aggregated

to represent a simulation of that project at completion. Components given an absolute value are added to the aggregated total with no need for a distribution to be created. We assume the state of one component is independent of the others e.g. a component finishing much later than expected does not impact the following component's completion time.

The Monte Carlo approach relies on a high number of simulations and so multithreading will be utilised to make the estimation as time efficient as possible. We believe this approach is feasible as much of NumPy operates outside of the Python Global Interpreter Lock, making multithreading a viable approach for this task in Python. This also highlights the importance of keeping the application responsive, as user interaction rates could fall while waiting for these simulations.

During project creation, we will ask the Project Manager to enter a list of 'Developers' that form the team for the specific project. This will then be used to check the compatibility of team members, by looking at past projects. Data on a developer's past projects and teams can be extracted from the SQLite database. This allows us to identify groups of people who potentially do not work well together, e.g. if a subset of the developers currently on the team have worked together on multiple failed projects in the past, we could deduce that they are not compatible, which would increase the risk probability.

The project manager is also prompted to include project requirements, for example, knowledge of Java or skills with SQL. The strengths of Developers within the team are then cross-referenced with the requirements of the project, a higher percentage of Developers with skills appropriate to the project increases the likelihood of success in our estimates.

When the Project Manager is creating a new project, we will also ask them to link their code repository, which we can track to identify changes in risk probability. This will be done through a series of metrics tracked by the repository log. These metrics include, but are not limited to:

$$\frac{Number\ of\ errors\ in\ the\ last\ 24\ hours}{Number\ of\ errors\ in\ the\ last\ 7\ days} * \frac{Number\ of\ total\ days}{Number\ of\ days\ left}$$

We will also use metrics from the soft skills questionnaire provided to all users, which will return a weighting for the risk probabilities calculated in Fig. 5. Finally, we will assign weightings to all these risk probabilities according to their importance to a project, which we identify through qualitative market research. This resulting risk probability (R_p) will be assigned to a colour representing the risk.

Green: $0 \leq R_p \leq 30$;	Amber: $31 \leq R_p \leq 60$	Red: $61 \leq R_p \leq 100$

These spreads are quite wide, especially the spread for displaying a 'High' risk probability (Red) because the model we are creating has a large uncertainty, which means we should stay on the conservative estimates for risk quantification. The uncertainty arises from the lack of training data for past projects, which we would need for fine-tuning the weightings mentioned above. Having conservative estimates is important as it is better that the project manager stays alert even if the risk is not too high, rather than the Project Manager assuming things are going well, when, in reality, the quantified risk may not be reflective of real-world, out-of-scope risks.

2.7. Web Application Page Hierarchy

The web application implements our user interface and has been designed to maximise **usability**, **confidentiality**, and **security**. The user is initially welcomed with a login page and can access the option to create a new user, which will also define the account type. To maximise security, on the front-end, inputted passwords are obscured with "*" visually and use POST requests, while on the backend, password hashing and salting ensure plain text passwords are never stored on the system.

Once logged in, depending on account type, the user will be displayed with either the Project Manager Homepage (Fig. 6) or Developer Homepage (Fig. 7). This distinction ensures additional, and confidential information (highlighted in purple in Fig. 6) can only be viewed by the project manager. This confidentiality ensures potentially high-risk estimates about a project are not seen by developers, which we argue could be a detriment to morale and work ethic. Our prototype utilises the Session extension for the Flask library which ensures this homepage, and the information it contains, can only be accessed if verified and logged in as a project manager.

As a project manager, the homepage displays all current projects owned by this individual as well as useful summaries of each project. This interface ensures that even at a glance, the project manager can gauge which projects are at the most risk. The More Info buttons access a new web page, which allows the project manager to see a more insightful breakdown of current metrics, as well as graph visualisations and suggestions made by our system about how best to decrease risk in the project. The project manager will also have the option to update project parameters using the Update Project Information page. Here the project manager can change or add any number of risk components, the project estimates will not update until the user commits the updates with the Update Estimates button which allows the user to check inputs before the system updates. From the homepage, the project manager can also access the Create New Project page which allows the project manager to create new projects which are then bound to them on the system.

The Developer Homepage provides basic information about current projects, including work to be done and the current team. The developer will have the option to add relevant skills via the Add Skills page, here the user can select skills from a variety of predefined talents. This reduces the need for direct input from the user which reduces the risk of injection attacks.

Both types of users have the option to take the Soft Skills page survey. This can be accessed from the Update Soft Skills button on the homepage. Sliders are used on this page to create an intuitive display and again reduce the possibility of injection attacks. Both types of users can also return to their respective homepage from any web page, improving the navigability of our design. Once logged in a user can return to their respective homepage or log out (return to login screen and end current session) from any page, and these interactions are omitted from Fig. 6 and Fig. 7 for brevity.

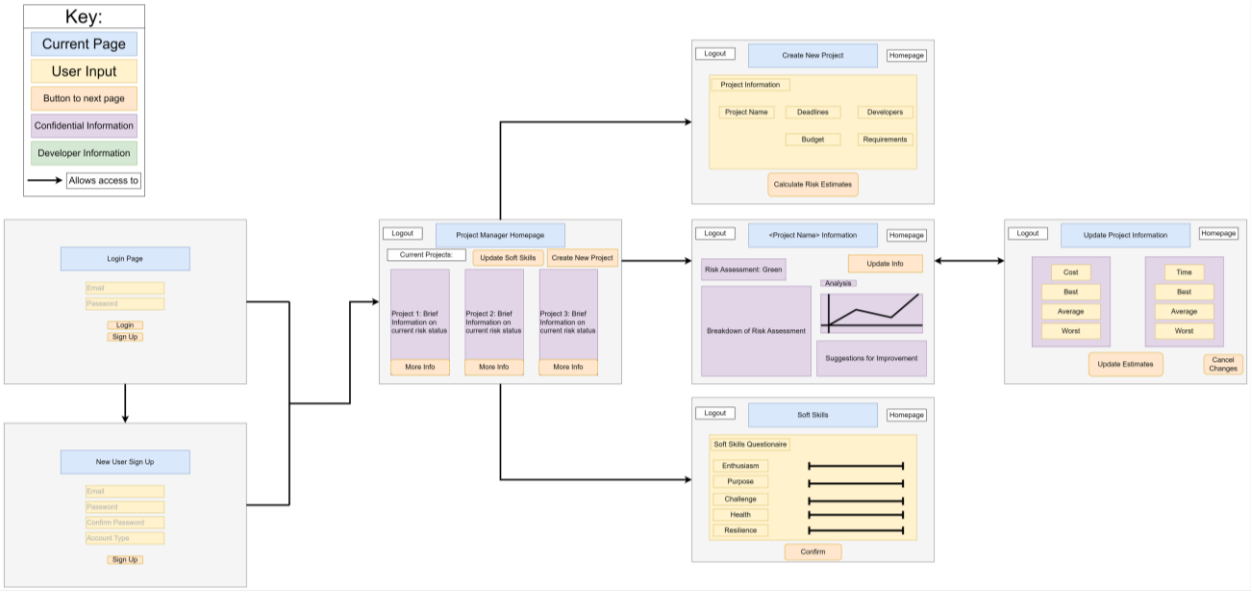

Figure 6. Visualisation of Project Manager Web Page Hierarchy

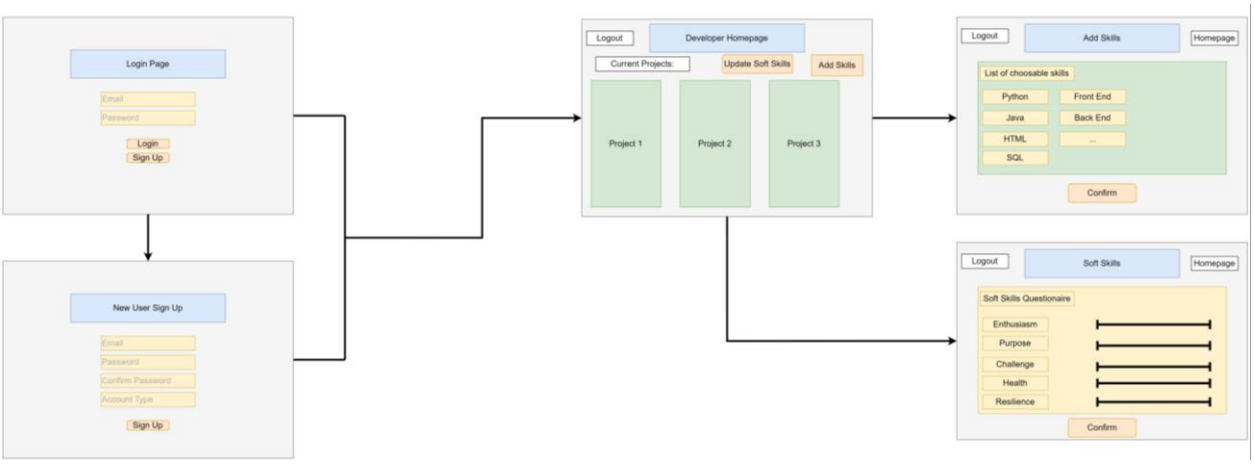

Figure 7. Visualisation of Developer Web Page Hierarchy

3. Process Documentation and Planning

3.1. Development Methodology

The prototype will be developed using an Agile methodology based on the Scrum model (Schwaber & Sutherland, 2020). The project will consist of a planning phase and multiple one-week sprints. Our Scrum Master is responsible for managing the team and ensuring that the workflow is running smoothly. Further details on team roles are described in the table in our Requirements Analysis Report.

The Scrum methodology has been chosen for the management of this project due to its flexibility and ability to align all team members toward common objectives. This approach is better suited than the traditional Waterfall method, especially for small projects that require the ability to make rapid changes (CS261, 2023). To streamline project management, Notion will be utilised as a central platform to maintain a Scrum board, track deadlines, and facilitate collaboration. This will enhance accessibility and communication, allowing all project-related information to be centralised in one place.

At the start of each meeting, the team will convene to prioritise tasks for the upcoming sprint. A Gantt chart (Fig. 9) will serve as a reference for task priority, but the task list may be adjusted based on the project timeline. Stand-up meetings will be conducted to assess progress and address any obstacles. The sprints will culminate in a sprint review, where developers will showcase their completed work, followed by a sprint retrospective to identify opportunities for improvement.

The Agile methodology is selected for this project as it offers efficient solutions for short-term projects (Verheyen, 2019). It enables quick product development through concurrent implementation and testing, maximising the efficiency of production. Additionally, its flexible nature accommodates any unexpected delays, especially when applied within a team with limited experience in the development life cycle. With this said, it remains crucial to have a well-structured plan in place before starting with Agile, as deadlines for deliverables are

fixed and require timely completion (Cohn, 2005). The nature of this project also results in minimal interaction with the client, Deutsche Bank, which prevents changing requirements and so it is likely the project stays aligned with the initial plan. By implementing the Scrum framework, the development team can deliver a functioning product incrementally, respond to changes in requirements if necessary, and continuously improve the development process (Schwaber & Sutherland, 2020).

3.2. Prioritisation Matrix

The matrix (Fig. 8) defines the tasks that need to be completed and the order in which to prioritise these tasks. The values in this matrix map directly to the design requirements in the Requirements Analysis Report, which is an ideal fit for defining tasks to be completed. Urgency is defined as tasks that would require a longer time, and therefore are likely to be time-consuming. Importance is defined as features that are key to the strategic purpose of the software product (Wang et al., 2010). This matrix is essential for our project as it prioritises the next steps if there is a delay or incomplete work from a team member. This allows us to reassign work in such a manner that the core functionality of the system is completed first. This matrix also helps us to anticipate which tasks will take more time, and hence, need more people assigned to those tasks, this helps in making the project efficient and streamlined.

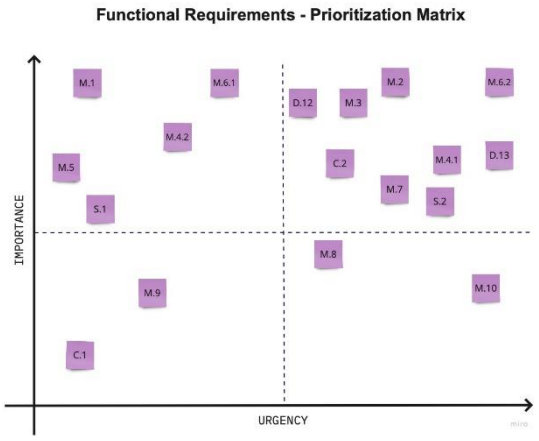

Figure 8: The prioritisation matrix

3.3. Schedule of Project

To outline the tasks needed to complete the final risk management system we have implemented a Gantt Chart. The timeline lists the features of the system that will be delivered in each weekly sprint, divided by the team responsible for them (Cohn & Ford, 2003), each colour essentially serves as clarity for the subsections of tasks. The final product documentation will be a collaboration by the entire team. As we are using the Scrum methodology, the order of tasks may change weekly to keep the project on track. The timeline serves as a reminder that the final report should be written incrementally as the project progresses.

The agile approach we have chosen allows us to implement and update timelines for tasks as needed. In this way, the team can adjust and respond quickly, integrating new priorities into the timeline when needed.

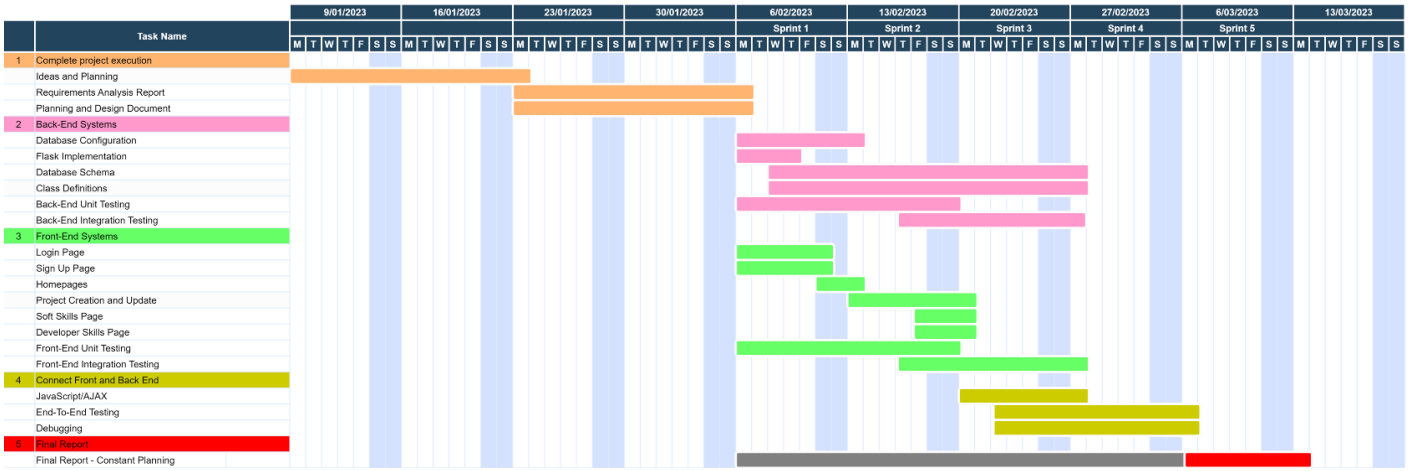

Figure 9: Gantt Chart visualising Sprint Cycles and Development Goals

3.4. Risk Assessment

Risk	Likelihood	Impact	Precaution	Solution
Loss of code	Low	High	Use cloud-based version control, e.g. GitHub, and store copies of code locally.	Revert to the latest copy of files and ensure lost work can be replicated.
Team member does not complete work	Medium	Medium	Ensure at least two people can complete each task, to reduce dependency.	Reassign work and ensure deadlines are met.
Code does not pass tests	High	Low	Ensure the code is written to industry standards.	Additional member allocated to help fixing the code.

Project taking longer than anticipated	Medium	Medium	Ensure members know their responsibilities and allocate time to account for potential delays.	Re-assign workloads to leverage members skills and capabilities.
Code infringes copyright.	Low	High	Ensure the use of libraries is in accordance with license terms.	Use alternative software to ensure it does not infringe copyright.

Table 1: Risk Assessment

4. Testing

4.1. Unit and Integrated Testing

We strive to adhere to a test-driven development framework, and so testing will play a critical role in our development life cycle. To ensure that the required system attributes have been fulfilled, unit tests will be developed for each one of the test cases. These tests will be repeated frequently, to guarantee that the code remains robust throughout the development process.

All tests will be implemented automatically whenever a pull request is submitted to the Git repository to ensure the code's robustness. The code can be merged only if the tests pass, which applies a continuous integration/ continuous deployment (CI/CD) approach (Duval et al., 2007).

There is no singular testing software that can assist with all the technologies in our prototype, and so we select a collection of testing tools (as discussed below) to create a rigorous toolkit which encompasses every component of our system. These testing methodologies enable unit, integration and end-to-end testing of our design.

4.2. Back-End Testing

Robust back-end software is essential to our system, as it provides key operations, such as calculating risk estimates and communicating with the SQLite database. To ensure our Python (Flask) framework is reliable, we use available literature (e.g., Arbuckle, 2010), to ensure thorough testing. We will also make use of Pylint throughout the development process to enforce a strong coding standard (pypi.org, 2023). Pylint also offers refactoring suggestions, which will help ensure our system implementation is logical and well-structured.

We will use Pytest to assist in our back-end unit and integration testing. The testing framework allows for detailed descriptions of error cases and Pytest-cov extension allows for test coverage reports.

As part of the back-end unit and integration testing, the following features will be explicitly tested for; Flask routes: Verify that Flask routes are functioning properly and delivering anticipated results, such as HTTP status codes and displayed templates (Grinberg, 2021). Python functions: Verify that Python functions, such as data processing or computations, are performing as expected (Arbuckle, 2010).

4.3. Database Testing

Our SQLite database must always be in a valid state and responsive during program execution. To ensure this, we employ unit and integration testing to validate ACID properties of our design (IBM, 2023):

Atomicity – If any part of a transaction fails, the entire transaction fails.

Consistency – The database will always be valid after a transaction, regardless of input.

Isolation – The result of multiple transactions in parallel is consistent with the output of the requests done in serial.

Durability – Once a transaction is committed, external factors cannot change the state of the database.

4.4. Front-End Testing

We will implement front-end testing to guarantee our web app is responsive, intuitive and user friendly. We use literature such as Zhongyuan, 2020 as a base to implement our testing framework. Form validation is also tested to verify that only valid inputs are sent to the back-end, adding a layer of security to the system.

We will use Selenium to help automate front-end testing, ensuring appropriate functionality in all parts of the UI design. Jest will also be used for testing as it supports the JavaScript framework approach we have taken (Flaviocopes, 2020).

4.5. Integration testing

Integrity checks are performed during integration testing to make that the system's components interact correctly and are stable. This step is typically taken later in the development process and is crucial for finding any compatibility issues or challenges with the transfer of data and information between components (coupling based approach as discussed by Jin & Offutt, 1999). It is crucial to ensure that the front-end and back-end components are interacting as intended and that data is being appropriately sent between them.

4.6. End-to-End testing

Google created the Node.js library puppeteer, which offers a high-level DevTools Protocol API for controlling headless Chrome. It is a helpful tool for both front-end and back-end testing as it can be used to automate a range of activities, including testing web pages e.g. clicking buttons, which proves useful when it comes to running end-to-end tests. We will also verify that AJAX requests are connecting with the server successfully and providing the expected results (Zhongyuan, 2020).

4.7. Test Cases:

Table 2 details how we are checking the functionality of our system based on the functional requirement table in the Requirements Analysis Report.

Case	Test Objective	Test Input	Expected Output
M1	Validate login functionality for Project Manager (PM) and Developers	Entering both valid and invalid emails and passwords	Login page appropriately allows or denies access
M2	Verify that the creation of a project on the system is successful	Inputs that create valid and invalid projects	If valid, project name is seen on the homepage and current status and risks can be viewed by PM. Else, PM notified of unsuccessful project creation.
M3	Verify different projects can be seen for different team members	Valid projects with different team members	Developers can see (and only see) assigned projects on their dashboard
M4	Verify that PM and PM only can give information on overall project status	Project status information input by PM and attempted input from Developer accounts	PM is able to update projects at will, while developers are unable to access or modify this information
M5	Verify that developers can provide soft metric information through soft skills questionnaire from dashboard	Fill in the questionnaire through provided slider input	Questionnaire responses appropriately stored in database
M6	Verify that PM dashboard shows initial evaluation of risk	Initial overall project status (by PM) and soft metrics (developers)	Red, amber and green traffic light graphic to denote riskiness displayed on dashboard
M7	Verify that PM dashboard updates risk evaluation after project status or soft metrics	Updated project status information is supplied. Soft metrics are provided again.	Red, amber and green traffic light graphic will change or remain the same accordingly
M8	Verify that PM dashboard updates risk evaluation periodically based on metrics computed from the code base	Input is provided automatically by linking the system to the code repository	After each specified period, the traffic light graphic will change or remain the same accordingly
M9, S1	Verify that PM has access to information on elevated areas of risk and suggestions to tackle these	Project status, soft metrics and changes in the code base are input	PM will have access to improvement suggestions
M10	Verify that manager/developer status is correctly tracked	Developer and Project Manager accounts will be registered to the system	System appropriately assigns accounts based on account type
S2	Verify that the system accounts for past chemistry of team members to produce a more accurate risk score	Successful and unsuccessful projects inputted with teams of developers	Higher risk weightings are assigned to teams who have previously worked on unsuccessful projects

Table 2: Test cases and their expected outcomes

5. Design

5.1. Use Case Diagram

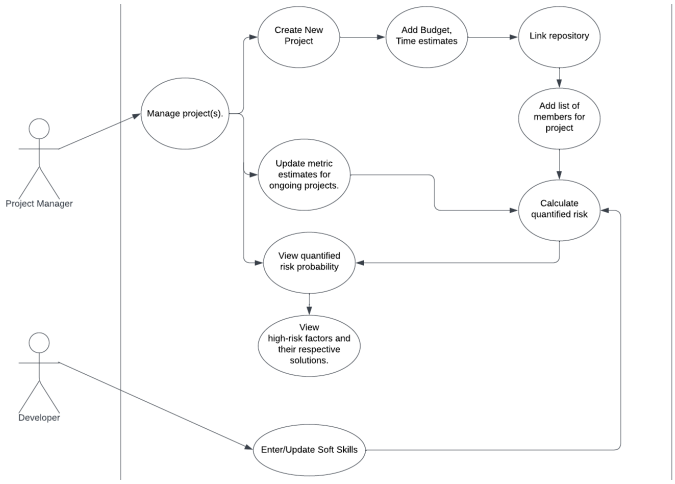

Figure 10: Use Case Diagram for Project Managers and Developers

The Use Case Diagram (Fig. 10) represents all the actions that the Project Manager and Developer can perform. The Project Manager has access to the bulk of the functionality, which includes creating a new project, updating project metrics (budget and time), and viewing the overall risk probability along with possible solutions to reduce risk. The Developers will only have access to the portion of the web application where they can enter information on their soft skills and strengths, as they should not be able to see the overall risk status, as listed in the requirements.

5.2. Sequence Diagram

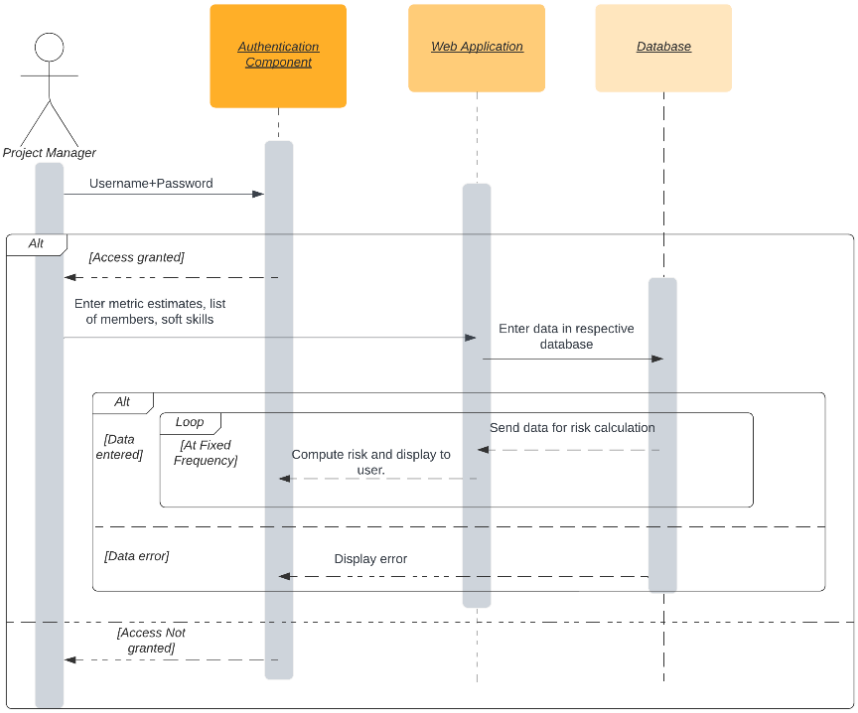

Figure 11: Sequence Diagram for creating a new project

In the sequence diagram (Fig. 11), we can see the high-level interaction between active structures in the system over time. In this example, we show how a Project Manager can create a new project, and what metrics are required to calculate a comprehensive risk probability. The risk is calculated at a fixed frequency, according to the needs of the Project Manager.

5.3. Web Page Design

Our design closely implements 'Gestalt Laws of Perceptual Organisation' (Gordon, 2020). A well-designed Human-Computer Interaction (HCI) can help increase user engagement and interaction with a website, leading to increased user satisfaction and an increase in return of visits. Factors worth considering include:

Proximity: Place related elements next to one another to imply a connection between them. For instance, put related links or buttons together to make their relationship clear to the user.

Similarity: To group related items together, use visual cues like colour, shape, or texture that are like one another. This aids the user in quickly identifying and comprehending the connections between various components on the page.

Additionally, our approach is aligned with Norman's Human Action Cycle (Batterbee, 2020). The user can create mental maps of the interface by, for instance, designating distinct colours for certain sections, which will allow them to locate what they're looking for easily and efficiently. It can also be useful to give the user explicit feedback by altering the colour of the interface, which results in an HCI with accessibility and usability at the forefront of its design. Three key section concepts are shown below in Fig. 12.

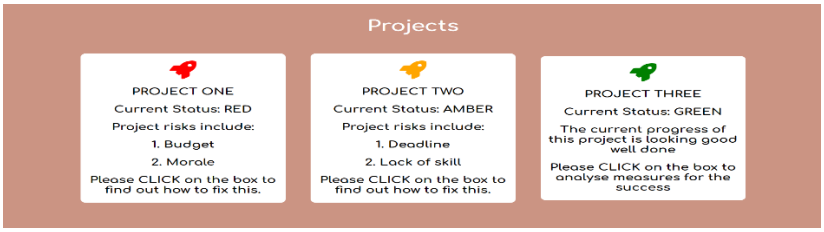

Figure 12: Each template will represent different projects tracking

Bibliography:

Arbuckle, D. (2010) *Python Testing: Beginner's Guide.* Mumbai: Packt Publishing

Archbold, J. (2023) *Software Engineering Topic 2: Software Development Methodologies 2*

AWhetter et al. (2023) *Pylint.* Available at: https://pypi.org/project/pylint/ (Accessed: 05/02/2023)

Batterbee, I. (2020) *Don Norman's seven important questions of user interaction.* Available at: https://uxdesign.cc/ux-psychology-principles-seven-important-questions-960579272880 (Accessed: 05/02/2023)

Cohn, M. (2005) *Agile Estimating and Planning.* Upper Saddle River: Pearson Education, Inc.

Cohn, M., Ford, D. (2003) *Introducing an agile process to an organization [software development]*. Available at: https://doi.org/10.1109/MC.2003.1204378 (Accessed: 05/02/2023)

Duval, P., Matyas, S., Glover, A. (2007) *Continuous Integration: Improving Software Quality and Reducing Risk.* Available at: https://www.oreilly.com/library/view/continuous-integration-improving/9780321336385/ (Accessed: 05/02/2023)
Flaviocopes. (2020) *Testing JavaScript with Jest*. Available at: https://flaviocopes.com/jest/ (Accessed: 05/02/2023)
Gordon, K. (2020) *5 Principles of Visual Design in UX.* Available at: https://www.nngroup.com/articles/principles-visual-design/ (Accessed: 05/02/2023)

Grinberg, M. (2021) *How to Write Unit Tests in Python.* Available at: https://blog.miguelgrinberg.com/post/how-to-write-unit-tests-in-python-part-3-web-applications (Accessed: 05/02/2023)

IBM. (2023). *ACID properties of transactions.* Available at: https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions (Accessed: 05/02/2023)

Jin, Z & Offut, A. (1999) *Coupling-based criteria for integration testing.* Available at: https://onlinelibrary.wiley.com/doi/10.1002/%28SICI%291099-1689%281998090%298%3A3%3C133%3A%3AAID-STVR162%3E3.0.CO%3B2-M (Accessed: 05/02/2023)

Schwaber, K & Sutherland, J. (2020) *The Scrum Guide.* Available at: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf (Accessed: 05/02/2023)

Schwaber, K & Sutherland, J. (2020) *The Scrum Guide.* Available at: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf (Accessed: 05/02/2023)

Verheyen, G. (2019) *Scrum A Smart Travel Companion.* Available at: https://ecole-management.com/BOOKS/1.pdf (Accessed: 05/02/2023)

Wang, H., Xie, M. & Goh, T. (2010) *A comparative study of the prioritization matrix method and the analytic hierarchy process technique in quality function deployment*. Available at: https://doi.org/10.1080/0954412988361 (Accessed: 05/02/2023)

Zhongyuan, J. (2020) *Design and Implementation of Full-stack Testing for Web SPA in JavaScript.* Available at: https://aaltodoc.aalto.fi/handle/123456789/46113 (Accessed: 05/02/2023)

Software used for diagrams:

Draw.io. Available At: https://app.diagrams.net/ (Accessed: 05/02/2023)
Figma. Available at: https://www.figma.com/ (Accessed: 05/02/2023)
Lucidchart. Available at: https://lucid.app/ (Accessed: 05/02/2023)
Miro. Available at: https://miro.com/diagramming/ (Accessed: 05/02/2023)