CS261 Requirements Report

Team Organisation

Roles

For a team of 6 people, we have assigned the following roles:

- **Project Manager** - Andreea Nicolae
- **Research** - Raihanah Lukman
- **Software Engineer** - Heath Nicholson, Aris Papakonstantinou, Billy Pentney, Cem Yilmaz

This decision was made based on an initial assessment of each member's strong points and how they feel they could contribute best to the project. The project manager is responsible for organising the team and making sure deadlines are met. Additionally, the researcher's role is to bring valuable information to the group ensuring that any decisions made in design and requirements are backed-up with research. The development team was naturally divided based on everyone's individual skill and past experience.

Methodology

Our methodology is an agile approach inspired by Scrum. We decided we wanted to work in cycles, with each cycle achieving a set of requirements, similar to the scrum cycles. A member of the development team will take a similar role to that of the Scrum master and ensure the deadlines are being met. However, since we are all undergraduate students with little experience, we could not fully implement Scrum since we are not familiar with it. Additionally, due to the nature of the project, we are unable to interface directly with the client, so the scrum methodology would not work. Instead we have opted for creating a plan for the development with weekly cycles that we can check the progress of. This works for us since we have a clear deadline and the requirements are already set out from the beginning.

Important decisions with regards to the project are generally taken democratically to ensure the engagement and agreement of all team members. Meetings were agreed to be organised with a minimum of 2 meetings a week, every Monday (online) and Friday (in-person). We have also agreed upon having small or urgent meeting opportunities when required.

Functional Requirements

Account Management

R.1 - A manager must be able to create and login to/logout from a password-protected account with the system using their email address.

R.1.1 - Each user should have access to the following utilities:

- Create multiple projects
- Mark a project as completed, or terminate it
- Add or remove deadlines for sections of the project, and mark them as met or not
- Add or remove the emails of team members for the given project, for data-collection purposes
- View the project metrics stored by our software for the project and receive any suggestions the software has
- View all the projects they manage

R.1.2 - If a user is logged in on a device and attempts to login on another device, the attempt should be denied and a prompt will be displayed to log out of the first device.

Metrics

R.2 - The software will measure a number of metrics about each tracked project in order to generate the risk assessment and provide appropriate suggested actions.

We have split the definition of metrics into two categories. Hard metrics are objective factors of a project that are prompted as an input during the creation of a project in our software and updated throughout. Soft metrics are defined as factors that are subjective - i.e. project attributes which are dependent on opinions and emotions of members of the team.

Hard Metrics

R.2.1 - The Hard Metrics will be provided by the Manager via user input
R.2.2 - Hard Metrics related to the code can be gathered directly by linking a Github repository

Metric	Definition	Reasoning
Budget	The fund allocated for the project	To generate a budget trajectory To check if the project is on track to be completed with the funds allocated
Bug Tracking	The number of known bugs and issues which are retrieved from the linked GitHub repository	To determine the proportion of outstanding bugs/issues
Commit Frequency	The mean time between commits to the linked Github repository	To determine how frequently the code-base is updated/the activity of the development team
Timeframe	The project's various internal deadlines as well as the final deadline	To track the progress of a project through the number of deadlines met/postponed
Team Size	The number of members in the project's development team	To measure the scale of the project; to ensure the budget is appropriate for the project size
Expertise	The software-development experience of each team member	To measure the sufficiency of skilled software engineers for a project; to ensure the budget is appropriate for the team's skill level

We have decided that the software should not consider code quality, as this metric is particularly difficult to measure and outsourcing the quality-check to third-party software presents a potential security risk.

Soft Metrics

R.2.3 - The data for soft metrics will be collected via short periodic surveys which must be completed by each member of the team.
R.2.4 - A unique, initial survey will be sent out to gather some of the data when a team member is added by the manager.

Risk Assessment

R.3.1 - When a project is first created by the manager, the system's Risk-Assessment model will provide an estimation of project riskiness based on the initial Hard Metrics.

R.3.2 - After each Manager update to the project, the system will identify new data for the Hard Metrics and update the Risk-Assessment accordingly.
R.3.3 - When the Risk Assessment is generated, the manager should be presented with a list of suggested actions which could be taken to reduce the risk, and this list is ordered by importance and relevance to the project.

Evolution

R.4.1 - When a project is marked as complete by the manager, our system will prompt the manager to input a final summary, evaluating the success of the project.

- The response provided by the Manager will be fed back into the Risk-Assessment model, to influence the assessments it provides for future projects managed via the software.

R.4.2 - After project completion, the software could continue to prompt the manager for regular project updates (e.g. 3-month reviews), in order to monitor the success of the project over its lifetime and evolution.

- This element should not be overlooked as costs during the Maintenance stage are estimated to represent 90% of the cost of a Software-Development project (CITE).

Visualisation of Output

R.5.1 - When viewing a project, the system will display tracked project metrics along with providing appropriate suggestions.

- The suggestions will be classified as High Priority and Low Priority based on the risk assessment.

R.5.2 - Project Analytics are presented visually, with graphs for:

- Timeline / tasks deadline (stacked bar chart, similar to a gantt chart)
- Budget trajectory (Line graph with forecast)
- Risk-level, over-time (Line graph)

R.5.3 - The project statistics and suggestions can be downloaded as a PDF file which can be shared with the rest of the team or the client if needed.

Non-Functional Requirements

R.6.1 - The software must be accessible via a website with a succinct but well-labelled user interface. That way, the software will be intuitive and accessible for non-technical users.
R.6.2 - The software will require little user interaction other than clicking and entering text making it as easy to use as possible. Furthermore, an "information" or "question mark" icon may be conveniently placed within the page consisting of helpful tutorials or tips for new users. This way, users can receive the minimal training they might need when using the software.
R.6.3 - System compatibility and extensibility

- The system must be accessible via Google Chrome version 109.0.5414.119 running on Windows 10 10.0.19045 Build 19045
- Consequently, it should run in other modern browsers and could be designed to run at mobile-friendly resolutions
- The load time of the website should be under one second depending on the host's throughput. The expected uptime of the website should be above 95% meaning that the website will almost always be up unless unprecedented circumstances occur.