COMP3000

Computing Project

2020/2021

A Web Platform for Visualising Optimisation Data

Links

Source code: *http://www.github.com/GoelBiju/Visualising-Optimisation-Data*

Backlog :

https://tasks.office.com/live.plymouth.ac.uk/Home/PlanViews/NcPq0h7cMkqbkOs9q8RDvZYAHDco?Type=PlanLink&Channel=Link&CreatedTime=637380989887870000

Goel Biju (10576090)

Project Vision

This project aims to develop a "*Web Platform for Visualising Optimisation Data*", for evolutionary computation research at the University of Plymouth, to visualise aspects of optimisation data that is generated from evolutionary algorithms. The web platform will provide the ability for the optimisation data collected from client optimisers to be visualised in real-time and for the playback of saved optimisation runs. The deployed web platform will act as a single tool which collects all the appropriate visualisations for optimisation runs.

Scope

The project will develop a system that contains 3 areas: the frontend, API and the database.

The frontend will allow for a user to interact and view visualisations from live and past optimisation runs. This frontend will be deployable as a platform on the web. The API allows for client optimisers to post data from optimisation runs which can be then stored in the database to be retrieved by the frontend via the API. The database and the API are essential when providing real-time and the playback of saved optimisation runs.

The scope of this project is mainly determined by visualisations we can add onto the front end. The database and API will provide core functionality to handle the visualisations that are provided by the frontend. The project will allow for graphs which will display 1, 2 and 3-dimensional data.

Plan

The project will follow an agile and iterative approach to development and from the initial brief of the project, requirements are gathered to set up sprints (with a 2-week duration). Each sprint will have a review and retrospective at the end and this allows showing the current state of the project to the stakeholder and get feedback on any changes they expect which will directly influence the direction the project undertakes.

As sprints progress, the project will approach various milestones on the roadmap. An example of this is the midway period where the aim is to have a prototype of the system built. This will only be possible if the project works towards a minimum viable product (MVP) for the stakeholder.

Since the project is conducted in an agile manner, the project can be shaped according to the stakeholder's input as the project progresses. The project planner/board will be set up with a backlog of items (derived from requirements engineering) to form all, the tasks required for the project to meet the expectations of the stakeholder.

In addition to the planning and designing, the development stage (which ensues planning) needs to be conducted accordingly to plan. It has been emphasised on the brief of the project that the software solution needs to be *robust* and *correct*.

To enforce this, the project will need to have validation and verification processes, in the form of DevOps and testing, to ensure the final solution is up to a high standard. Providing good version control and incremental updates alongside continuous integration which allows for unit, integration/usability testing are approaches that needs to be taken onboard to meet the project's demands.

Risk Plan

The following table highlights some of the various risks that this project may face and their likelihood (rated between *unlikely*, *possible* and *likely*) of happening. In addition to that, it highlights what impact it may have on the project if it occurs and what preventative measures are to be taken to ensure it can be dealt with or avoided.

Risk	**Likelihood** (Unlikely, Possible, Likely)	**Impact**	**Prevention**
Hope creep	**Likely**	Project is behind schedule and reporting the false expectation that it will be back on schedule soon. This means unable to deliver at pertinent points during the project and cannot catch up in time and produce the expected level of work.	Staying to schedule and having manageable tasks is one way to ensure this does not occur. "Hope creep" can be due to too small chunks of work or very large pieces (linked to risk of having large work items – see below). The preventative measure here is to make sure that we make a suitable plan of tasks which meets our working demands as well as the project demands.
Scope creep	**Possible**	Unexpected changes and uncontrolled growth in the project which can disrupt progress and a useful solution is not created.	As mentioned initially when defining the project vision, the scope should focus on the core tasks required to have an MVP/prototype working for the stakeholder. This means aiming to stick to the tasks that are

			only necessary from the initial project brief and not adding else unless necessary.
Lack of proper/inadequate requirements engineering	**Unlikely**	This can disrupt the whole project and lead it to collapse if core requirements are not collected.	Requirements engineering does need to be thorough and assess what is required (as core) and what is additional initially. This defines what tasks to prioritise and to spend the most time during sprints. As well as this, it should help identify the large tasks which need to be broken down to smaller chunks of work.
Lack of communication with stakeholder	**Unlikely**	Having no communication can be detrimental to the project as it puts the creator and the stakeholder in a position where they do not know what to do or what is going on regarding the project. This may directly relate to “scope creep” (as mentioned above) where you may develop without having an idea as to which bits are important to the user or not.	Communication is mandatory always. The project aims to have retrospectives with stakeholder at the end/start of sprints to inform on progress made and gain feedback which can then influence the next steps required.

Large work items not managed correctly (time and effort)	**Possible**	Similarly, to “scope creep”, having unmanageable chunks of work in the backlog will mean the project will not progress in the time frame and an MVP cannot be reached in time.	Breaking large work items (from requirements) into manageable chunks. Amount of effort needed to carry out a task is underestimated/ignored, split large tasks. The time for specific tasks may vary and need to prioritise what is needed in a minimum viable product (MVP).
Visualisations can be complex	**Unlikely**	An aspect of this project is that visualisations for the optimisation data have been illustrated through research papers.	During the initial meetings with the stakeholder, we discussed the graphing and visualisation requirement from the project and what is feasible. The key point from the meeting was to get an initial prototype which incorporates a basic visualisation (in the form a 1D/2D graph) which allows us to build the core.
Incorrect use of/not using the appropriate tools for the task	**Possible**	It may be that the technology sources picked may not be appropriate when fulfilling a specific user story. As a result, it can hinder and delay the project.	It may that initially the wrong tools/technologies are not chosen but there needs to be an Architectural Decision Record (ADR) kept on what choices have been made and what other alternatives there and why a specific technology was chosen.
Changes to requirements	**Likely**	It may be the case that requirements get changed due	The main preventative measure here is to ensure the requirements

specification during coding		to technical issues or a change in a specific requirement. It could also be a development error where requirements were not developed in an ideal manner.	engineering is thorough and any clarifications required for a requirement on the project (to derive the user story) has been discussed with the stakeholder. Leaving specific requirements ambiguous will not help the project progress.
Specification takes longer than expected	**Possible**	It could be possible that a particular requirement takes up too much time which can further cause delays in sprints or delay sprint items.	If items are in manageable chunks, then this should not occur, but it is worth reviewing items as sprints progress to see which items need to be prioritised and where time needs to be utilised correctly.
Inclusion of unnecessary features with no use ("Gold plating")	**Unlikely**	In this case, we may include unnecessary requirements or work on items which may have no benefit to the stakeholder or serve no purpose in the project. This can waste a lot of the time and you would not be spending time creating anything meaningful.	Requirements scrubbing will need to take place to remove complex or unnecessary requirements throughout the duration of the project. Developing an initial prototype and working to the key tasks in the beginning, we can make sure our core user stories are fulfilled before trying to add anything else. As the requirements may develop, it is worth reviewing at the end of the sprint the current state of the backlog and how items need to be changed.

Late changes to requirements	**Possible**	Requirements are subject to change and it maybe that they could change at late points. This may put time constraints on what can be done to meet the new requirement.	Through incremental updates to the project and by ensuring that there is code has been created with control will help to prevent introducing any breaking changes to the project.
Real-time performance problems	**Possible**	It can be possible that there may be specific performance issues or unforeseen technical challenges when dealing with sending data in real-time.	Through prototyping on a small scale initially, with basic visualisations, and testing out the framework, we can get our idea working before proceeding to work on larger user stories.
Development technically too difficult	**Likely**	Development may be slow or difficult when approaching a specific task. This can again consume a lot of time on the project.	If there is a technical challenge it is worth analysing technically what can be changed to simplify the problem. It may be that we can simplify the problem and then change it to work as we originally intended it to. Lack of experience in a technical domain; the novelty of the software – learn about it by getting the appropriate training or simplify the issue into something that can be done. This may be applicable when developing the whole stack and creating the visualisation required. Using the resources available, train

			yourself to the tools required for the tasks.
Developing the wrong software functions	**Unlikely**	There may be situations where the software has been developed incorrectly or not as the stakeholder intended. This can consume time when developing again to correct the initial mistake.	Making sure the stakeholder is always involved in the process through communication and making sure there is clarity to how the software should function is important for every user requirement that we have.
Developing the wrong user interface	**Possible**	Just as the software functions can be susceptible to being developed incorrectly, the user interface on the frontend can also be designed not according to expectations.	Include prototyping of the user interface i.e. paper/wireframes (performing task analysis or surveys to collect information as to what features/functions are required) which can then allow for user involvement meaning the user is always influencing the project.

Keywords

Visualisation, Optimisation, Web Platform, Evolutionary Computation