Software Project Proposal

Midterm assessment – 30% of your mark for this module

This coursework assignment explores the design and development of your software project. The deliverable here is a proposal which defines and describes a number of key elements of your project. You should produce a document that is no longer than 8,000 words in length and should be submitted as a single PDF document. This proposal should be used to explore design decisions, consider the context of use and identify the process by which the software project is developed. As such, your proposal should include a reasoned justification that explores the following topics:

1) A clearly defined set of deliverable components of the software and the job of work required to bring these components to completion.
2) The defined timescale of work, including any dependencies, milestones or contingencies.
3) A formal specification of the desired system (e.g. UML, technical and functional specification.) You should also include user-acceptance criteria for testing at this stage.
4) A clearly defined scope for the project defining areas that you will and will not be delivering on.
5) Some evidence of requirements elicitation involving some/all of your project stakeholders.
6) A research summary that highlights the challenges of working within your chosen domain.
7) Evidence that compares your project to similar software tools (e.g. market analysis.)
8) A description of your approach that discusses the motivations and reasoning for working in a particular manner (e.g. User-Centred Design, Test-Driven Development.)
9) Some early prototypes showing how the project will work and highlighting the strengths and weaknesses of your proposition.
10) Some early evidence of assumption testing and validation of your designs to date (e.g. user tests or automated feedback such as W3C validation/accessibility testing, heuristic tests etc.)
11) A critical evaluation of your concept, your project in its current state and the proposed software project.

The document should highlight a clear and systematic rhetoric with critical analysis and an overall evaluation regarding the current state and feasibility of the approach presented.

Marking will follow these general guidelines (out of 30 possible marks.)

>70% 21 or more marks	Shows good critique, a functional understanding of all of the concepts taught and describes the iterative design process in detail. Students should take an analytical approach and evidence an advanced array of techniques for higher marks. Groups should evidence a systematic approach and attempt to frame their projects against a variety of factors such as competitors in the market and wider academic and non-academic considerations. A successful project will include evidence of technical merit, clearly defined steps for iteration and improvement and a strong report to account for the design and development of these things. For higher marks, students should select projects that are ambitious, novel and utilise contemporary techniques and technologies.
65-69% 19-20 marks	Good functional understanding with a clear grasp of the core concepts. Groups should produce evidence that they have made clear, logical decisions with sufficient research to support decisions. At this stage, groups should present a good, reasoned account of their project with some critical analysis around the core concepts and techniques. There may be some minor flaws in either the process or in the technical aspects of this marking band but students should show extensive knowledge in at least some of the core learning outcomes of topics studied thus far.
60-64% 18-19 marks	A reasonable understanding of the core concepts and competencies required, with a functional implementation of user centred design and software development techniques that are fit for purpose. Groups should be able to distinguish between good and bad design decisions and make use of some of the techniques studied in the module. There should be some formative research and commentary to dictate rhetoric and design decisions.
50-59% 15-17 marks	Groups should show some understanding of fundamental concepts of system and software development lifecycles, with a consistent and clear approach to their work. They should be able to present their ideas in a meaningful way and have a grasp of the software development lifecycle and an approach to building and evaluating software in an iterative, collaborative way.
40-49% 12-14 marks	Groups should show some grasp of core concepts of the module, but not enough to critically assess their own design decisions and/or flaws in their premises.
<40% less than 12 marks	Unacceptable standard of work, presenting little or no relevant information and only showing evidence of understanding in some key areas of software engineering.