IT5007: Software Engineering on Application Architecture

Course Project Evaluation Rubrics

This document will serve as a guideline for the course project evaluation. Please be informed that the actual evaluation may diverge from this document if there is a need. The project has the following evaluation components and define various ways of getting points. **You do not have to satisfy all of the criteria below** but try to do as much as you can in the given time, considering the workload for IT5007 as well as other courses you are enrolled in. The final marks awarded is relative to other projects in the class.

Total points: 35

Deadline: 30th November

Tentative Evaluation Rubrics

General (6 points): Relevance of Problem Statement, Solution Architecture, Legal aspects and business model (open source vs. commercial), Competition analysis

Implementation (12 points): Front-end Design and Implementation, Back-end design and implementation, Integration

Misc Software Engineering Aspects (3 points): Documentation, Usability, Modularization/Library integration, Code Originality

Novel Features (10 points): Novel front-end and back-end features. This component is computed relative to the other projects in the class.

Presentation (4 points): Presentation in week 13.

General

1) Problem Statement: Novelty of the problem, challenges involved in solving the problem, Is the problem relevant in 2/5/10 years? complexity of the solution and understanding of the problem domain will be evaluated. →Add to Documentation (i.e., Create a word document or a ReadMe file with these details and add it to the Github classroom repository.
2) Solution Architecture: Wireframes/Figma illustrations/Information architecture/Mock Up can be considered towards this. Modularizing the overall solution and architecting the solution. →Add to Documentation (i.e., Create a word document or a ReadMe file with these details and add it to the Github classroom repository.
3) Legal/Other Aspects: Use of open source code in the project as well as the possibility of open sourcing the project itself need to be explored. Ideas for protecting the project from being copied will be evaluated. → Add to Documentation (i.e., Create a word document or a ReadMe file with these details and add it to the Github classroom repository.
4) Competition Analysis: Explore the nearest market product that competes with your solution. →Add to Documentation (i.e., Create a word document or a ReadMe file with these details and add it to the Github classroom repository.

Implementation

Note that the split between the points awarded for UI and Back-end will vary on a case-by-case basis. So, if your solution is UI-heavy, focus on adding more novelty to the UI; and vice-versa. Note that you only have to cover **some of** the below declared points in your final submission.

1) UI Implementation

a. Design: Evaluate if the UI design is well thought out. Most important elements/components of your solution will need to occupy more space visually. You can also consider using a color scheme/contrast/positioning to highlight the most important/relevant features. Link between the pages (i.e., navigation) should be intuitive to the user. UI Kit: colors, fonts, button, icons – should be uniform throughout the website. Use of bootstrap/own CSS for enhancing the look and feel.
b. Novel Features: Examples include React Routing, advanced animations enabled through web code (e.g., react), other libraries. In the past, this has proved to be a main difference between A (i.e., A+, A, A-) and B (i.e., B and B+) grades. But, start aiming for these after making sure that the basic design of your website is stable and working.
c. If you are wondering if a particular feature would be considered novel, please write to the instructor to check!
d. Usability/Performance: Broken URL links will be penalized (links that connect to back-end/API are exempted from this check), make sure to perform form validation in Javascript.
e. **Most importantly**:Update the Readme file on Github classroom repository with a list of front-end features that you have coded in your application to make it easier to evaluate against other projects in the class.

2) Back-end Implementation
   a. Architecture: The choices involved in deciding what are the incoming requests supported by your back-end, how each of those requests are processed, and if you are using 3rd party services, how do you interface with them. Are you going to talk to 3rd party API from your front-end or from your back-end? Add these details to your documentation.
   b. Implementation: At least one heavy back-end technology (e.g., Database, ML framework, or 3rd party API) needs to be implemented. The back end must be able to process an incoming request from a browser and provide a response using the back-end technology. Complex back-end technology will carry more points. Routing through multiple back-end technology will also be considered a bonus. In the past, this has proved to be the main difference between A and B grades. Integration with many 3rd party APIs and the ability to use their data/service effectively is also considered novel.
   c. Authentication: You can maintain your own authentication database. Integration with 3rd party services will also carry points. Maintaining a user session throughout the website will carry points.
   d. Setup Automation: Scripts to automatically setup the back-end environment. For instance, just by running npm start, the database, and other services must be initialized and start running. Use of python or any other script (e.g., Make) is also allowed.
   e. Other novelties: Interactions between back-end services (e.g., fetch from database and then send a request to google to do something), Performance enhancement, etc.
   f. If you are wondering if a particular feature would be considered novel, please write to the instructor to check!
   g. **Most importantly**: Update the Readme file on Github classroom repository with a list of back-end features that you have coded in your application to make it easier to evaluate against other projects in the class.

Miscellaneous Software Engineering Aspects

1) Documentation: add comments to your code. This can be done at the function level (i.e., add comments on top of each function)
2) Usability: The system should be usable. Usability refers to how easily and efficiently users can interact with your website to achieve their goals. You can consider a typical user who is lazy and tries to give invalid inputs or skips giving relevant inputs.
3) Code Modularization: creating library components for commonly used tasks.
4) Code Originality: how much of the code was taken from other projects/github? It is completely ok to borrow code but you must declare it in the ReadMe file.

Final Submission

1) You are also supposed to update the github classroom repository with your → latest code, latest ReadMe (with documentation, updated list of features, your name and matric number).
2) We will be checking if you have made continuous progress in the project over the semester using the github classroom commits. Committing all your code/materials in week 13 is not allowed and you will be penalized.
3) We will be checking for plagiarism. If you plan to reuse/copy code from elsewhere (e.g., another github repository), get the permission of the instructor beforehand.
4) There will be a final presentation in Week 13 for your project.