IT5007: Software Engineering on Application Architecture

Course Project Evaluation Rubrics

This document will serve as a guideline for evaluating the course project. 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. The final evaluation for the course project component is relative to other projects in the class.

Interim Evaluation

Points: 10

Deadline: 15th October 2023

1) Solution Architecture (3 points): Wireframes/Figma illustrations/Information architecture/Mock Up can be considered towards this. Modularizing the overall solution and architecting the solution. Create a word document (i.e., a project report) with these details and add it to the Github classroom repository.
2) UI Implementation (7 points)
    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.
    b. Use of bootstrap/own CSS for enhancing the look and feel
    c. 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.
    d. Documentation: add comments to your code
    e. Most importantly: Present a list of front-end features (i.e., add to the report or README file on git repository) that you have coded in your application to make it easier to evaluate against other projects in the class.

Final Evaluation

Points: 40

Deadline: 3rd Dec 2023

Project Report

1) Class Participation (1 point): Participation in the class discussions on MS Teams will be evaluated. Cases where the student writes an explanatory post for the peers will be considered. This is an individual component. So, each student must post something on MS Teams to get this point.
2) 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.
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.
4) Competition Analysis: Explore the nearest market product that competes with your solution.
5) You are encouraged to add a list of features, Project Schedule, and details of the project modules assigned to each team member.

Code and Documentation

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.

UI Implementation

Building on what you have already submitted for interim evaluation, do the following:

1) Architecture/Development: Code Modularization, creating library components for commonly used tasks, Code Originality (how much of the code was taken from other projects/github)
2) Novel Features: Examples include server-side rendering, 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.
3) Most importantly: Present a list of front-end features (i.e., add to the report) that you have coded in your application to make it easier to evaluate against other projects in the class.

Back-end Implementation

1) 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. Add these details to your documentation.

2) 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.

3) Authentication: Integration with 3rd party services will carry more points. Maintaining a user session throughout the website will carry points.

4) 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.

5) Novelty: Interactions between back-end services (e.g., fetch from database and then send a request to google to do something), Performance enhancement, etc.

6) Documentation: add comments to your code

7) Most importantly: Present a list of back-end features (i.e., add to the report) that you have coded in your application to make it easier to evaluate against other projects in the class.

Final Submission

1) You are supposed to submit the modified/updated Project Report.
2) You are also supposed to update the github classroom repository with your latest code.
3) The classroom repository should have working code, instructions on how to execute the code, and documentation.

Tentative Evaluation Rubrics

Basic: Class Participation, relevance of Problem Statement, Solution Architecture, Legal aspects, Competition analysis

Implementation: Front-end Design and Implementation, Back-end design and implementation, Integration

Misc: Documentation, Usability, Modularization/Library integration, Code Originality

Novel Features: Novel front-end and back-end features