CSE470: Software Engineering, Project Outline

Project Purpose

The purpose of the project is to provide hands-on training to the students on how to develop a software application from scratch.

Key Definitions and Examples

Requirement: an action/capability that is expected of a software/product to fulfill
Feature: a set of logical actions that need to be performed to fulfill a part of a requirement

Example of a requirement

1. Customers can add products to their cart
2. Customers can track their delivery status

Example of features

- For requirement (1) in the examples, the features that need to be implemented are:
    - The system should be able to display products to customers
    - The system should be able to book products and mark them as added to the cart of a customer if the product is in stock
- For requirement (2) in the examples, the features that need to be implemented are:
    - The system should be able to show all undelivered orders from the customer
    - The system should be able to store data regarding the status of the order
    - The system should be able to store data regarding the products in the order
    - The system should be able to show the status of the products and the order to the customer

Tech stack: The language and the environment which is used to develop and run an application
Example of tech stacks

- **LAMP** Stack (Linux OS, Apache web server, MySQL database, PHP language)
- **MERN** Stack (MongoDB database, Express JS backend, React JS frontend, Node JS environment)
- **Php Laravel**

Project Constraints

- Students may develop either a web application, an Android application, an iOS application, or a desktop application.
- Students may use any tech stack/programming language/scripting to develop their application
- The application developed **MUST** follow the **MVC architecture**
- Students may use existing frameworks (Laravel, Express, Angular, Vue) to help with the development as long as they implement the MVC architecture. **(Django is not permitted, and using Flask is highly discouraged.)**
- Students may use libraries/packages to help with the development of their project as long as it **does not** implement any major feature of the project. A library/package implementing any major feature is considered illegal, and the student using the library will be penalized. Consulting the respective faculty member before using a library/package is recommended.
- Students must develop the project individually and from scratch. They may not take help from any source, copy code from any source, or use any tutorials that directly show how to implement the project that they have chosen.
- Students may not use a code base that they/others may have worked on before.
- Students must select a project that they want to implement, which is of lower to moderate complexity.
    a. Login/registration **will not** be considered a feature.
    b. Students must define the requirements and the features of their project while submitting their project idea.

Project Process Details

- There will be **one project per four students**, and each student is individually responsible for completing the project as a team.
- The project will have **a minimum of 20 features.**
- There will be **four sprints** to complete the entire software implementation.
- The sprint duration will be approximately **7-14 days.**
- Students **MUST** submit their project idea, requirements, and feature before the beginning of the first sprint.
- A section faculty may change the project, the requirements, and/or the features if the section faculty determines that the project is not complex enough, or too complex, or for purposes of removing potential duplication.

- Students need to submit their work progress through a **Sprint submission Google form** at the end of each sprint.
- Each student must use **GitHub** as a version control tool for their project and add the section faculty as a collaborator to the repository during the first sprint.
- Each student must push their code to the project repo at the end of each sprint, regardless of whether there are incomplete features, errors, or bugs.
- Students must complete their projects within four (4) sprints.
- After the sprints, each student must attend a 10-minute viva where
    - They will show their project demo in 5 minutes.
    - They will be asked questions about their project code individually for 5 minutes.
- All vivas will be recorded and stored for review purposes, if possible.
- The viva may be taken offline or online, and the dates and times will be assigned by the section faculty.

Project Marking Rubric

SN	Category	Marks (20)
1	Software Requirement Specification (SRS) Document	2
2	Sprint submission and version control (git)	3
3	Project completion (frontend and backend) and quality (unique idea)	2
4	Implementation of MVC architecture	3
5	Project Viva (individual contribution and Q&A)	10

Project CO Assessment

SN	Category	COs	Marks	Marking Criteria
1	Software Requirements Specifications (SRS) Document	CO2	2	**0:** No submission or plagiarized **0.5:** Incomplete, unorganized, unclear requirements **1:** Acceptable report, but lacks clarity and structure **1.5:** Well-structured and mostly complete, minor gaps

				2: Exceptionally detailed, clear, and organized SRS following all guidelines
2	Sprint submission and version control (Git)	CO5, CO6	3	**0:** No Git activity and sprint submission **1:** Very few commits and sprint submissions **2:** Regular commits, good collaboration **2.5:** Frequent commits with proper branches and teamwork **3:** Excellent use of Git: commits, issues, branches, and collaboration clearly shown
3	Project completion (frontend and backend) and quality (unique idea)	CO5	2	**0:** Non-functional or mostly incomplete **0.5:** Minimal implementation, major issues **1:** Functional but lacks polish or innovation **1.5:** Fully functional and reasonably designed **2:** Creative, polished, fully working frontend and backend with good design
4	Implementation of MVC Architecture	CO3	3	**0:** No architecture used **1:** Attempted MVC but poorly implemented **2:** MVC applied, but missing full separation **2.5:** Proper MVC with separation and reusable code **3:** Clean, modular, maintainable MVC with excellent adherence to design principles
5	Project showcase (individual contribution and Q&A)	CO6	10	**0–2:** No contribution or unable to answer questions **3–5:** Some understanding, unclear explanations **6–7:** Moderate understanding with decent explanation of own tasks **8–9:** Clear explanation, confident, used tools effectively **10:** Deep understanding of the entire project, clear logic, confident use of tech and architecture