CSC440/540 Team Project

Introduction to CSC440/540 Team Project

Software engineering is about the rules and tools for building software products to solve real-world problems. Building a software product is hard mainly because of the complexities inherent and accidentally introduced in building the software product. In this team project, students learn how to build a software system–Knowledge Management System using Wiki–in a team. Specifically, they learn how to apply the software engineering ideas, theories, and rules to build a high-quality software system that solves a real-world problem by managing the complexities. Students also learn how to make documents to support efficient software development and maintenance. The professor works as a coordinator of your team project. Your team, as a whole, is responsible for the software product you build.

Project Goals

1. Application of Theories: apply software engineering ideas/theories/rules to a real-world software product development.
2. Teamwork: learn and experience how to define and solve problems (mainly software development related issues) in a team.
3. Software Development: learn and experience real-world software development tools, design methods, and processes.
    1. build a professional level software product (Knowledge Management System) with software tools including Python, PyCharm, Flask, Wiki, UML, UnitTest and with software documentation tools including Markdown and GitHub
    2. use the developed software product and documentation for their software portfolio.

Project rules

1. Plan ahead of your action. Planning is a group effort in that you meet together to set up a plan with milestones. Also, uploading the plan on Wiki440 is a team leader's job. Each team should make the plan known (no surprise rule), and check the progress (whether each team member accomplishes the milestone goals).
2. Each team can change their schedule anytime they want as long as (1) they make a record of the changes and (2) finish the planned work before a deadline. Any change should be recorded and submitted as a part of deliverables.

3. Use 'GitHub' to publicize your project artifacts. No plagiarism is allowed, but you refer (not a blind copy but reinterpretation or internalization) to other team's coding or documents with (1) citation and (2) reason why your team needs to do that.

Project Stages

Why a team project in stages?

This team project is designed to mimic a real-world situation where (1) you work for a software company (2) as a software engineer. I assume you are hired as a junior-level software engineer, but you aim to be the leader of the team: i.e., a manager (or higher) or a senior-level software engineer (or higher) in the long run. Team members take on the role of team leader, in turns, to coordinate the team effort when you choose the `democracy` system. Otherwise, one leader takes the responsibility of the leader's role for the rest of the semester.

As a junior software engineer, you are typically given a short period to familiarize yourself with software development tools, and a company's software products once hired. The D stage and C stage mimic each of the situations. The D stage is for learning the tools including Python, PyCharm, and Flask. The C stage is for learning, or reverse engineering, how the current product, the existing Wiki, is structured and designed using the tools learned in stage D. Once C stage is completed, you will extend the features of the current Wiki in the B stage. A stage is to take what you learned and developed a new product, Knowledge Management System.

You also can think that each stage represents a unique iteration from unknown-unknowns to known-knowns. In other words, I expect four iterations to build a new software product from an existing one.

Cooperation is encouraged, but plagiarism is a crime

The deliverables for your team are provided at the end of each stage later in this document. I encourage everyone to cooperate for learning Python, Flask, and Wiki in C and D stages to jump into B and A stage to start the software design and implementation. You can share any study material or documentation during your study with any other students. However, you should be careful to cite (or let known) the sources you used when you used others' help.

Class Wiki (Wiki440) as a `main` communication tool

We have the wiki system for CSC440/540 (Wiki440) deployed with the existing Wiki. You will use Wiki440 for (1) checking the wiki features and (2)

communication among team members and the coordinator (the professor).

1. The coordinator will use Wiki440 for coordinating each team project.
2. The team will use Wiki440 to record its project progress, link their project documents and artifacts.
3. Each team use Wiki440 for information upload and send an email to the coordinator if the information should be known to the coordinator.

Stage Goals - from D to A

D stage - learning the tools

In this stage, you learn Python/PyCharm and Flask.

1. The coordinator (the professor) provides D stage study materials (Python, Python Decorator, Flask, and Jinja2) and gives outside class lectures.
2. Each student submits D stage deliverables, but each student is encouraged to work together to enhance their understanding of study materials.

C stage - learning the existing wiki system

In this stage, you understand the structure of Wiki system built with Python, Flask, and RESTful API.

1. Students can use PyCharm to reverse engineer the existing wiki system.
2. Students understand the software architecture and design by understanding the wiki system.
3. Students understand RESTful APIs, Web request and response architecture, HTTP protocols, and how Python's OOP paradigm can reduce complexities in developing software.
4. Each team decides what and how to reverse engineer the wiki system. For example, the generation of HTML page from request to HTML can be one topic, the conversion of markdown to HTML can be another. Each team member is assigned a task. The task should not be a trivial task, but it should be reasonably hard and contribute to the understanding of the wiki system.
5. Each team member should share what they learned.
6. Each team grades their C stage deliverables, or teams can pair to grade each other.

B stage - contribution

In this stage, you extend the existing Wiki system by adding features.

There are unlimited possibilities, but I suggest only some of the possible extensions from MVC (Model, View, Controller) perspectives. I expect your team can come up with better ideas. These are just some of the possible examples.

Model

1. Use SQL or MySQL for database system instead of markdown.
2. Support other formats than markdown including asciidoc, LaTeX, or any possible data format.

View

1. Better UI.
2. Javascript - e.g., better online editor
3. Calendar or other facility features.

Controller

1. Plugin system
2. Subitem structure, i.e., wiki/service is a subitem of a wiki
3. Security - login system (the only skeleton is implemented for current wiki system)
4. Version control system, users can revert to old Wiki pages if necessary.

A stage - application

Upon completing B stage, you have (1) a database and (2) Python code to control and manage the database. This means that you don't need to use the wiki to process the database. In this stage, you can build Knowledge Management System (KMS) that processes the database (the wiki database in markdown files) for your needs. For example, you can build a command line Python script to collect all the definitions stored in the database.

Each student makes at least one standalone python application with the following requirements. Each student can make any program as long as it uses your our other team members (even outside of your team) PIP libraries, and it is a standalone Python application that uses the Wiki database.

1. Each team member builds a library for their feature in B stage to upload the library into a PIP repository.
2. Each team member builds a command line tool using `argparse`.

Deliverables

You need to submit project deliverables when you finish each stage.

Plans and retrospect (Common to stages)

1. Schedule
   - Your team should record schedule changes and track your schedule.
2. Meeting agenda files
   - When your team members meet together for the team project, the team leader (or anyone to whom the team leader delegates the task) should make a meeting file.
3. Rules collection
   - Before you finish your stages, your team members meet to collect the rules that you could apply to your tasks.
4. Retrospect
   - Before you finish your stages, your team members meet to retrospect to discuss what went well, what didn't go well, and what each member learned.

Deliverables at each stage

The `checklist.md` and `A/B/C stage rubrics.docx` specifies the expected deliverables.

Schedule

The professor works as a coordinator of your team project. Your team, as a whole, is responsible for the software product you build.

1. Each team should set up a todo list and a deadline for each stage.
2. A team leader's main job is to monitor the progress of each member's accomplishment. The team leader should report the coordinator if anything different from the schedule occurs.
3. Surprising other team members may be the worst thing you can do as a team member. If some team member surprises other team members no matter what reason, the team member should be warned and get low points in peer review grading.

Grading

This project is worth 400 points for students.

- Grading from the deliverables (300 points)
- Grading from peer reviews (100 points)
  - `Peer review rubrics - team` specifies how the 20 points are allocated for a team evaluation.

– `Peer review rubrics - individual` specifies how the 80 points are allocated for an inside-team evaluation.