Custom Project Plan

Name: Sham Ganesh Thamarai Kannan

Date: 09/10/2025

Project Pitch

The Software:

Description:

A Job Finder System using Python. The software will match the applicant with suitable jobs based on the applicant's skills and the skills required by the job. Then the suitable jobs will be stored locally on the applicant's device.

Software Architecture and Framework Plan Overview:

My software will use an API provided by Remotive, a job portal, and query it using requests to get recent jobs that have been posted on the portal. I will make a UI for the python software using Streamlit (commonly used for displaying data in dashboards). The software will ask for the user to enter his/her skills as tags. It will then use the given tags to compare the skills of the applicant with the skills required for the job using a simple NLP pipeline implemented with Spacy. If the job is a good match for the applicant, it will then display the relevant information to the applicant to proceed with the job application. To avoid displaying redundant/duplicate data, my software will store the previously retrieved information in csv format on device (similar to BAT).

Tech Stack and Frameworks:

Main Programming Language:

Python

Summary of Libraries/Frameworks planned:

UI – Streamlit

Data API – Remotive (queried using requests)

Data Processing Pipeline – Spacy (for NLP)

Pandas – Data Strorage and filtering

Testing Plan:

Since my project idea involves a lot of APIs and Frameworks, whitebox testing alone would not be sufficient to check if my code performs as intended. I am planning on testing my project using:

1. category partition testing and boundary value analysis (example: where the conditional matches the applicant's skillset with the job's required skills to check if the match is over a certain percentage)
2. Mocking to simulate functionality of other factors(example: wherever required in the Streamlit UI/giving made up data instead of retrieving data from files stored to improve efficiency of tests and preserve data integrity)
3. Pairwise Testing where the number of testcases are extremely high and cannot be covered easily.

for writing unit tests in python using unittest framework. I will also try to follow an OOP approach to programming to facilitate mocking and easier testing. I will attempt to make the tests cover a significant part of the code (around 70% coverage).

Development Plan:

I am going to follow the TDD process by reversing my usual working style to reduce bias as outlined by TDD. To adhere strictly to TDD, I will first write the tests first before writing code. Even if I don't implement one test at a time, I will try to write all the test for each function I plan to work on to assess its expected functionality as soon as I finish coding it out. I will record my progress by taking screenshots of the code at various stages of its development and how it all culminates to the final working software. To keep track of the project development timeline, I will use git and commit frequently (after every set of tests for a function or element) and push the remote repository after I complete working on a file, or after any major change has been made to the code. I will start working on my project and discuss my ideas with the teaching team to get their input, views, and their guidance. I will also run pylint and pycodestyle after I finish working on a particular file to check if the appropriate documentation has been added and the good programming practices have been followed. Checking the code for styling issues at regular intervals will allow me to correct any bad programming habits, and also break up the task into smaller chunks that will make styling and linting the code less daunting, rather than doing all of it at once for the entire codebase. In short, I plan on starting by exercising an extended red-green-refactor loop that encompasses testing an entire function of code at once and incrementally fix the errors found.

Extended Testing Plan:

1. Usability Testing:
   Since I am going to use streamlit to generate web application like dashboard, I will perform:
   - In House Usability Testing: First refine the UI myself, and improving page structure
   - Unmoderated Usability Testing: Then send a questionnaire asking a couple of friends and family members to review it for it's quality, implement any common valuable feedback received from the participants.

   Approach:

   1. Establish Clear Objectives for the UI
   2. Prepare a questionnaire to test the UI
   3. Perform tests in house first
   4. Improve the UI if needed
   5. Ask Friends and Family to test the UI – fill out questionnaire (to avoid missing out on important details due to bias)
   6. Document the results from questionnaire and record the feedback
   7. Improve UI further (if needed)

Link to Questionnaire Created for Usability Testing

https://docs.google.com/forms/d/e/1FAIpQLSeb1VRQwKroKGBMjrJnbuZVFkW_0c8dvguAbJiezSuvfDRZig/viewform?usp=header