Kendal Guizado

Milestone Two

For Milestone Two, I will focus on showcasing my skills in software design and engineering through the enhancements made to the tenant screening application. This project demonstrates key software engineering principles such as modularity, maintainability, and user-focused design improvements, which align with the Computer Science program's outcomes.

Artifact Description

The tenant screening application was initially created during my first semester at SNHU as a Python project. It was designed to take input data from a landlord or agent about a rental applicant and compute a total score for each applicant, providing decisions on approval. In this enhancement, we transformed it from a Python-based application into a front-end web application. This enhanced version allows users to evaluate multiple tenants based on various criteria. The enhancements now enable functionality to handle multiple applicants, auto-populating shared fields for property address and rent, and tracking applications more efficiently with a structured table.

application-form.component.ts is the core component where most of the functionality resides. Here, I implemented the dynamic form logic for the tenant screening process, allowing users to input data for one to three applicants. I created a mechanism to automatically copy the property address and monthly rent for additional applicants from the first applicant to save time in multiple applicant scenarios. The onSubmit function processes the applicants' data, consolidating it into an application summary, and resets the form after submission. I also implemented an application ID for each submission and logic to handle multiple applicants applying together.

application-form.component.html renders the user interface for collecting applicant data. Here, we display the form fields for applicants and dynamically update the fields based on the number of applicants selected. This file also contains the logic for displaying the submitted applications in a table format, where each entry shows the application ID, the names of all applicants on one line, the date applied, property address, total score, and decision. I also added single-line borders to the table for better readability.

Justification for Inclusion

I selected this artifact for my ePortfolio because it showcases my ability to transition from back-end Python code to a dynamic, front-end Angular application. The new features demonstrate my skills in modular design, efficiency, and user experience improvements. One key enhancement is the implementation of shared fields, which automatically populate data for applicants 2 and 3 based on tenant 1's inputs since they are applying together. Additionally, the inclusion of a structured data table that displays the applicants' names, shared details like property address and monthly rent, and decisions for each application highlights my front-end development abilities and the application's practical, user-focused design.

Course Outcomes Alignment

The enhancements made to this artifact align with the course outcomes I planned in Module One. The refactoring of the application's structure and the implementation of modular code address Course Outcome 3, improving the design and evaluation of computing solutions. I also successfully designed and developed professional-quality software, as seen through the clarity and maintainability improvements to the code, directly meeting Course Outcome 4. The integration of multiple applicants and shared data fields shows my ability to manage complex input scenarios and balance user-friendly design with efficient software architecture.

Reflection on the Enhancement Process

Through this process, I learned the importance of maintaining modularity and scalability in software design. Refactoring the code into reusable functions, like updateApplicants(), allowed me to handle the complexities of dynamically adding multiple applicants while keeping the code efficient and easy to maintain. The challenge of auto-filling shared fields between applicants required careful logic and a well-planned structure within the Angular framework. Each applicant's form was linked to the data flow, and the table displaying all submitted applications was enhanced with features like an application ID and consolidated applicant names, providing a much clearer and user-friendly interface.