Kendal Guizado

Milestone Four

For my Milestone Four enhancement, I focused on integrating MongoDB into the tenant screening application. This artifact, originally created as a simple Python program, has evolved into a full-stack application with a Node.js backend and an Angular frontend. The goal of this enhancement was to transition from a stateless system that only processed data in real-time to one that supports persistent data storage using MongoDB. This enhancement enables the system to store and retrieve tenant applications, making it more dynamic and scalable.

Artifact Overview

The tenant screening application evaluates rental applicants based on a series of financial criteria, including credit score, income-to-rent ratio, and employment length. Initially, the application only processed this information in memory and displayed the results directly without storing them. To address these limitations, we integrated MongoDB as the application's database. MongoDB was chosen for its flexibility and scalability, allowing the application to store applicant data, track submission history, and support future expansions like filtering and analytics.

The backend now includes the following key files and functionality:

- server.js: This file was updated to establish a connection to MongoDB using Mongoose. We defined a Mongoose schema to model the applicant data, allowing us to store information such as applicant names, credit scores, property addresses, and decision results. The schema design ensures consistency across all records, making the data structured and easily accessible. We also modified the existing API endpoint to save applicant data to MongoDB after calculating scores and decisions.

- tenant-screening.service.ts: This Angular service was extended to include a new getAllApplicants() method, which sends an HTTP GET request to retrieve all stored applicants from the backend. This allows the frontend to display stored applications and decisions.
- application-form.component.ts: This component was updated to fetch stored applications from MongoDB using the getAllApplicants() method when the component loads. The fetched data is displayed in a table, allowing users to see a history of all submitted applications, including names, dates, and property addresses.

These updates ensure that the application not only calculates eligibility scores but also persists data, making it a comprehensive solution for landlords and agents to manage tenant applications.

Justification for Inclusion

I selected this artifact for my ePortfolio because it demonstrates my ability to design, develop, and integrate a robust database solution into a full-stack application. This enhancement highlights my skills in data modeling, backend development, and the implementation of data retrieval mechanisms. Specifically, the use of Mongoose for schema definition and database interaction showcases my understanding of how to structure data efficiently. The enhancements align with my goals of demonstrating proficiency in database integration, secure data handling, and backend optimization, making this artifact a valuable addition to my ePortfolio.

The integration of MongoDB and the design of a Mongoose schema align with Course Outcome 4 by applying industry-standard practices in data management and designing a solution that leverages database technology. For example, the server.js file's API endpoint not only calculates the score but also saves it along with applicant details into MongoDB, reflecting best practices in software engineering. Additionally, by implementing error handling mechanisms and ensuring the security of database interactions, I have made progress toward Course Outcome 5, which emphasizes the importance of security and privacy in software architecture.

Skills Demonstrated and Course Outcomes

- Course Outcome 1 (Collaboration): Throughout this enhancement I organized the code structure to support team-based development. For example, the clear separation of responsibilities between the backend logic in server.js, frontend services in tenant-screening.service.ts, and user interface components in application-form.component.ts makes the project easier to manage and enhances collaboration. Additionally, documentation and comments in the code ensure that team members can understand the functionality and structure, making the project suitable for a collaborative environment.

- Course Outcome 3 (Algorithms and Data Structures): The integration of MongoDB required designing a suitable data model that supports efficient data retrieval and storage. The Mongoose schema in server.js defines each applicant's details, ensuring consistency and type safety during database operations. The design of queries to retrieve stored applications also required consideration of performance and query optimization, making the application capable of handling larger datasets as more applications are submitted. This aligns with the course outcomes by showcasing my ability to design and evaluate solutions that involve complex data handling and storage.
- Course Outcome 4 (Software Engineering/Database): The implementation of MongoDB involved structuring data using Mongoose, designing a schema for applicants, and integrating it seamlessly with the existing backend logic. The tenant-screening.service.ts file plays a crucial role in connecting the frontend to this new backend structure, ensuring that data flows smoothly between the Angular interface and the database. This reflects best practices in software engineering, as the database design supports the application's needs while allowing for future scalability. The transition to a database-backed application means the tenant screening system can now manage larger datasets and provide a more robust and scalable solution.
- Course Outcome 5 (Security): During the enhancement, I implemented secure data handling practices. This includes validating input before storing it in the database and implementing error handling to manage connection issues with MongoDB. For example, in server.js, the connection to MongoDB is wrapped in a

try-catch block, ensuring that the server gracefully handles connection errors and provides meaningful feedback in case of failures. By securing database access and ensuring that only valid data is stored, I have worked to mitigate potential security risks.

Reflection on the Enhancement Process

Integrating MongoDB into the tenant screening application was both challenging and rewarding. I applied recently learned concepts on how to design a database schema using Mongoose to structure data according to the application's requirements, and I wrote queries to retrieve and store data efficiently. One of the primary challenges was ensuring that the data model could handle multiple applicants while remaining flexible enough to accommodate future updates.

Another significant learning experience was understanding how to integrate the frontend with the backend for data retrieval. I had to adjust the Angular service in tenant-screening.service.ts to perform HTTP GET requests and update the application-form.component.ts to display data from MongoDB accurately. This required me to think critically about how data flows through the entire application, from user input on the frontend to storage in the database and back to data display.

By adding MongoDB, I transformed the application from a simple, stateless service into a full-stack solution that supports persistent data storage and retrieval. The updates in application-form.component.ts to display stored applications ensure that users can see the entire history of their submissions, making the system more valuable in a real-world context.