Akhmatov Timur

Project Description

Name: **RentGadget**

Home Appliance Rental System is a web-based application that allows users to place their own (household) appliances for rent and rent appliances placed by other users. It can be any appliance, be it a vacuum cleaner, a drill, a laptop, a portable speaker - anything, for example, that is not currently in use and you want to put it somewhere for a small, product-appropriate fee.

Below are the requirements that can be specified in the minimal version of the product must include requirements marked must, the rest will be marked should and could according to the MoSCoW methodology.

User stories and quality scenarios will be discussed and shaped in the negotiation session, also for the requirements described below there can be a wide range of decisions about the way of implementation, for example the system itself can be a simple web-app or telegram web-app, also for the implementation of any functionality it is possible and desirable to use ready-made solutions and APIs for them.

Key features:

1. User authentication and authorization. (**must**)
    a. Can be implemented using standard OAuth, OpenID or other mechanisms, for example, using third-party authentication/authorization services. (**must**)
2. Appliance catalog browsing and search functionality. (**must**)
    a. This can be implemented in the form of cards and a simple matching search mechanism. (**must**)
    b. To create a card, the user can enter the name and category of the appliance and attach a photo. (**must**)
    c. Going to a specific card should display a description and owner contacts.(**must**)
    d. As well as add the rental price and time period for which he will rent it. (**should**)
    e. Display the rating left by users for each user or his or her posted household appliances. (**should**)
3. Rental management system.(**should**)
    a. Add payment functionality. (**should**)
    b. Add a notification system for rental requests and the end of the rental period (**should**)
4. Feedback and review system (**could**)
    a. Users can leave feedback on items they have rented. This can help other users to make rental decisions, at least in the form of a five-point star scale. **(should**)
    b. Users can rate the feedback as helpful or not helpful. This will help highlight the most useful reviews. (**could**)
    c. Users can respond to reviews, allowing them to discuss their experience with the product. (**could**)

   d. Appliance owners can respond to reviews, allowing them to communicate with tenants and improve their products. (**could**)

Technology Stack

Backend: Python 3.11 using FastAPI framework.
Frontend: Streamlit.
Database: SQLite.
Version control system: GitHub.

Quality requirements

The quality attributes and the minimum threshold to be achieved are described below:

1. **Maintainability:** The code base should be well structured and documented to facilitate support and future improvements.
   a. Maintain a supportability rating of at least B by SonarQube analysis.
   b. Code complexity should not exceed a cyclomatic complexity of 10 per function.
   c. Interfaces should be documented
   d. Flake8 with no warnings
2. **Reliability:** The system must be stable and reliable under normal usage conditions.
   a. Maintain a reliability rating of at least B by SonarQube analysis.
   b. Ensure all critical functions have appropriate unit tests with code coverage of at least 60%.
3. **Performance:** The system must be responsive and performant to ensure a smooth user experience.
   a. Minimum Threshold: Response times for all API endpoints should be less than 200 milliseconds.
   b. Implement caching mechanisms to optimize frequently accessed data and reduce server load.
4. **Security:** User data and transactions should be handled securely to prevent unauthorized access and data leakage.
   a. Maintain a security rating of at least B by SonarQube analysis.
   b. Implement proper input validation and sanitization to prevent injection attacks.
   c. Utilize HTTPS protocol for secure communication between client and server.
   d. Bandit without critical vulnerabilities

CI Pipeline

1. Integrate SonarQube analysis into CI pipelines to automatically assess code quality and compliance to quality requirements.
2. Ensure code is analyzed for maintainability, reliability, performance, and security on every commit.
3. Configure automatic code inspections to detect and fix any quality issues early in the development process.