Simple Real Estate
Artifact Report

Authors: Nicholas Boland - Craig Lillemon

Contents

1 Design Purpose **1**

2 GitHub Link: **1**

3 License: **1**

4 Software Requirements **2**

5 Architecturally Significant Requirements: **3**

6 Diagrams: **3**

1 Design Purpose

- This product is a simple real estate management program, focused primarily on personal, non-corporate landlords. It shall provide capability to manage properties and rents associated. Focus on creating maintenance and expense reports, alongside generated profit reports based on rent and those expenses. It will be implemented as a SaaS (Software as a Service), Using Python with Django which enforces a MVT architecture.
- Intended Customers and Intended End-Users are personal, non-corporate landlords. Value created is simplicity in generating profit and expense reports based on given data.

2 GitHub Link:

`https://github.com/Nicholasgboland/CS4320-Semester-Project/tree/main`

3 License:

- The license we decided to use was the MIT Open Source License
- MIT License

4 Software Requirements

- *Functional requirements / Use-cases:*
  - UC1: Add/Remove Properties
  - UC2: View/Modify Properties
  - UC3: Add/Remove Units
  - UC4: View/Modify Units
  - UC5: Add Maintenance Records and Quotes
  - UC6: Add Expenses
  - UC7: Generate Profit/Expense Reports
  - UC8: Add/Remove Tenants
  - UC9: View/Modify Tenants
  - UC10: Add Rental Agreements
  - UC11: View/Modify Rental Agreements
- *Functional Requirements / Use-cases implemented in the prototype:*
  - UC1: All features implemented
  - UC2: All features implemented
  - UC3: All features implemented
  - UC4: All features implemented
  - UC5: Ability to add Maintenance Records implemented, but not to add Quotes
  - UC6: All features implemented
  - UC7: Ability to generate Expense Reports implemented, but not to generate Profit Reports
  - UC8: All features implemented

  - UC9: All features implemented
  - UC10: All features implemented
  - UC11: All features implemented

- *Quality Attributes / Non-Function Requirements:*
  - QA1: Security
  - QA2: Usability
  - QA3: Manageability
  - QA4: Modularity

- *Architectural Constraints:*
  - CON1: The application must be accessed via a web browser
  - CON2: User must have an active network connection
  - CON3: Prior reports must be stored and accessible
  - CON4: Modular design allowing future modifications with ease

- *Architectural Concerns:*
  - CNR1: Developing a ground up system
  - CNR2: Leveraging team's knowledge in Python, Django, HTML, and SQLite
  - CNR3: Work allocated to small development team

5 Architecturally Significant Requirements:

- Functional ASRs: UC1, UC2, UC3, UC4, UC8, UC9, and UC10
- Non-Functional ASRs: QA4
- Architectural Constraint ASRs: CON1 and CON4

6 Diagrams:

- The four UML diagrams chosen were the Use Case Diagram, Model/Class Diagram, Sequence Diagram, and the Activity Diagram.
- All diagrams delivered as separate PNG files in the GZIP file containing this document.