**📘 Software Requirements Specification (SRS)**

**Project Title: Automated Property Valuation Assistant**

**Prepared by:** Sharon Mwangi  
**Registration No:** 24/01195 BSD  
**Supervisor:** Purity Mureithi  
**Institution:** KCA School of Technology  
**Date:** June 2025

**Table of Contents**

1. Introduction
2. Overall Description
3. Specific Requirements
4. System Features
5. External Interface Requirements
6. Non-functional Requirements
7. Assumptions and Dependencies
8. Appendices

**1. Introduction**

**1.1 Purpose**

This document specifies the software requirements for the **Automated Property Valuation Assistant**—a web-based AI application that provides real-time property value estimates using machine learning and real estate data.

**1.2 Intended Audience**

* Project Supervisor
* Real estate agents and buyers
* Software developers and testers
* Academic assessors

**1.3 Scope**

The system will take inputs such as property location, size, number of rooms, and amenities, and return a real-time estimated price based on machine learning models trained on verified datasets. It will also allow users to interact with the valuation system via a web interface.

**2. Overall Description**

**2.1 Product Perspective**

The product is a standalone web application developed using Python, HTML/CSS, Flask/Django (backend), and React/JavaScript (frontend). It integrates with cloud-based services and APIs for real estate data.

**2.2 Product Functions**

* Accept user inputs for property attributes
* Fetch and preprocess data from APIs
* Use trained ML models to calculate property value
* Display results in a user-friendly interface
* Allow user feedback for refinement

**2.3 User Classes and Characteristics**

| **User Type** | **Description** |
| --- | --- |
| Buyers & Sellers | Non-technical users needing price guidance |
| Agents | Users needing fast estimates for clients |
| Admin/Developer | For model training, data updates, debugging |

**2.4 Operating Environment**

* Web-based interface (desktop and mobile browsers)
* Hosted on AWS or GCP
* Python backend (Flask or Django)
* TensorFlow, Pandas, Scikit-learn

**3. Specific Requirements**

**3.1 Functional Requirements**

* **FR1:** The system shall allow users to input property data (location, size, features).
* **FR2:** The system shall return an estimated property price.
* **FR3:** The system shall validate user input.
* **FR4:** The system shall allow users to submit feedback on estimate accuracy.
* **FR5:** The system shall log all valuation requests and feedback for analysis.

**3.2 Data Requirements**

* Property transaction datasets (location-based)
* Market trend APIs (e.g., Zillow, local listings)
* GIS data for location accuracy
* Socio-economic indicators (optional enhancement)

**4. System Features**

| **Feature** | **Description** | **Priority** |
| --- | --- | --- |
| Smart Valuation | Use of ML models for pricing | High |
| Real-time Feedback Loop | Learn from user corrections | Medium |
| Interactive UI | Responsive and user-friendly | High |
| Location-Aware Estimates | Use GIS for contextual pricing | High |

**5. External Interface Requirements**

**5.1 User Interface**

* Clean input form with dropdowns and map location selector
* Result page with predicted price, confidence level, and comparison to market trends
* Feedback form for corrections or comments

**5.2 Hardware Interface**

* None beyond standard user devices (laptops, phones)

**5.3 Software Interfaces**

* Real estate listing APIs
* Machine learning libraries (TensorFlow, Scikit-learn)
* Database (PostgreSQL or Firebase)

**6. Non-functional Requirements**

| **Category** | **Requirement** |
| --- | --- |
| Performance | Return predictions in under 5 seconds |
| Security | Data encryption for API calls |
| Usability | Simple UI for non-technical users |
| Maintainability | Modular architecture for easy updates |
| Reliability | System uptime > 95% |

**7. Assumptions and Dependencies**

* Sufficient training data will be available via APIs or public datasets
* Users have access to the internet
* The machine learning model will be able to generalize well across different regions

**8. Appendices**

**A. Acronyms**

* **API** – Application Programming Interface
* **ML** – Machine Learning
* **GIS** – Geographic Information System
* **UI** – User Interface

**B. Tools and Technologies**

* Python, Flask/Django, Scikit-learn, TensorFlow
* HTML, CSS, JavaScript (React)
* Google Maps API or OpenStreetMap for geolocation
* Cloud hosting (AWS, Firebase, or GCP)