Smart Real Estate Investment Dashboard

Project Management Plan

Department of Information Technology and Management

Cyber Security Technologies (ITMS 548)

Prof. Dr. Maurice Dawson Jr.

Illinois Institute of Technology

Authors

Aryan Deshmukh

Tanishq Joglekar

Mayur Koli

Sudhanshu Panda

Table Of Content

1. **Introduction**
    1.1 Project Overview
    1.2 Scope Statements
    1.3 Goals and Objectives
    1.4 Stakeholders and Key Personnel
2. **Project Organization**
3. **Acquisition Process**
4. **Monitoring and Control Mechanisms**
5. **Systems Security Plans and Requirements**
6. **Work Breakdown Structure (WBS)**
7. **Project Success Criteria**
8. **Communication Management Plan**
9. **Risk Management Plan**
10. **Software Configuration Management (SCM) Plan**
11. **Training Plan**
12. **Quality Assurance Plan**
13. **Project Measurement Plan**
    13.1 Description
    13.2 Performance Measurements
14. **Reference Materials**

1. Introduction

1.1 Project Overview

This project is to build an open source real estate insights application by Python with Streamlit. The app will let users search through properties with filters, and view more information on data like weather, nearby restaurants, and crime statistics. Combining static and real-time data sources, the solution immerses the user in a dynamic and enlightened walk through properties.

1.2 Scope Statements

- **In-Scope**:
  - Build a web application with Streamlit.
  - Combine the information of four sources: real_estate. csv (local data), Open-Meteo API (weather), Foursquare API (restaurants), and Chicago Crime API (crime).
  - Introduce data filtering, plotting and dashboard capabilities.
  - Make sure your application is available through the web.
  - Keep up project codebase on Github with documentation.).
- **Out-of-Scope**:
  - Commerce, or booking, features.
  - Real-time property listing updates outside of the local data set.
  - Development and deployment of mobile applications.

1.3 Goals and Objectives

- Develop an easy-to-use, visually appealing real estate experience.
- Empower users to act with a single version of the truth.
- Foster the development of open-source projects.
- Use data analysis methods to discover and nation the data.

1.4 Stakeholders and Key Personnel

- **Stakeholders**:
  - Educational community, including students and teachers in the following.
  - Those hoping to buy, rent or get a leg up on the real estate market.
  - App developers and analysts.
- **Key Personnel**:

- **Project Manager:** Manages planning, execution progress, and team coordination.
- **Backend Developer:** It is responsible for API integration and data processing logic.
- **Frontend Developer:** Writes the Streamlit GUI and makes it responsive**.**
- **Data Analyst:** Takes care of data cleaning, processing and producing visualizations.
- **QA Tester:** Functions as stage for security, functionality and usability testing.

2. Project Organization

- **Development Team:** Responsible for API integration, data handling, and building backend logic.
- **Design Team:** Focuses on crafting user interfaces and enhancing usability.
- **Quality Assurance Team**: Tests the platform for bugs, inconsistencies, and security vulnerabilities.
- **Project Management:** Oversees overall project flow, risk mitigation, and team communication.

3. Acquisition Process

- **Programming Language:** Python
- **Framework:** Streamlit for GUI, Pandas for data processing, Plotly for visualizations.
- **Data Sources:**
  - real_estate.csv (local property listings dataset)
  - Open-Meteo API (weather)
  - Foursquare Places API (restaurants)
  - Chicago Open Data API (crime statistics)
- **Hosting:** Application to be hosted locally initially, with options for cloud deployment using Streamlit Cloud.

4. Monitoring and Control Mechanisms

- **Task Management:** Organize project tasks, assignments and due dates with Trello.
- **Progress Reviews**: A review of progress and to discuss blockers over Zoom each week.
- **Issue Tracking:** GitHub Issues (report bugs here, also requests, etc).
- **Version Control**: GitHub: branches, pull requests, etc- for consistent collaboration.

5. Systems Security Plans and Requirements

- **API Security**: All calls made for external data must be accompanied by the use of HTTPS.
- **Management of Credentials**: API keys and other private information stored in environment variables.
- **User Privacy**: User's personal data won't be collected.
- **Secure Your Code:** Validate user input and handle errors to avoid injection.

6. Work Breakdown Structure (WBS)

Work Breakdown Structure (WBS)

1. **Planning**
   1.1 Analysis and documentation of requirements.
   1.2 Dataset and API resource curation.
2. **Design**
   2.1 Develop wireframes idea for UI of Streamlit.
   2.2 Data architecture design.
3. **Development**
   3.1 Developing the Streamlit app framework.
   3.2 Combine local and API sources of data.
   3.3 Add dynamic filters and dashboards.
4. **Testing**
   4.1 Perform unit, integration and system testing.
   4.2 Undertake user acceptance testing (UAT)).
5. **Deployment**
   5.1 Finalize application deployment.
   5.2 Prepare end-user documentation.

7. Project Success Criteria

- Complete integration of four data-via sources.
- Display information in an accurate and timely manner.
- Intuitive user interface that reacts to users behavior.
- Positive UAT session feedback.

8. Communication Management Plan

- **Daily Updates**: Group chats on WhatsApp.
- **Weekly Gatherings**: Checking in on Zoom.
- **Document Sharing:** Google Drive for centralized documentation storage.
- **Development Communication:** GitHub Issues and Pull Requests for task tracking and code reviews.

9. Risk Management Plan

- **Key Risks**:
  - API quarantine, API key limitations or failures.
  - Inconsistency or latency in data.
  - UI/UX problems which drive the user drop-off.
- **Mitigation Strategies**:
  - Fallback on other APIs.
  - Validation of the data at every integration point.
  - Execute usability tests and adapt for feedback.

10. Software Configuration Management (SCM) Plan

- **Version Control**: GitHub wants to use separate main, dev and feature branches.
- **Merging Process**: Pull requests should be reviewed before merging.
- **Continuous Integration:** Manual testing on the dev side.

11. Training Plan

- Write README with all the setup to be done step by step.
- Create a step-by-step video showcasing the main features and use-cases.
- Include an FAQ to answer the most asked questions about setting up and using MPD.

12. Quality Assurance Plan

- **Testing Types**: Unit tests for components, integration tests for APIs, UI tests for usability.
- **Automation:** Testing scripts for data retrieval reliability.
- **Bug Tracking:** Maintain an open GitHub issue tracker.

13. Project Measurement Plan

13.1 Description

The performance of the project will be judged by the stability of the system, the accuracy of the data's portrayal and the reaction time of the GUI as well as the feedback of the users from the test session.

13.2 Performance Measurements

- The API response rates during the stress tests are 95%+.
- An interface load time of under 3 seconds.
- User rating of 85% or higher in survey.
- Performance of each major element as defined in WBS.

14. Reference Materials

- Open-Meteo API Documentation - open-meteo.com
- Foursquare Places API Documentation - https://location.foursquare.com/developer/
- Chicago Open Data Portal - https://data.cityofchicago.org/
- Streamlit Official Documentation https://docs.streamlit.io/
- Pandas and Plotly Official Guides.