Smart Mirror Architecture Document

Table of Contents

1.0 Introduction

The Smart Mirror Architecture Document is designed to show and identify the architecture systems used to design and implement the Smart Mirror project.

The document contains an overall view of the system hierarchy, logical views of the system components, and a process view of the system's communication.

2.0 High Level Hierarchy

2.1 Hierarchy Diagram

2.2 Hierarchy Description

The architecture system for the Smart Mirror application is architected so that user information is private and so that widgets can be customized. This architecture system is

designed to allow the communication of information between the various widgets so that they are coordinated, but can also function as individual system dependencies. The user interface (UI) is simple so as to allow immediate interaction. There are several levels

between the level of presentation and the lowest level, due to the significant challenges present in the optimization and control of process design. The Database level is the lowest level of the hierarchy.

3.0 Components Classification

3.1 Presentation Layer

Purpose: To display forms, controls, images and widgets to create a fluid and efficient user experience.

Specific Nature: The presentation level will be responsible for showing the user appropriate images, menus and widgets. This layer provides an interface to show data in specific forms as defined. When a user enters the graphical interface, the code corresponding to that event will be called and display the widget.

Subcomponents: TKinter, Receiving data

Tkinter – The Tkinter subcomponent is a library of type Image Viewer that is used when widgets need to be displayed for the user. Its responsibility is to display the app image

Receiving data – The Receive data subcomponent is responsible for the action of receiving location, weather and time data based on the geographical position, therefore this subcomponent sends the data to the lower level to allow the code to store this information and process it directly on the requested widget.

3.2 Controller Layer

Purpose: Processes and responds to events, typically user actions, and may invoke changes on the model.

Specific Nature: The controller layer in our project will be in charge of collect information received form the lower layer or from the higher layer. It is responsible near the interpretation of the gesture of the user at the presentation level and it is responsible about the data and record that comes from the bottom.

Associated Constructs: StartupScreen Controller

- StartupScreen Controller – StartupScreen Controller class will control all the canva, from the position of the various widgets to the graphical interface that the user can use to interact to receive the necessary information

3.3 Business Layer

Purpose: This layer is in charge of the heavy algorithm business logic found in complex solutions.

Specific Nature: This layer will be used to compute the algorithm for finding the user's current position and the the data for the various widgets

Associated Constructs: Widget Manager, User Manager, Error Manager

- **Widget Manager** – Widegt Manager class will consist will consist of only properties describing a Widget.
    - *Widget Manager*
        - Various functions will provide the user with the various data.
- **User Manager** – User Manager class will consist of a method to find the current user position.

- *User Manager*
    - *User_Position()* – Will compute the current user position. Will return a *User Record* data type.

- **Error Manager** – Error Manager class will consist of a method to find the current error.
    - *Error Manager*
        - *Get Error()* – Will return the current error depending on the actions of the user. Will return an *Error Record* data type.

3.4 Record Layer

Purpose: This layer is in charge of containing the classes that strictly consist of data.

Specific Nature: This layer will be used to store User data, Landmark data, and Error data and Location data. These classes will only contain properties (variables) that describe each data type.

Associated Constructs: User Record, Widget Record, Error Record

- User Record – User Record class will consist of only properties describing a user information. This class will be static, meaning there is only one copy of this class in memory once initialized until the end of the program.
    - User Record
        - Weather_Frame – It will keep the type of user who is using the device. The possible values are technical and non-technical. The function will be able to provide the weather and temperature in a detailed way but that is understandable even if you are a less experienced user. This function is string type but also numeric integer type.
        - Date_Frame - The function can automatically provide the current time according to the user's location. This function is numeric integer type.
        - Position - The function detects the longitude and latitude of the user. This will be of type string.
- Widget Record – Widget Record class will consist of only properties describing a Widget.
    - Widget Record
        - Newsreader – Function that detects the main news of the day between national and world newspapers, according to the settings. This function is of type string.
        - Class_Schedule – Shows the user's school or work hours on the canvas, taking the information from a JSON file. This function is string type but also numeric integer type.

        - Apod_Dict – Receives the raw data and transmits it to JSON files. This property is of type string.
        - StartupScreen – Initialize the canvas on which the widgets will go.
- Error Record – Error Record class will consist of only properties describing an error.
    - Error Record
        - Name – Will hold the name of the error. This property is of type string.
        - Description – Will hold the description of the error. This property is of type string.

3.5 Data Access Layer

Purpose: This layer is in charge of communicating to the database. This layer should handle all of the database transactions and connectivity.

Specific Nature: This layer will be in charge of communication to our database which will in turn lead to populating the record layer.

3.6 Database Layer

Purpose: This layer is in charge of storing data in persistent storage.

Specific Nature: This will be our database management system. There will store Data User, Widgets features, Positions, Errors, and eventually Language Support.