Architecture Document

SF-Agriculture

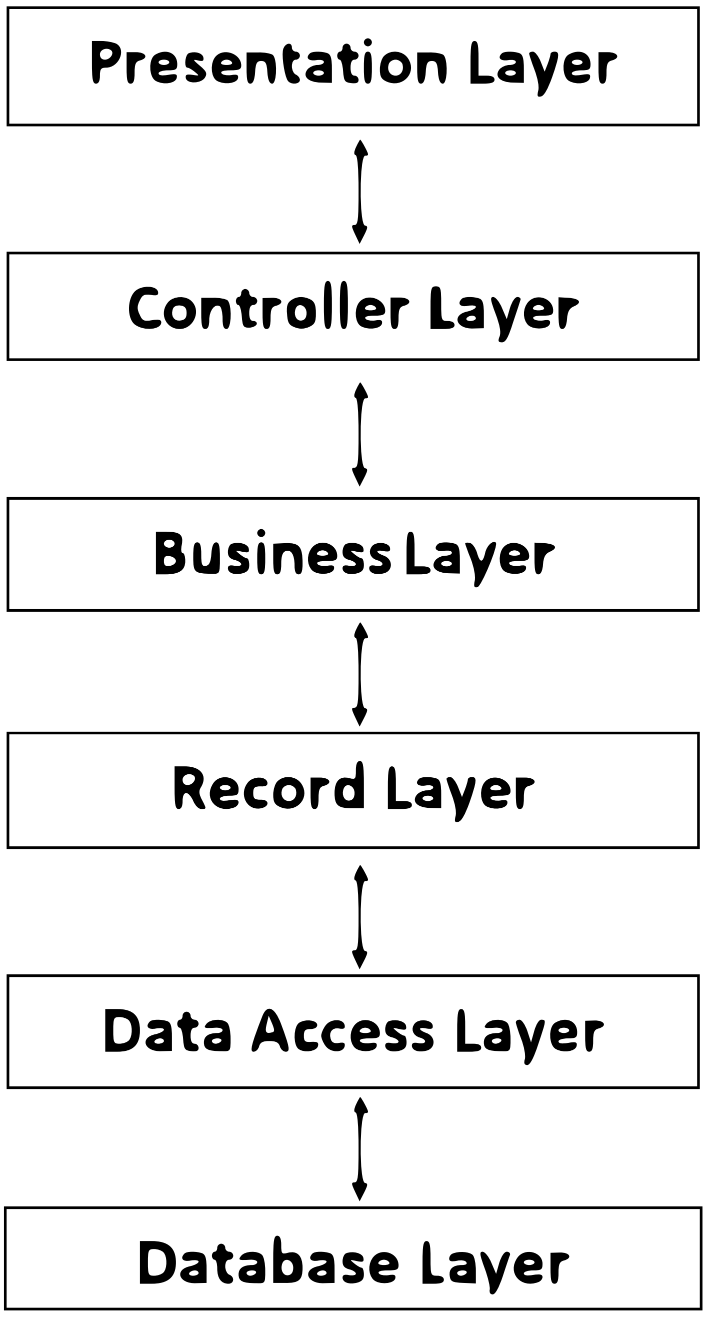
Spring 2020

1.0 Introduction

The SF-Agriculture Architecture Document is designed to illustrate and identify the high level architecture systems used to design and implement our application. 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 SF-Agriculture application is an n-tier architecture. This architecture system is designed to allow for proper information hiding, modular components, and single system dependencies. The abstraction of the presentation layer, and consequently the Interface, allow for a flexible pipeline for the optimization of the graphic interface to meet customer needs and expectations. The Database layer is the last level in the hierarchy.

3.0 Components Classification

3.1 Presentation Layer

Purpose: Create the interface that will be available later for the user

Specific Nature: The level relating to the graphical interface will be responsible for displaying images and menus. This level will also be responsible for managing stylus clicks. This layer will also be responsible for generating appropriate threads. The need to generate extra threads is due to the fact that the app's main thread will check for clicks on events, but we also need another thread that is constantly running to get the position of the field in question.

Subcomponents: Current State, Field Viewer

Current State – The Current State will be a global class that will get updated by the Presentation Thread. The Current State class will be read from the main thread of the app at a specified time interval as governed by a timer.

Field Viewer – The Image Viewer subcomponent is used during the Walking Tour and during the Interactive Map mode of the application. Its responsibility is to display the appropriate image as determined by the Landmark held in the Current State.

3.2 Controller Layer

Purpose: Responds to user events

Specific Nature: The controller layer in our program will be in charge of getting the closest landmark to the current user position. It will do this on a constant interval of k. k is a 10 second time step that will be determined through testing. This layer populates the closest landmark based off of user input. In this case, user input is the user walking around. This will notify the presentation layer when the closest landmark has been updated.

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 field's position. This algorithm will be located in a class called FindField. The algorithm to find the fields will be located in a class called FieldPosition. This layer will also contain a class called Error Manager. This class is in charge of getting the appropriate error message based on the actions of the user.

3.4 Record Layer

Purpose: This layer is in charge of containing the classes that strictly consist of data. Little to no functional methods will be found in these classes.

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

3.5 Data Access Layer

Purpose: This level has the task of communicating with the database and manages database connectivity.

Specific Nature: Our database in this project will consist of XML files and serial device data from satellites. For the use of Google Maps. You will need to create a class for managing communication with Google Maps. The ReadData class will be used to read the XML file of the landmarks and populate the classes based on the data in the XML file. The User class will be used to initialize a User Record class. The user's class will only deal with initializing the User Record class to null since we won't store different types of users in an XML file. The Error class will be responsible for reading an XML file containing any logged connection errors.

3.6 Database Layer

Purpose: This layer is in charge of storing data

Specific Nature: This level represents the stable part of the application and will be managed with the use of XML files. These together will be our database management system. These XML files will store reference points, errors and data regarding the location of the field and the temperature.

4.0 Process View

4.1 Process View Description

The vision of the process is essential to understand how the components communicate with each other in a concurrent application. By better understanding the necessary communication paths between the components, it may be possible to optimize the data flow and storage of the application.

4.2 Application Thread

The main thread is created at program run time (this is not a user created thread). This thread manages the flow of the base program by controlling the navigation between the modules and processing the window events.

4.3 Presentation Thread

This thread is user created, when the application enters the Walking Tour mode. This thread is responsible for seeking the nearest landmark and then relaying this information to the Presentation Layer, where this thread requests that Image Viewer and Audio Player subcomponents update the current presentation when necessary.

4.4 Connection Thread

This thread is created by the user and has the task of always checking whether the connection to Google Maps for viewing the fields from above. This thread also has the task of keeping track of the seconds of time in which the connection was not valid. If the connection has not been valid for some time, we will notify the user and try to reconnect.

4.5 Google Maps Thread

This thread will track the location of the field and it will provide our program with all the necessary information on the location of the plot of land. The connection thread will constantly check with this thread for a valid connection.

4.6 Device Thread

This thread will always read the current state of the device, if the state of the device has changed, it will generate an event that notifies the change of the state of the device, with the result that the Google Maps thread updates the current position.