Commerce Bank Team two Architecture Document

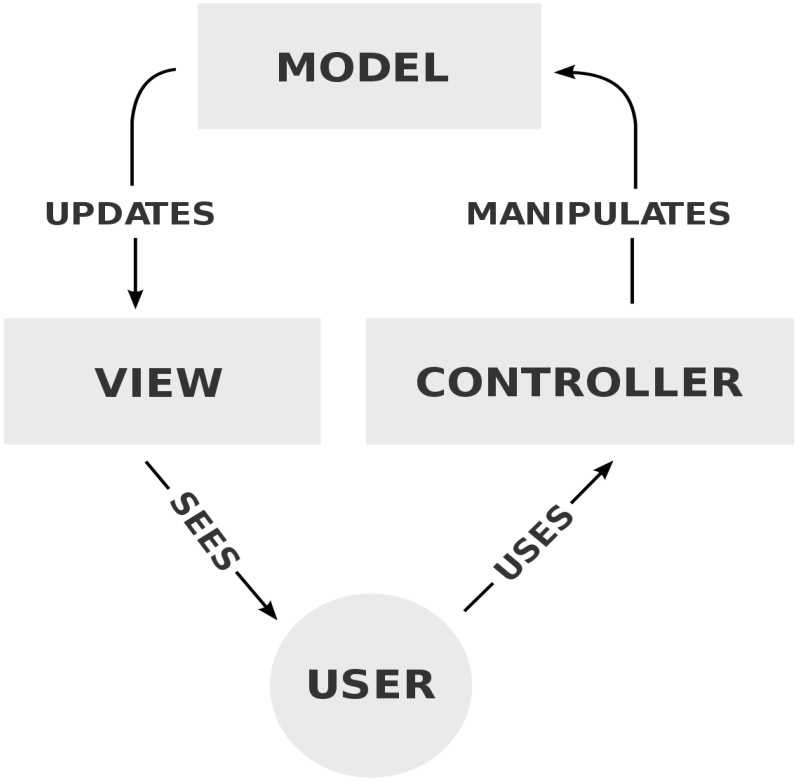
Table of Contents

1.0 Introduction

The Commerce Bank Team two Architecture Document is designed to illustrate and identify the high level architecture systems used to design the Commerce Bank notifications application. This document will contain an overview of the application hierarchy, logical views of the system components, and a process view of the system's communication.

2.0 High Level Hierarchy

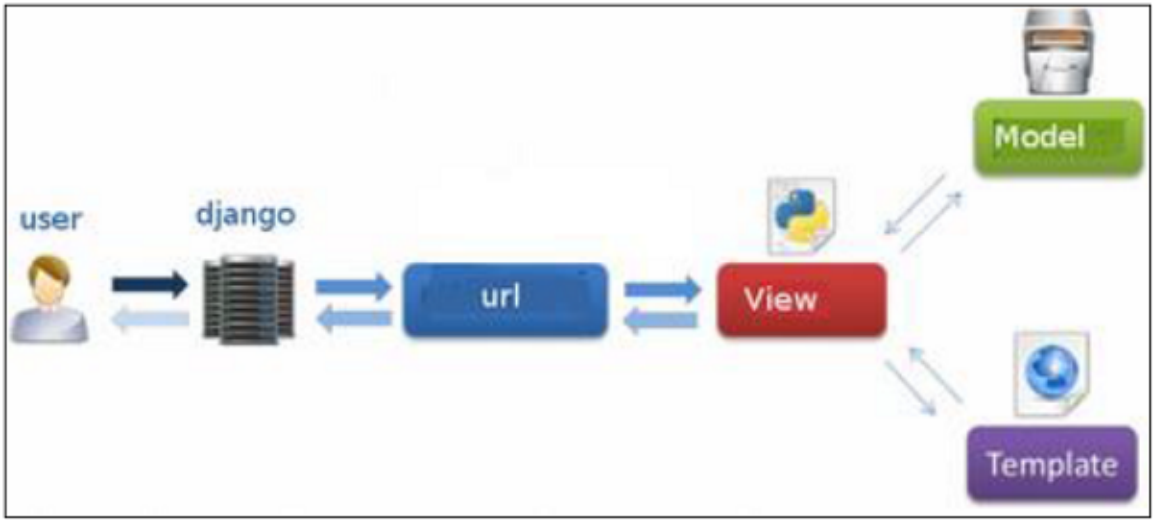
2.1 Hierarchy Diagram



2.2 Hierarchy Description

The architecture system for the application is an MVC (or technically model view template since Django itself takes care of the controller differently) architecture. MVC is commonly used for developing web applications and graphical user interfaces many popular programing languages (Javascript, Python, C# and more) have MVC frameworks that can be used for mobile or web development right out of the box. Using Django the developers provide the model, view and template (template being an HTML file mixed with Django template language) and the framework itself abstracts the controller portion of the MVC architecture. Models are the lowest level responsible for maintaining data, the view is responsible for displaying all or a portion of the data to the user, and the

controller controls the interaction between the model and view (this part is contained within the Django framework itself but traditionally would be custom code). This architecture separates application logic from user interface layout and helps support separation of concerns.



(How django slightly differs from traditional MVC)

3.0 Components Classification

3.1 Model Layer

Purpose: Represent how data is organized and allow data manipulation

Specific Nature: The model layer is the established connection to the database. It contains the fields of the database and the behaviors of the data. It also allows the manipulation of the data from the database and passes that information back and forth between the application and the database. In our application, the model holds the connection to our database which will also allow the manipulation of said data and will communicate directly to the controller for instructions.

3.2 Controller Layer

Purpose: The controller controls interactions between the the model and view

Specific Nature: The controller layer in MVC is responsible for providing updates to what the user sees as it is updated in the database and facilitating any interaction between the front and back end of the application. This can mean updating GUI, verifying logins,

getting transaction positions etc. This layer can be thought of as the “middleware” and will be responsible for making API calls.

3.3 View Layer

Purpose: This layer is responsible for displaying all or a portion of the data to the user. Contains any business logic that refers to specific templates (this part is Django specific).

Specific Nature: This layer acts as the bridge between the model layer and the Django templates. Django suggests that the view layer should contain the business logic instead of the logic used to present information to the users contrary to the traditional MVC architecture. In Django's modified MVC architecture the template layer will handle the presentation logic. The view layer also acts as the controller layer in a traditional MVC layout.

3.4 Django Templates

Purpose: This layer is not typically found in a traditional MVC layout and is specific to Django (sometimes called Model, Template, View + Controller). Templates move the presentation logic traditionally present in the View layer to the templates

Specific Nature: This layer will be used to provide everything that is presented to the users. HTML code will be stored in the templates as well as special syntax that will describe how dynamic content will be rendered.

4.0 Process View

4.1 Process View Description

The Process View is essential in understanding how the separate components and subcomponents communicate with each other in a concurrent application. By better understanding the necessary paths of communication between the components, it may be possible to optimize the data flow and storage of the application, as well as ensuring thread-safety.

4.2 Application Thread

This thread is the main application thread that is created at runtime of the program. The program creates the thread; this is not a user created thread. This thread handles the basic program flow by controlling navigation between forms, and processing window events, including the handling of user input to the graphical forms.

4.3 Presentation Thread

This thread is user created, when the application enters the home screen. This thread is responsible for providing the user with the information that will be provided to them about their account. This information will be provided by the Django templates

4.4 View Thread

This thread is user created. This thread is in charge of checking and updating the user information based upon account activity. Actions such as interstate transactions will activate this thread updating the user upon the action taken with their account affecting the presentation thread. The activations of this thread may depend on a users individual notification settings.

.