Application Design & Django Architecture

Design phase of SDLC

Transforms business requirements into forms able to be implemented using code

These forms are called **design artifacts** – includes wireframes, user stories, ER diagrams, etc.

Several layers of design, should decide programming language early on

High-level (architectural) design specifies major system components and how they interact; addresses issues such as scalability and maintainability

Low-level (detailed) design breaks down major components into smaller units; decides each unit's purpose and how they interact; algorithms and data structures used

Architecture patterns & design patterns

Provide programming-language-agnostic solutions
to common problems

Architecture patterns are concerned with overall application's structure (*high-level design)*

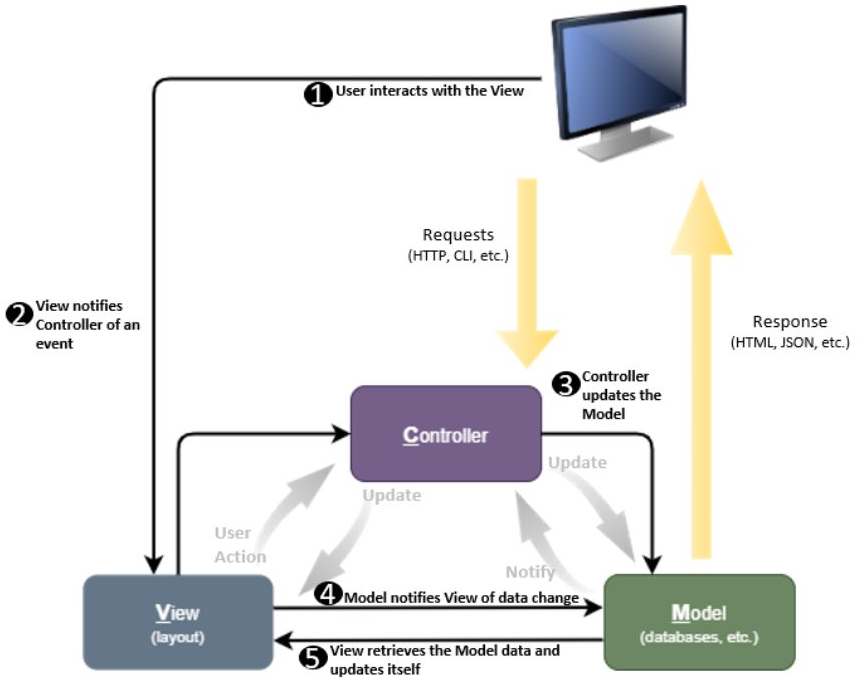
Design patterns are concerned with specific technical problems (*low-level design)*
- e.g. Observer design pattern provides solution
for having objects subscribe to state changes in other objects

Model-View-Controller (MVC)

Model: Set of classes representing data, contains business logic to describe how data can be manipulated, updates Controller when data changes

View: User interface where user interacts with app, visual layout of data from Model

Controller: Controls data flow into Model, updates View when data changes, mediates between View and Model

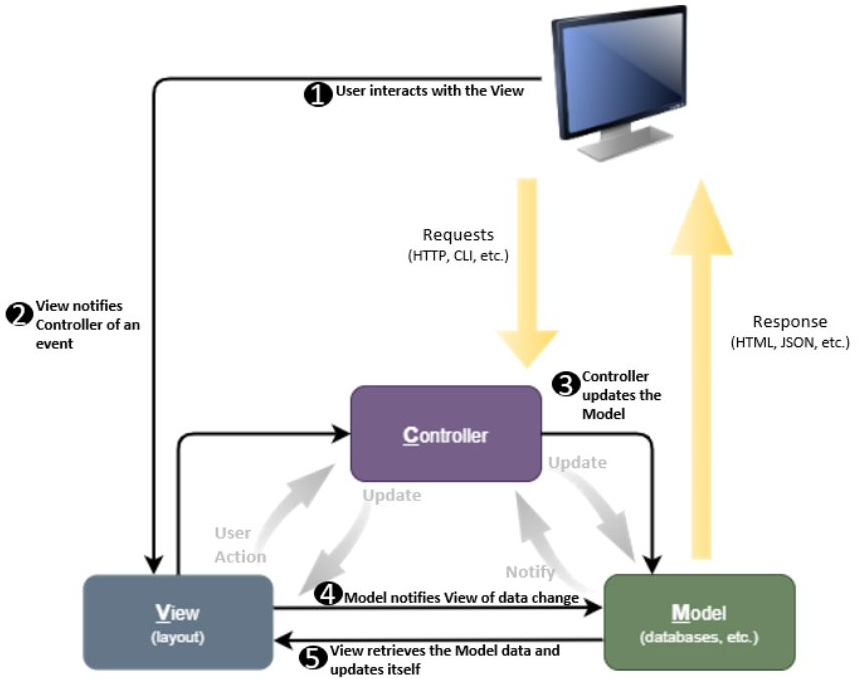


Model-View-Controller (MVC)

Model: Set of classes representing data, contains business logic to describe how data can be manipulated, updates Controller when data changes

View: User interface where user interacts with app, visual layout of data from Model

Controller: Controls data flow into Model, updates View when data changes, mediates between View and Model



MVC vs Django Framework MVT

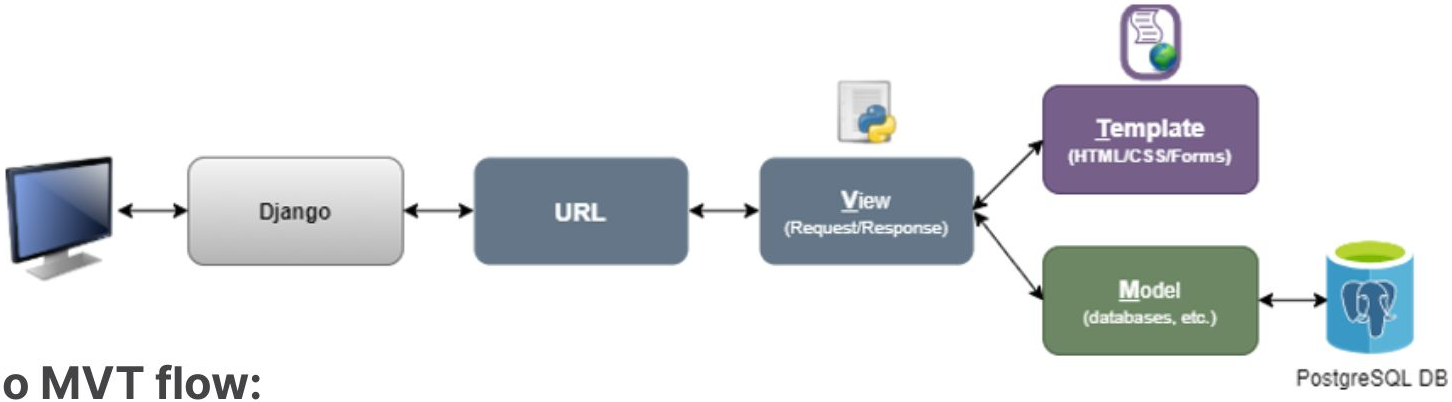
Django uses the **MVT** pattern – **Model View Template**

Model: Same as in MVC

View: Like the *Controller* in MVC – controls data flow to Model, mediates between Model and Template

Template: Like the *View* in MVC - generates the HTML pages that the user sees and interacts with

Model-View-Template (MVT)



Django MVT flow:

User makes **request** to Django app

App uses defined **URL patterns** to determine which View to use

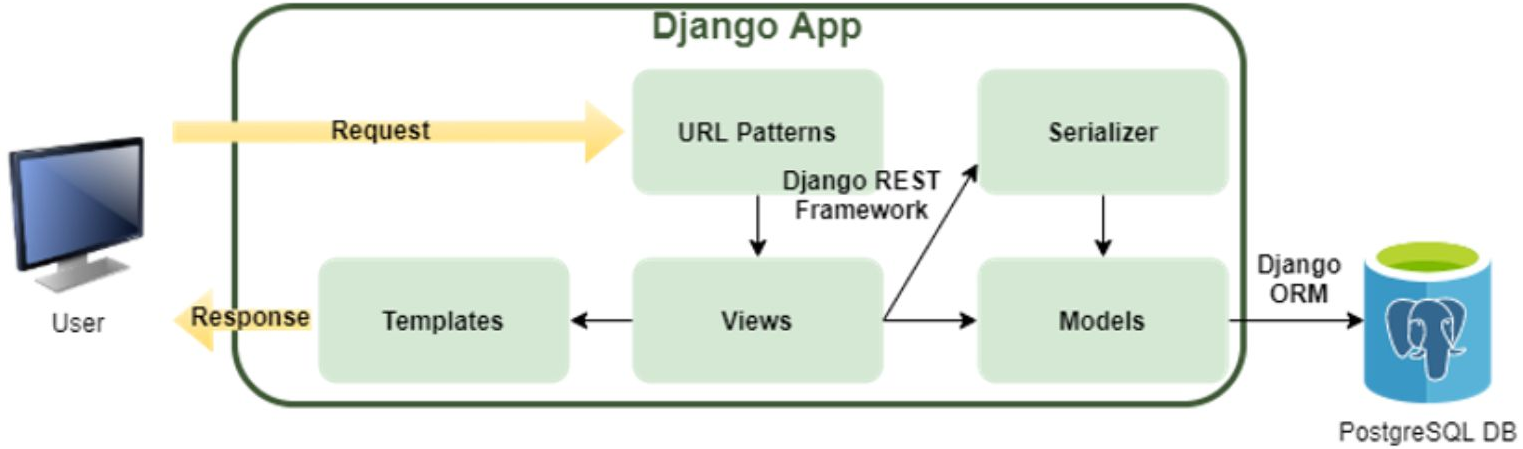
View determines which Models are needed, passes request to Models

Models interact with database through built-in Django ORM, return data to View

View sends data to Template

Template generates HTML pages that are passed back to user

Django Rest Framework (DRF)



Basic Django app uses **Django Web Framework**
Django REST Framework (DRF) is added to produce **RESTful APIs**
Helps produce **endpoints** that allow access/updates via RESTful API
Also **serializes** data returned in HTTP response
(transforms to RESTful format such as JSON or XML)