**NANYANG TECHNOLOGICAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE and ENGINEERING** 

Software System Analysis and Design (CZ3003) Assignment

Name-Kirath Singh

Matriculation Number – U1822329J

Email address – [KIRATH001@e.ntu.edu.sg](mailto:KIRATH001@e.ntu.edu.sg)

Choice of open-source project – Django (built using python)

**Table of Contents Page number**

1. Introduction 2 to 3
   1. Need for a framework
   2. Source of Information
   3. Files/Folders used in the overall architecture of Django
2. Analysis Models of Django 4 to 6
   1. Use case diagram
   2. State Transition diagram
   3. Context diagram
3. Architecture diagram 7
4. Quality attributes addressed by the design 8
5. Design Style 9
6. Discussions 9 to 10

**1.0 Introduction**

Django is a high-level, open-source, Python Web framework that encourages rapid development and clean, pragmatic design. It is a robust and approachable framework that lets the developer focus on the application by having salient parts pre-baked that are fairly standard practice in almost all web-based applications.

Django was created in 2003 when web developers working for a newspaper company in Lawrence, Kansas, needed a better way to organize their code. Because the developers were surrounded by many newspaper journalists, clear documentation became an integral part of the project that became known as the “Django” project. Django has since blossomed into a massive online open-source community that has a solution for just about everything a developer can think of, from authentication to content management systems.

Django relies on the Model View Template architecture as opposed to the usual Model View Controller architecture used by most of the open-source web application development frameworks. The main advantages of using Django are: -

1. **Ridiculously fast –** Django was designed to help developers to take applications from concept to completion as quickly as possible
2. **Reassuringly secure –** Django takes security seriously and helps developers avoid many common security mistakes.
3. **Exceedingly Scalable –** Web applications built using Django are highly scalable and some of the busiest web-based applications leverage Django’s ability to quickly and flexibly scale.
4. **SEO optimised –** Django makes SEO easier by maintaining a website through URLs rather than IP addresses.
5. **Documentation -** Django’s documentation is one of the best on the market. It’s easy to read, even for people with no technical background.

Due to Django’s versatility , it is not being used by many companies, including Instagram, Pinterest, Mozilla, and Eventbrite. Companies everywhere are actively using Django and have invested in its development. In 2020, it is the 4th most wanted web framework according to StackOverflow.

**1.1 Need for a framework**

To understand what Django is for, we need to take a closer look at the servers. The first thing is that the server needs to know that you want it to serve you a web page.

Imagine a mailbox (port) which is monitored for incoming letters (requests). This is done by a web server. The web server reads the letter and then sends a response with a webpage. But when you want to send something, you need to have some content. Django is the one that helps us in creating the content.

When a request comes to a web server, it's passed to Django which tries to figure out what is requested. It takes a web page address first and tries to figure out what to do. This part is done by Django's **URL resolver**. It is not very smart – it takes a list of patterns and tries to match the URL. Django checks patterns from top to bottom and if something is matched, then Django passes the request to the associated function declared in the **view**. The view takes care of all the processing that needs to be done with the data as well as the response that needs to be sent in the form of a HTML template.

**1.2 Source of Information**

The information for Django has been gathered from the official documentation of Django which was created during the launch of Django and has been updated on a regular basis especially when a new feature or an update is added to this open-source framework. The official documentation can be found at <https://www.djangoproject.com/> . Another source of information is an online blog written by Christiana Kopecky, the blog can be accessed at <https://www.educative.io/blog/what-is-django-python> .

**1.3** **Files/Folders used in the overall architecture of Django**

* **urls.py -**Itcontains project-level URL configurations. This module is pure Python code and is a mapping between URL path expressions to Python functions (your views). This mapping can be short or long if needed. It can reference other mappings and because it is pure Python code, it can be constructed dynamically.

**For example,** path ('index/', views. Index, name='main-view') – maps [www.xyz.com/index](http://www.xyz.com/index) ( where [www.xyz.com](http://www.xyz.com) is the base URL of the web application) to the index function in the views.py file and that very function will be executed in order to generate a HTML response for the respective URL.

* **views.py -** A view function, or “view” for short, is simply a Python function that takes a web request and returns a web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, etc. Views.py essentially acts as a file that contains all the view functions.

By default, the functions in views.py are synchronous, however, views can also be asynchronous (“async”) functions, normally defined using Python’s async def syntax. Django will automatically detect these and run them in an async context. However, an async server based on ASGI will be required to get their performance benefits.

* **models.py -** A model is the single, definitive source of information about the data. It contains the essential fields and behaviours of the data being stored. Generally, each model maps to a single database table and each attribute of the model represents a database field. Models can also be used to define relationships between multiple schemas including many-one , many-many , etc.
* **settings.py –** Itis a core file in Django projects. It holds all the configuration values that the web application needs to work with; database settings, logging configuration, location of static files(containing images and other media) as well as any associated API keys if the web application requires the linkage of any external APIs.
* **admin.py –** This file is used to display models declared in models.py, in a collective manner. Django has a built-in admin panel that reads metadata from the declared models, such as fields and enables the admin of the application to perform CRUD (Create, Read, Update, Delete) operations manually. Only the admin of the application reserves the privilege of accessing the admin panel. The admin panel of an application can be accessed by entering the required admin id and password at [www.xyz.com/admin](http://www.xyz.com/admin) , where www.xyz.com is the base URL.
* **Template** – This folder is used to store the HTML templates used in the Django application. Storing of templates collectively allows the reusage of HTML templates instead of creating the same HTML template multiple times for similar tasks.
* **manage.py -** This file is used to interact with the project via the command line (start the server, sync the database, etc).

**2. The diagrams below analyse how an application created using Django works: -**

* 1. **Use Case diagram**

<<include>>

URL

Template

View

<<include>>

<<include>>

User requesting a URL



<<include>>

Model

**User via browser**

Database

HTML response by the Django server

Django server

**Explanation**

1. User enters a URL
2. URL is processed in the URL stage
3. The function mapped to the URL is executed from the **View stage**
4. The **View stage** can perform CRUD operations (Create, Read, Update, Delete) from the models in **Model stage**.
5. The **Model stage** has access to the Database and can modify or read attributes from the schemas declared in the **Database**.
6. The **View stage** then generates a response in the form of **Template Stage**
7. This response is viewed on the user’s browser
   1. **State Transition diagram**

BROWSER

Response sent as an HTML template with data binded by VIEWS

Browser requests a URL

TEMPLATES

URLS

If url is found in urls.py

Binds processed data, received from the model to the html template, to be displayed in the browser at the requested URL

If URL is not found in urls.py

VIEWS

No response received

Sends read value

Request for read/write data

404 ERROR

MODELS

Requested data

Updating/deletion/read of data from schemas in the database

DATABASE

**States present in the state transition diagram are: -**

* BROWSER
* TEMPLATES
* URLS
* VIEWS
* 404 ERROR
* MODELS
* DATABASE
  1. **Context Diagram**

Sends HTTP response to the Browser in the form of an HTML template

Sends data from the front-end

Template

(<filename>.html)

Write data

Read data

MODELS

(models.py)

Forward request to view

Sends HTTP request

BROWSER

URLS

(urls.py)

VIEW

Views.py

1. **Architecture Diagram**

HTTP request

Database

External Viewer (Browser)

**Template**

**Model**

urls.py

models.py

HTML template

**View**

views.py

**Explanation**

1. The user enters the URL of the web application in the browser. The data from the url (GETrequest) can be accessed in the views.py file
2. The URL is matched in the urls.py file and the function mapped to that URL is triggered in the views.py file.
3. The function corresponding to the matching URL is executed in the views.py file. This function sends or reads data to and from models.py file (contains the models).
4. The function can also perform delete and update operations in the models.
5. The model co-ordinates with the database to perform CRUD(Create, Read, Update, Delete) operation
6. After processing the data, the views.py file generates a response which can be in the form of a HTML template, an error, etc.
7. Additionally, some data can also be binded to the HTML response sent by the function in views.py.
8. The corresponding HTML template with binded data (if any) is sent as a response to the browser.

**4.0 Three quality attributes addressed by the design**

1. **Testability –** Django promotes the methodology of Test-Driven Development which enables developers to write new code after their old code passes the automated test cases. This way, the developers are ensured that the code they write is clear and bug-free. This Test-Driven Development methodology can be used by the developers while writing every small functionality of an application. The three laws of test-driven development that every developer should ideally follow are: -

* Writing a failing unit test for a feature.
* Make just enough changes in the code needed to pass the failing unit test.
* Refactor the code if possible.

Django creates a tests.py file, by default, for the developers thereby encouraging them to follow the Test-Driven Development practice and ensuring bug-free and optimised code.

1. **Security** – Django is an open-source framework that is widely known majorly for the various out-of-box security features it provides, some of these features are: -

* **SQL injection** - SQL injection is a type of attack where a malicious user can execute arbitrary SQL code on a database. This can result in records being deleted or data leakage.

Django’s query sets are protected from SQL injection since their queries are constructed using query parameterization. A query’s SQL code is defined separately from the query’s parameters. Since parameters may be user-provided and therefore unsafe, they are escaped by the underlying database driver.

* **Cross site request forgery (CSRF) protection -** CSRF attacks allow a malicious user to execute actions using the credentials of another user without that user’s knowledge or consent.

Django has built-in protection against most types of CSRF attacks. [CSRF protection works](https://docs.djangoproject.com/en/3.1/ref/csrf/#how-csrf-works) by checking for a secret in each POST request. This ensures that a malicious user cannot “replay” a form POST to the website and have another logged in user unwittingly submit that form. The malicious user would have to know the secret, which is user specific (using a cookie).

* **Cross site scripting (XSS) protection** - XSS attacks allow a user to inject client-side scripts into the browsers of other users. This is usually achieved by storing the malicious scripts in the database where it will be retrieved and displayed to other users, or by getting users to

click a link which will cause the attacker’s JavaScript to be executed by the user’s browser. However, XSS attacks can originate from any untrusted source of data, such as cookies or Web services, whenever the data is not sufficiently sanitized before including in a page. Django templates protect the web-application against a majority of XSS attacks by escaping specific characters which are particularly dangerous to HTML. This feature protects users from malicious inputs from cross site scripting.

1. **Portability** – The biggest strength of Django lies in the fact that it is compatible with almost all the databases that exist. This is achieved with the Django models that enable provide a mapping to the underlying database at the backend. Hence the switch between multiple databases can be made with a single line of code without having to worry about the syntax used for querying in various databases.

**5.0 Design Style**

A web framework is a software tool that provides a way to build and run web applications. A web-framework usually has a basic model view controller architecture software design pattern for developing web applications, but **Django** is a bit different in a good way. It implements the concept of **Model-View-Template (MVT) which is slightly different from the traditional Model-View-Controller used by a majority of frameworks**. The main difference between the two patterns is that **Django**itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving only the implementation of the template (HTML file mixed with **Django Template Language (DTL)**) which makes Django easy to use for developers.

**Model -** Model is a logical data structure that acts as the middleware and data handler between database and view. The Model provides a definition of how the data formats as coming from the view so, it stores in the database and vice-versa, i.e., the retrieving information from the database transfers to the view in the displayable format.

**View** **-** View is the part that receives data as well as requests from client side and accordingly formats the data via model so that it can be stored in database. Additionally, it communicates with the model for retrieving data which can be transferred to the template for viewing. At the end of execution, the view returns a response to the client in the form of a HTML template, error, image, etc.

**Template** **-** Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted. Template provides much more extendibility to the frontend developers than the traditional MVC architecture provides. One template can be used by different views to show various formats of data, additionally, it keeps all the content that is rendered by the browser. The template part is visible to the client side, while on the other hand, model and views reside on the server side.

1. **Discussions**

Some of the key aspects that separate Django from the rest of the web-development frameworks are: -

1. Django’s models are written in python and provide a mapping to the underlying database structure. Moreover, Django uses a model to execute SQL queries behind the scenes to return python data structures known as **Query Sets**. Writing models as mappings for underlying database has several advantages: -

* **Consistency** – Since SQL is inconsistent across different databases, hence not having to write SQL queries for each database to which the application is deployed, will lead to consistency as well as save time and hassle.
* **Version Control** – Storing models in the codebase makes it easier to keep track of design changes rather than creating or modifying schemas in an actual database.
* **Advanced Metadata** – Having models described in python code instead of SQL allows for special datatypes (example, email-address) and provides the ability to store much more metadata than plain SQL bases database.

1. One of the biggest advantages of using Django is that it can handle traffic API usage of more than 400 million plus users helping maximize scalability and minimize web hosting costs price. An application developed using Django can easily serve a high number of hosts with a relatively cheap hosting price which can also sometimes be free (by using Heroku for deployment).
2. Django is highly scalable due to the fact that it has been developed in such a way that it can handle any kind of hardware additions, this is the main reason as to why the busiest sites of the world like Instagram, Pinterest, Disqus, etc rely on Django.