**Django**

**What is Django?**

Django is a Python framework that makes it easier to create web sites using Python. Django takes care of the difficult stuff so that you can concentrate on building your web applications. Django emphasizes reusability of components, also referred to as DRY (Don't Repeat Yourself), and comes with ready-to-use features like login system, database connection and CRUD operations (Create Read Update Delete).

**Advantages of Django**

1. **Rapid Development**: Django's "batteries-included" approach provides a lot of built-in features, which speeds up development2.
2. **Security**: Django comes with built-in protection against common security threats like SQL injection,

cross-site scripting, and cross-site request forgery2.

1. **Scalability**: Django can handle high-traffic websites and applications12.
2. **Versatility**: It can be used for a wide range of applications, from simple websites to complex, database-driven applications12.
3. **Community Support**: Django has a large and active community, providing extensive documentation, tutorials, and third-party packages12

**Disadvantages of Django**

1. **Monolithic:** The Django framework offers a specific technique for defining and performing jobs. It has a logical file structure that is simple to understand. However, this also means that you are not permitted to use your file structure. It’s because the framework has a style of doing things commonly known as “The Django way.” If you do not adhere to these guidelines, you may be unable to release anything using Django.
2. **Not Suitable for smaller projects:** When it comes to web development frameworks, Django is a

popular choice for larger projects. However, it may not be the best option for smaller businesses. If you’re looking for a more lightweight option, consider using a simpler web development framework like Flask. Flask is more suited to smaller projects and offers fewer features than Django, but it’s still powerful enough to get the job done.

**Applications of Django**

1. **Content Management Systems (CMS)**: Django is widely used to build CMS platforms like Django CMS and Wagtail12.
2. **Social Media Platforms**: Platforms like Pinterest and Instagram have used Django for their backend1.
3. **E-commerce Sites**: Django can be used to create e-commerce websites with features like product catalogs, shopping carts, and payment processing12.
4. **Scientific Computing**: Django is used in scientific projects for data analysis and visualization12.
5. **APIs**: Django REST framework is a powerful toolkit for building Web APIs12.

**How does Django work?**

Django follows the MVT design pattern (Model View Template).

* Model - The data you want to present, usually data from a database.
* View - A request handler that returns the relevant template and content - based on the request from the user.
* Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.

**Model**

The model provides data from the database.

In Django, the data is delivered as an Object Relational Mapping (ORM), which is a technique designed to make it easier to work with databases.

The most common way to extract data from a database is SQL. One problem with SQL is that you have to have a pretty good understanding of the database structure to be able to work with it.

Django, with ORM, makes it easier to communicate with the database, without having to write complex SQL statements.

The models are usually located in a file called models.py

**View**

A view is a function or method that takes http requests as arguments, imports the relevant model(s), and finds out what data to send to the template, and returns the final result.

The views are usually located in a file called views.py.

**Template**

A template is a file where you describe how the result should be represented.

Templates are often .html files, with HTML code describing the layout of a web page, but it can also be in other file formats to present other results, but we will concentrate on .html files.

Django uses standard HTML to describe the layout, but uses Django tags to add logic:

The templates of an application is located in a folder named templates.

**URLs**

Django also provides a way to navigate around the different pages in a website.

When a user requests a URL, Django decides which view it will send it to.

This is done in a file called urls.py.

**API**

An API (Application Programming Interface) in Django allows different applications to communicate with each other. Django REST framework (DRF) is a powerful and flexible toolkit for building Web APIs in Django. It makes it easy to develop APIs for your Django applications, enabling you to expose data and functionality to other services, such as mobile apps or other web services

**Rest API**

The REST API for Documents enables you to interact with folders and files stored in Oracle Content Management Cloud and to create sites from templates and other sites.

Django REST framework is a powerful and flexible toolkit for building Web APIs.

**Some reasons you might want to use REST framework:**

The Web browsable API is a huge usability win for your developers.

Authentication policies including packages.

Serialization that supports both ORM and non-ORM data sources.

Customizable all the way down - just use regular function-based views if you don't need the more powerful features.

Extensive documentation, and great community support.

Used and trusted by internationally recognized companies including Mozilla, Red Hat, Heroku, and Eventbrite.

**Funding:**

REST framework is a collaboratively funded project. If you use REST framework commercially, we strongly encourage you to invest in its continued development by signing up for a paid plan.