

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Django Rest Framework CRUD application | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Samarth Kumar |  | | | |
|  | | | |  |  | | | |
|  | | |  | | |  | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  | | |  |
|  | INTRODUCTION | | | | | | |  |
|  |  | | |  |  | | |  |
|  |  | |  | | |  | |  |
|  |  |  | The purpose of this report is to document the development process of a CRUD (Create, Read, Update, Delete) application using Django Rest Framework (DRF) and Ajax. The application allows users to perform basic operations on a database, including creating new records, retrieving existing records, updating records, and deleting records. | | |  |  |  |
|  | | | | |
|  |  |  |  |
|  |  |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | |  |  | |
|  | | Application Architecture | | | | |  | |
|  | |  |  | | |  |  | |
|  | Details of project The application follows a client-server architecture, where the Django backend serves as the server, handling the requests and responses, and the frontend interacts with the user. The communication between the frontend and backend is achieved using Ajax, enabling asynchronous data exchange without the need to reload the entire page. | | | | | | |  |
|  | Backend Development | | |  |  | | |  |
|  | The backend of the application is built using Django and Django Rest Framework. The database used in this project is PostgreSQL , all the data is stored, retrieved , deleted etc. from this database only | | |  | Create a virtual environment and activate it.  Install Django and Django Rest Framework using pip.  Create a new Django project and set up the necessary configurations.  Create an app within the project to manage the CRUD functionality. | | |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Models and Database  The models are stored in models.py file in Products app in the project folder.  The models have tables namely:  Name, Category, Quantity, Price. The field name and category are a Char Field meaning they can store characters whereas the field Quantity is an Integer Field and field price is a Decimal Field.  All these data are stored in PostgreSQL database. | |  | | | | |  |
|  | |  | |  |  |  |  | |
|  |  | |  | | | | |  |

URL Routes :

path('', views.Index, name="Index"),

    path('api/products/', views.ProductsAPI, name="ProductsAPI"),

    path('api/products/add/', views.AddProductAPI, name="AddProductAPI"),

    path('api/products/<int:id>/', views.ProductsDetailsAPI, name="ProductsDetailsAPI"),

    path('api/products/edit/<int:id>/', views.EditProductAPI, name="EditProductAPI"),

    path('api/products/delete/<int:id>/', views.DeleteProductAPI, name="DeleteProductAPI"),

JavaScript and AJAX:

I have written JavaScript Ajax code to fetch the API details for each api view for CRUD functionality

TESTING:

The application is manually tested with the help of postman and it is found that every functionality is working properly without any exceptions and error.