**Task-01**

**1. Task Description**

**Objective:**

The objective of this task is to build a Django web application that accepts a CSV file upload, parses the CSV content, and stores the data in a model (database table) that reflects the structure of the CSV file. The CSV file contains 4 columns, which will be dynamically created based on the headers in the uploaded CSV file. Additionally, the model includes a Date field to store the timestamp of each record.

**Steps Involved:**

1. Set Up Django Project and App:

- Created a new Django project called **dataframe\_project**.

- Created a Django app called **data\_app**.

2. Model Definition:

- Defined the **DataFrameModel** model with four fields dynamically matching the CSV headers, and an additional **Date** field.

3. CSV Upload Functionality:

- Developed a form that allows users to upload a CSV file.

- Used Pandas to parse the CSV file and extract the data.

4. Admin Panel:

- Registered the **DataFrameModel** in the Django admin for easy management of the database records.

5. Database Integration:

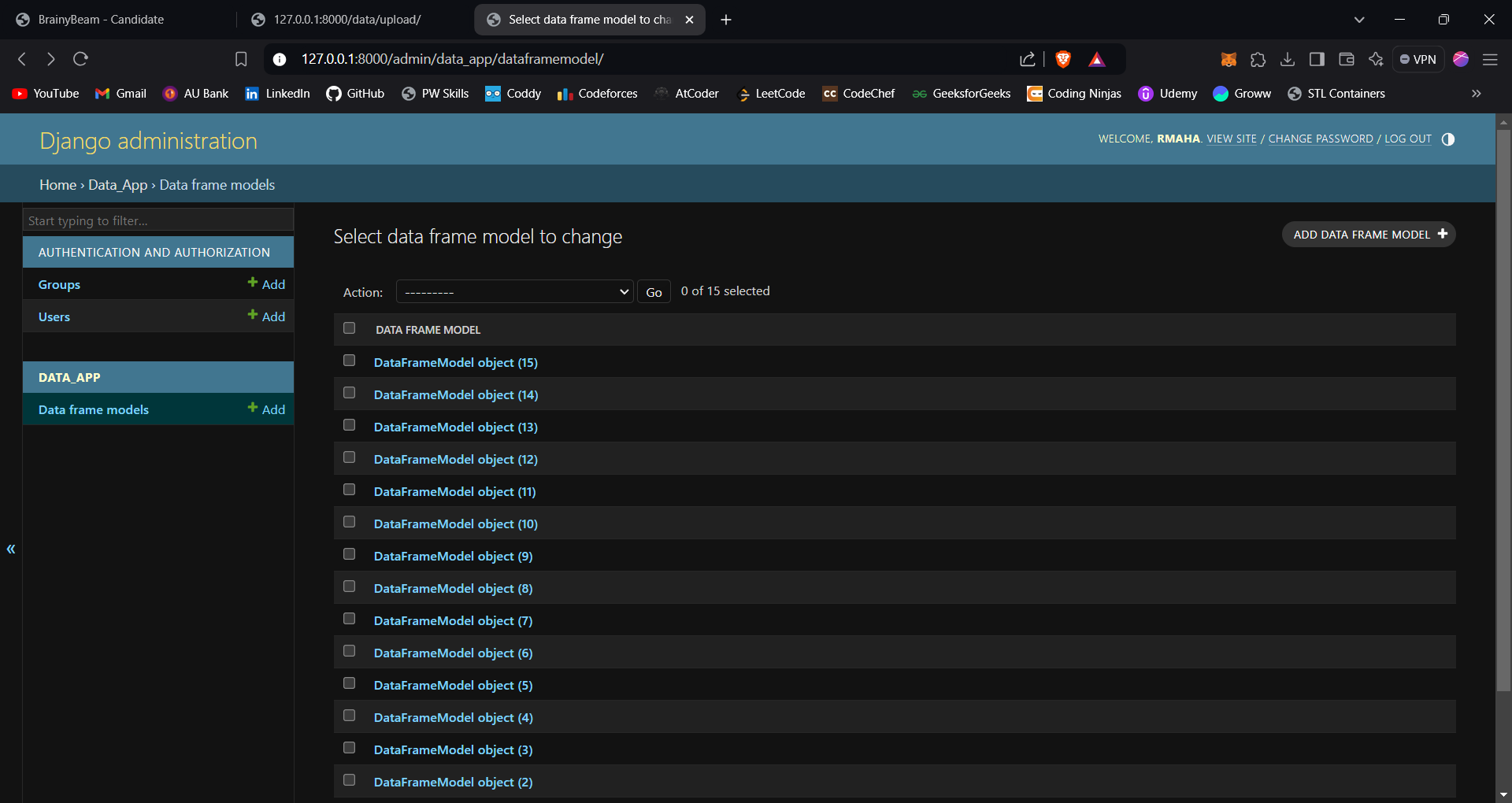
- Mapped the parsed CSV data to the Django model and saved it to the database.

6. Error Handling:

- Managed errors such as file format validation and database constraints during the file upload and processing.

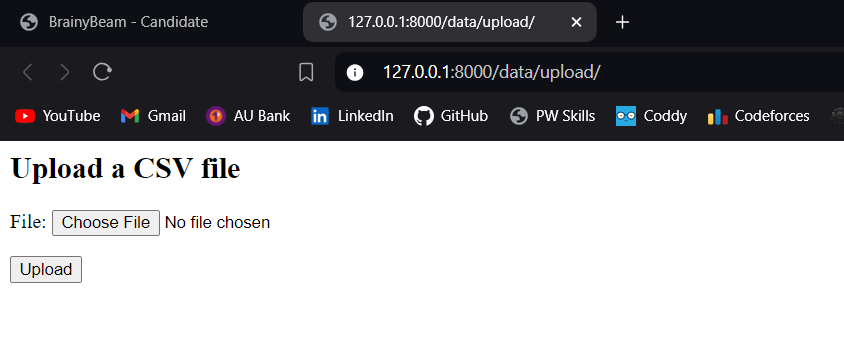
**2. Screenshot of Output**

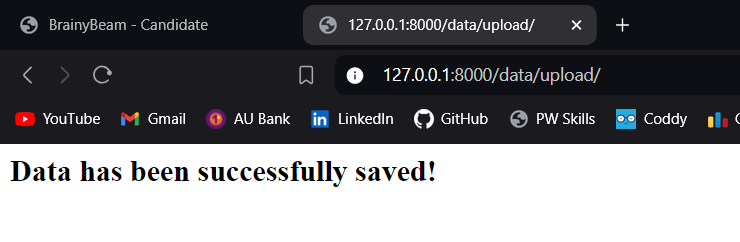
Below is the screenshot of the Django admin panel, indicating successful login and creation of the **DataFrameModel**:



**CSV Data Successfully Uploaded to the Model:**

Below is the screenshot showing how the CSV data has been uploaded to the model:





**3. Description of Widgets/Algorithm Used in the Task**

1. Widgets:

- CSV Upload Form: Created an HTML form using Django's form handling to allow users to upload CSV files.

- Admin Interface: Registered **DataFrameModel** in the Django admin panel for easier viewing and management of uploaded data.

2. Algorithm:

- Pandas CSV Parsing:

- Utilized the **pandas.read\_csv()** function to parse the uploaded CSV file. This function reads the CSV file into a **DataFrame** object, which can then be iterated over to extract data row by row.

- Data Mapping and Saving:

- The rows of the **DataFrame** are iterated over, and each row is saved into the **DataFrameModel**. We used **DataFrameModel.objects.create()** to insert each record into the database.

- Error Handling:

- Implemented try-except blocks to handle file parsing errors and incorrect file format issues.

- Admin Interface:

- Registered the model **DataFrameModel** in **admin.py** using Django's built-in **admin.site.register()** method, which allows admin users to view and manage records in the database.