# Architecture Design

## CPA (Customer Personality Analysis)

| | |
|---|---|
| **Written By** | Shashank Shukla |
| **Document Version** | 0.1 |
| **Last Revised Date** | |

## DOCUMENT CONTROL

### Change Record:

| VERSION | DATE | AUTHOR | COMMENTS |
|---|---|---|---|
| 0.1 | 19- May - 2021 | Shashank Shukla | Introduction and architecture defined |
| 0.2 | 20 - May - 2021 | Author 2 | Architecture & Architecture description appended and updated. |
| | | | |
| | | | |

### Reviews:

| VERSION | DATE | REVIEWER | COMMENTS |
|---|---|---|---|
| 0.2 | 21- May - 2021 | | |

### Approval Status:

| VERSION | REVIEW DATE | REVIEWED BY | | APPROVED BY | COMMENTS |
|---|---|---|---|---|---|
| | | | | | |

## Contents

# 1. Introduction

### 1.1 What is Architecture design document?

Any software needs the architectural design to represents the design of software. IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectures.

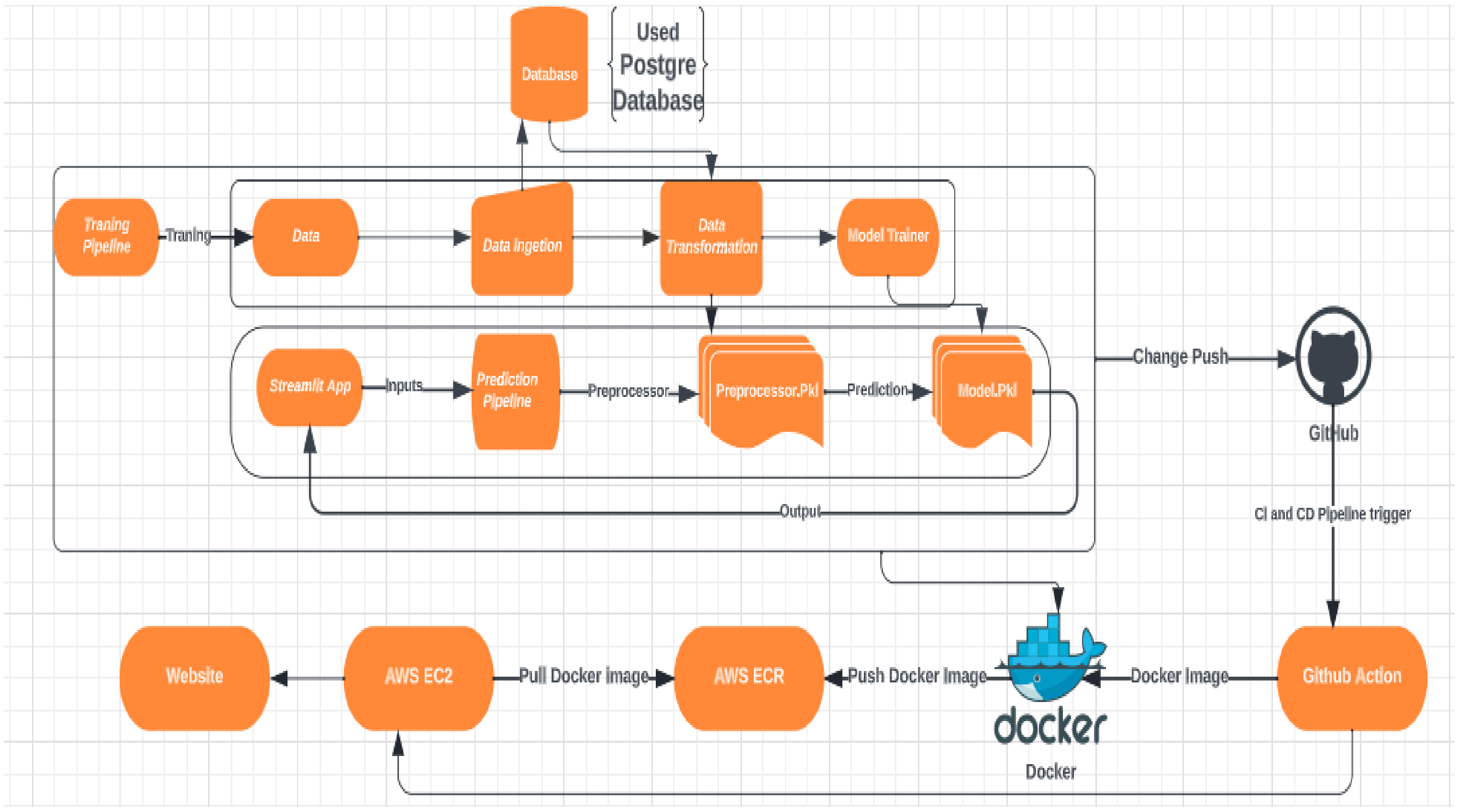
Each style will describe a system category that consists of :

- A set of components (eg: a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

### 1.2 Scope

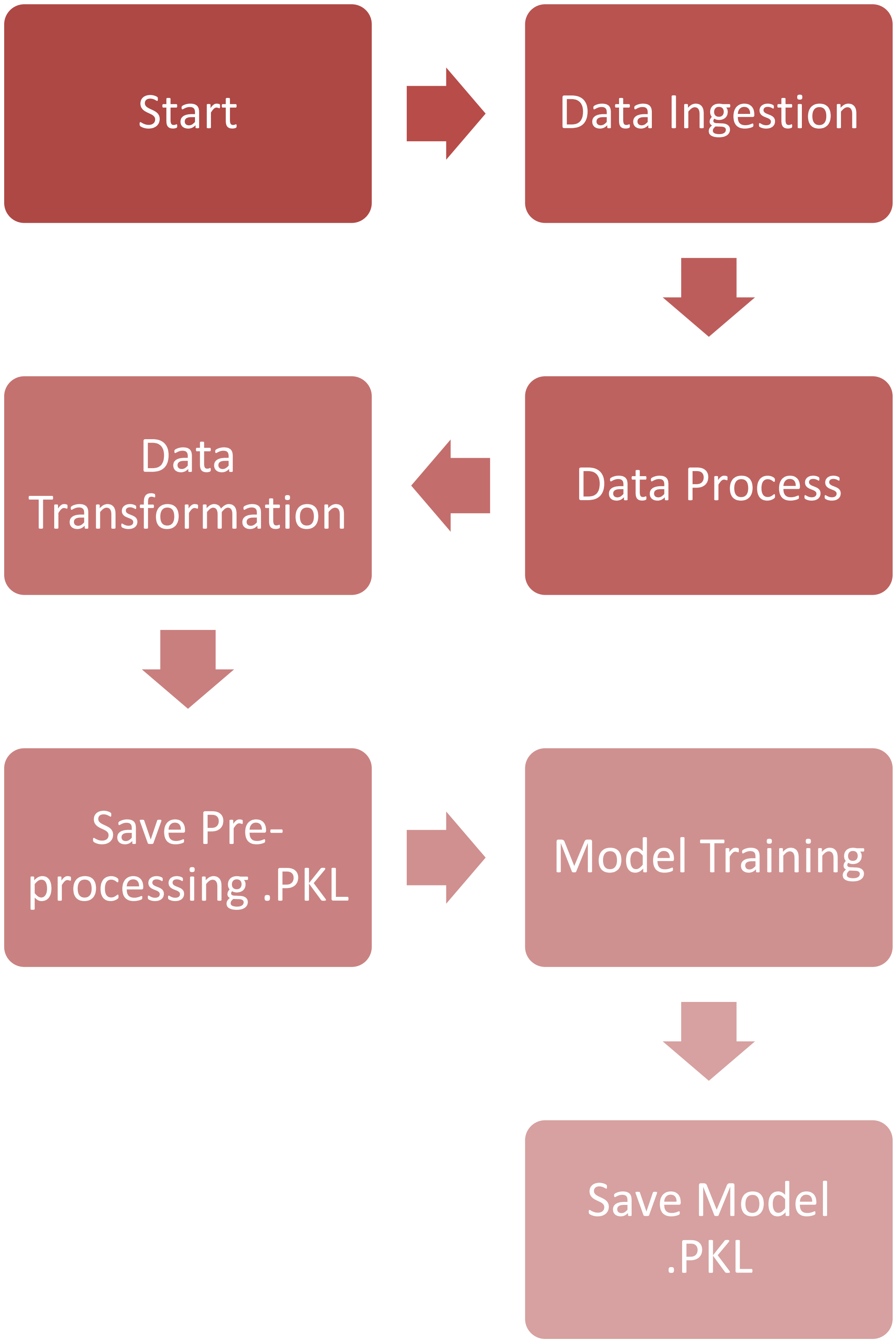
Architecture Design Document (ADD) is an architecture design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the design principles may be defined during requirement analysis and then refined during architectural design work.

## 2. Architecture



### 2.1 Training Pipeline Architecture

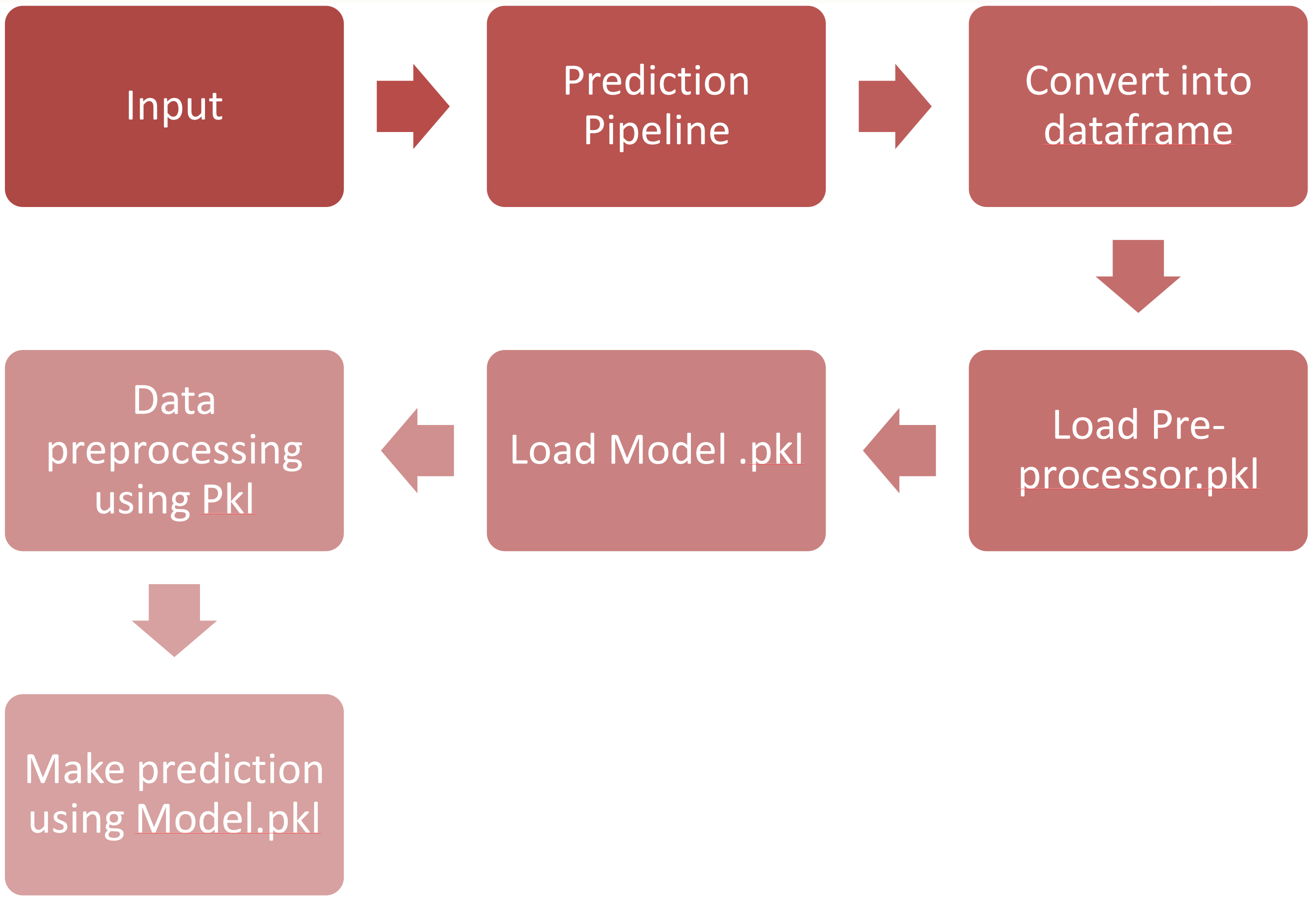
In the context of our data processing pipeline, we have developed an integrated training pipeline that encompasses data ingestion, data processing, data transformation, and model training. This section outlines the key steps and components of this training pipeline.


Start
Data Ingestion
Data Process
Data Transformation
Save Pre-processing .PKL
Model Training
Save Model .PKL

The following diagram shows Training Pipeline

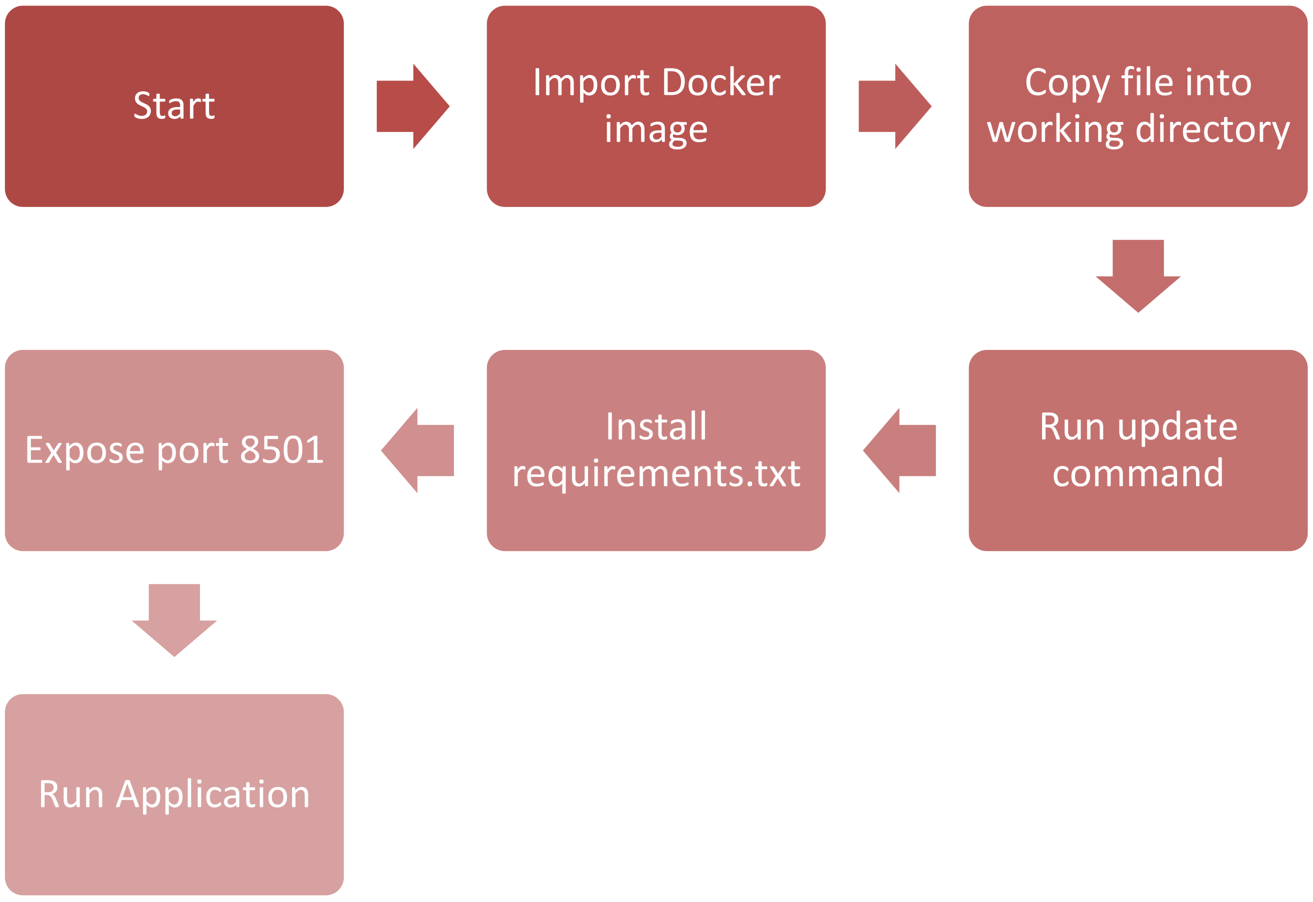
### 2.2. Prediction Pipeline

This Pipeline is responsible for making predictions using a machine-learning model. It loads both the preprocessor (.pkl) file and the trained model. The input data is passed through the preprocessor to prepare it for prediction, and then the model is applied to generate predictions.

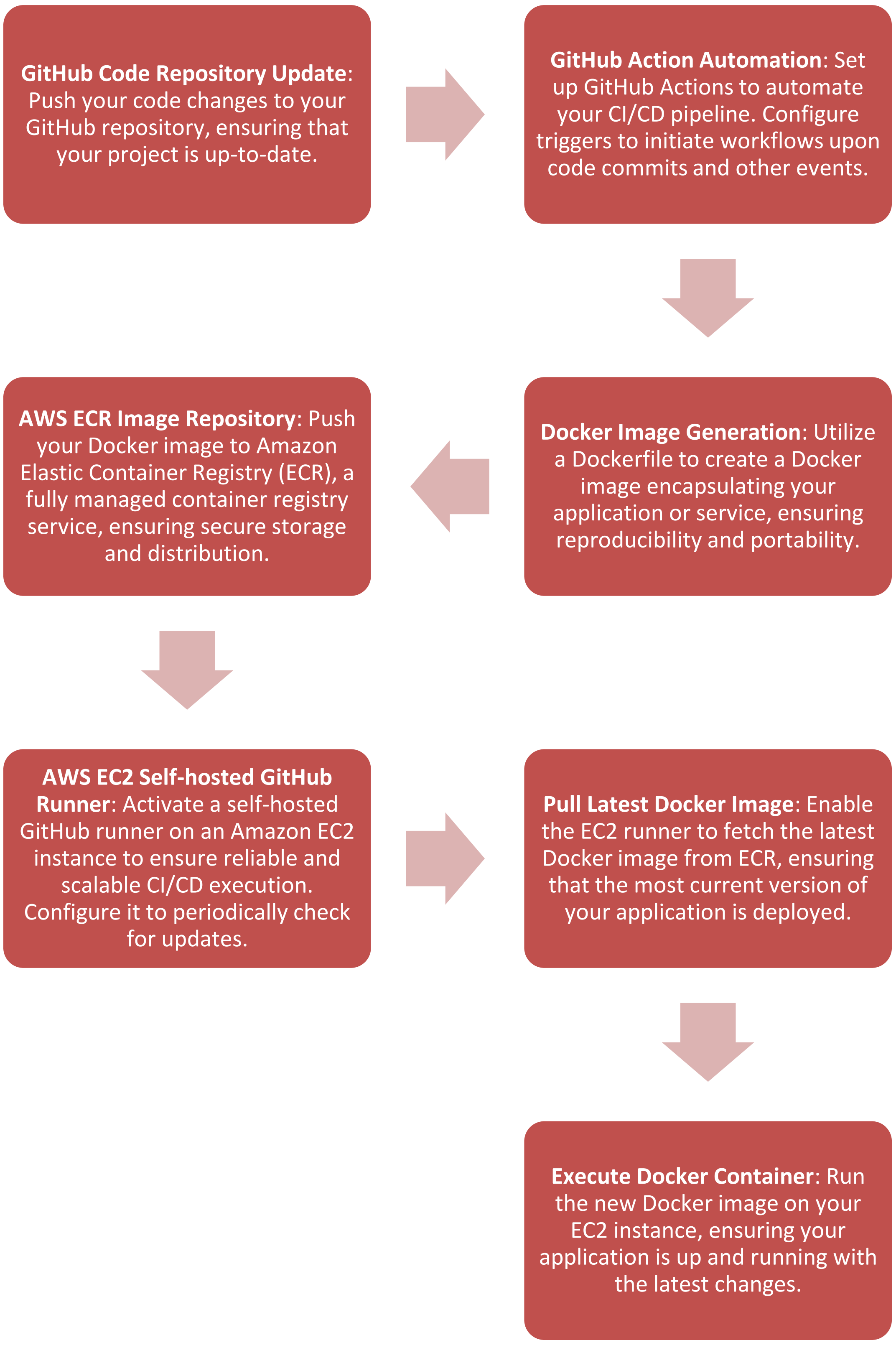


## 2.3 Docker Build Integration

Within our project, a Dockerfile is meticulously crafted. This file outlines the precise steps necessary to construct the Docker image. It specifies the base image, necessary dependencies, and the setup of our application.



## 2.4 CI/CD Pipeline Architecture


**GitHub Code Repository Update**: Push your code changes to your GitHub repository, ensuring that your project is up-to-date.
**GitHub Action Automation**: Set up GitHub Actions to automate your CI/CD pipeline. Configure triggers to initiate workflows upon code commits and other events.
**Docker Image Generation**: Utilize a Dockerfile to create a Docker image encapsulating your application or service, ensuring reproducibility and portability.
**AWS ECR Image Repository**: Push your Docker image to Amazon Elastic Container Registry (ECR), a fully managed container registry service, ensuring secure storage and distribution.
**AWS EC2 Self-hosted GitHub Runner**: Activate a self-hosted GitHub runner on an Amazon EC2 instance to ensure reliable and scalable CI/CD execution. Configure it to periodically check for updates.
**Pull Latest Docker Image**: Enable the EC2 runner to fetch the latest Docker image from ECR, ensuring that the most current version of your application is deployed.
**Execute Docker Container**: Run the new Docker image on your EC2 instance, ensuring your application is up and running with the latest changes.

## 2.5 Streamlit application

• The user visits the Home Page (Form Page) and fills in the required data.

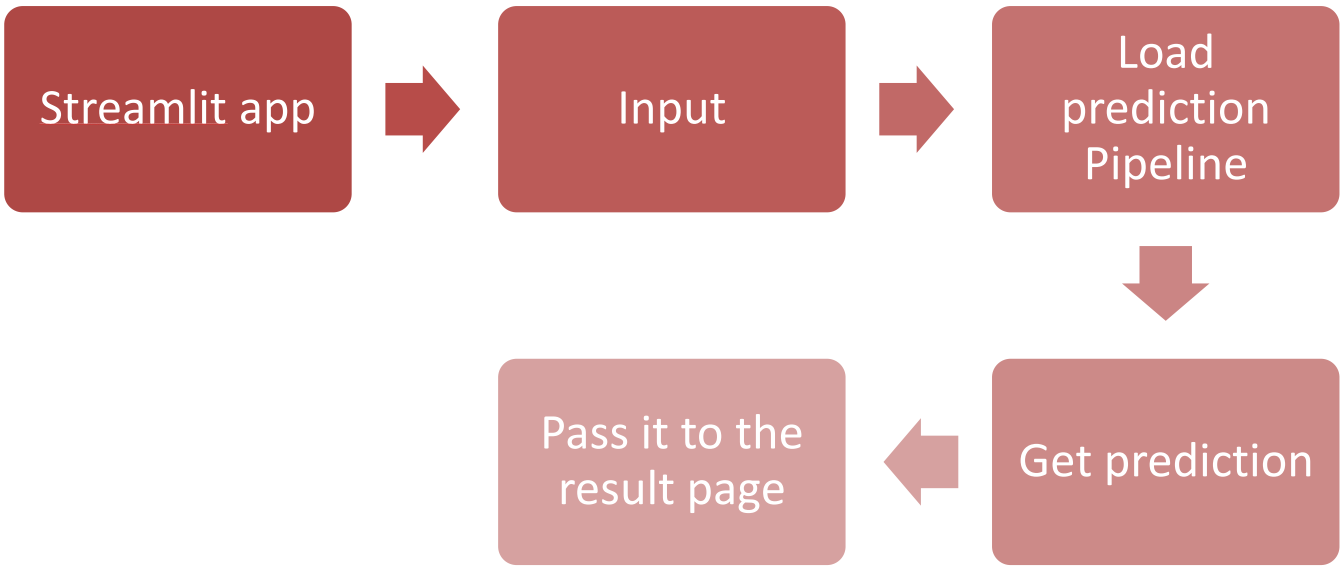
• Upon hitting the submit button, the app triggers the prediction process.

• The user is redirected to the Result Page, where the predicted results are displayed in a chat format.

• The user can navigate to the Charts Page to see cluster analysis and insights.

• The Make Your Own Chat Page allows users to interact with existing chat data and create their own conversations.

By organizing your Streamlit app in this manner, you provide a user-friendly interface for data input, prediction display, cluster analysis, and interactive chat functionality. This structure should help users easily navigate and interact with your app. to



## 2.6 AWS (EC2)

As our chosen deployment target, we've established an AWS EC2 instance to serve as a robust and scalable hosting environment for our application.

A fundamental component of our CD workflow is the deployment step. This step efficiently deploys our Docker container onto the AWS EC2 instance. To achieve this, we've employed a self-hosted runner configured with a YAML file. This runner pulls the Docker image, performs necessary configurations specified in the Dockerfile, and deploys the application on the EC2 instance. This process may also involve pulling the Docker image from the Amazon ECR repository, ensuring a seamless deployment.

## 2.7. Database

**Data Ingestion:**

❖ The process begins with data ingestion, handled by the DataIngestion class.

❖ The data ingestion returns the table name in which data is stored that name stored into the variable table_name.

**Data Processing and Dimensionality Reduction:**

❖ Following data ingestion, the DataCleaning class is utilized for data cleaning and dimensionality reduction.

❖ This step produces a cleaned table named cleaned_table_name and an unencoded DataFrame called Without_encoding.

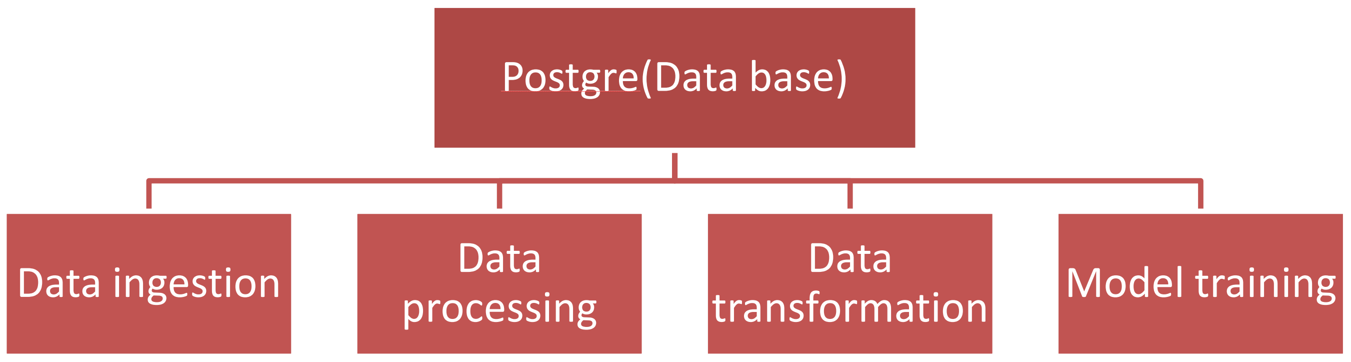
**Data Transformation:**

❖ Data transformation is the next step, performed using the DataTransformation class.

❖ The result is a transformed training DataFrame, train_df. 5.6.2.4. Model Training:

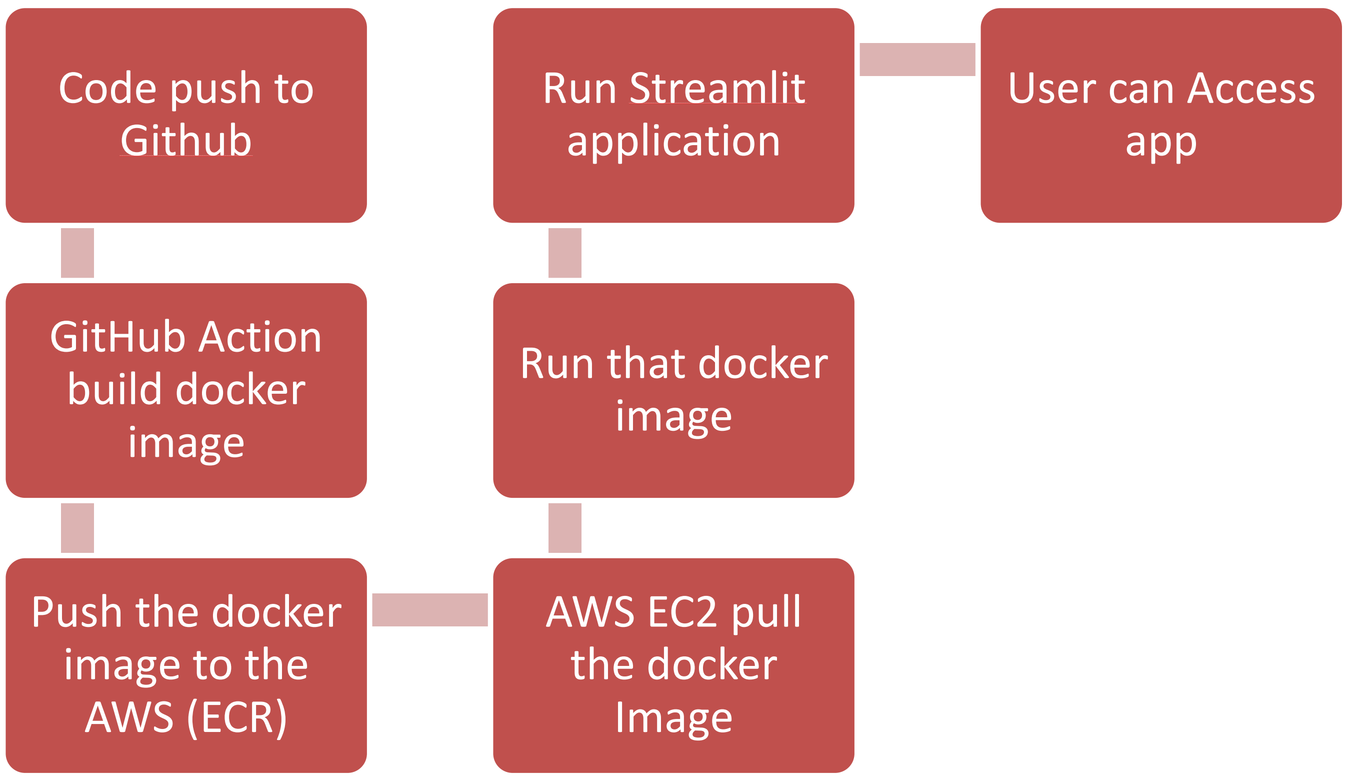
❖ Finally, the Model Trainer class is employed to initiate the model training process.

❖ The processed data and unencoded Data Frame are used to train the machine learning model.



## 3. Deployment Description

In the deployment phase, we've hosted the entire website on an Amazon Elastic Compute Cloud (EC2) instance. This EC2 instance serves as the hosting environment for our website, making it accessible to users over the internet. This setup allows us to deploy and run our web application in a scalable and reliable manner on the AWS infrastructure.


Code push to Github
GitHub Action build docker image
Push the docker image to the AWS (ECR)
AWS EC2 pull the docker Image
Run that docker image
Run Streamlit application
User can Access app