**MindSync**

**A Project Report**

***Submitted by***

Manoj Kumar Pradhan (22BCE11415)

Mohammad Ammar (22BCE10394)

SK Sahil Islam (22BCE10440)

Shishir Agarwal (22BCE10771)

Rushikesh Raut (22BCE10965)

*in partial fulfilment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE AND ENGINEERING (CORE)**



**SCHOOL OF COMPUTING SCIENCE AND |**

**ENGINEERING VIT BHOPAL UNIVERSITY**

**Kothri Kalan, Sehore,**

**Madhya Pradesh, 466114**

**Chapter 1**

**Project Description, Outline & Novelty**

* 1. **Introduction**

Attention-deficit/hyperactivity disorder (ADHD) is marked by an ongoing pattern of inattention and/or hyperactivity-impulsivity that interferes with functioning or development. People with ADHD experience an ongoing pattern of the following types of symptoms:

* **Inattention** means a person may have difficulty staying on task, sustaining focus, and staying organized, and these problems are not due to defiance or lack of comprehension.
* **Hyperactivity** means a person may seem to move about constantly, including in situations when it is not appropriate, or excessively fidgets, taps, or talks. In adults, hyperactivity may mean extreme restlessness or talking too much.
* **Impulsivity** means a person may act without thinking or have difficulty with self-control. Impulsivity could also include a desire for immediate rewards or the inability to delay gratification. An impulsive person may interrupt others or make important decisions without considering long-term consequences.

(Resource: [National Institute of Mental Health](https://www.nimh.nih.gov/health/topics/attention-deficit-hyperactivity-disorder-adhd))

Our team endeavours to bridge this gap by leveraging advanced machine learning techniques and neuroimaging data to develop an objective and automated system for ADHD diagnosis. By integrating structural brain MRI data and personal characteristics, we aim to enhance accuracy and efficiency in ADHD classification.

* 1. **Motivation for the work**

The motivation for our research stems from the need to overcome the limitations of traditional ADHD diagnosis methods, which often rely on subjective assessments. By harnessing the power of machine learning and neuroimaging data, we aim to provide a more objective and efficient approach to ADHD classification, ultimately facilitating early intervention and tailored treatment strategies.

* 1. **Problem Statement**

We're tackling the challenge of spotting ADHD in its early stages. Right now, diagnosing ADHD relies a lot on guesswork and can take a long time. With machine learning, we want to change that. Our goal is to build a system that can quickly and accurately identify signs of ADHD before they become too big of a problem. By using machine learning and looking at brain scans and personal information, we aim to create a tool that doctors can use to catch ADHD early on.

* 1. **Project Objectives**

Develop a machine learning framework capable of automated ADHD diagnosis utilizing structural brain MRI data and personal characteristic features. Identify salient features from brain MRI volumes and personal characteristic data to improve classification accuracy. Evaluate the performance of various classifiers in ADHD classification through rigorous ten-fold cross-validation. Investigate neuroanatomical correlates of ADHD by analysing volumetric and cortical thickness alterations in specific brain regions.

* 1. **Project Outcome**
* Our project aims to deliver a machine learning tool that can accurately detect ADHD early on.
* By analysing brain scans and personal data, our model can identify patterns associated with ADHD and provide healthcare professionals with valuable insights.
* This will enable earlier interventions and better outcomes for children with ADHD.
  1. **Novelty of this Project**

Our project stands out for its integration of structural brain MRI data and personal characteristic features within a machine learning framework for ADHD diagnosis.

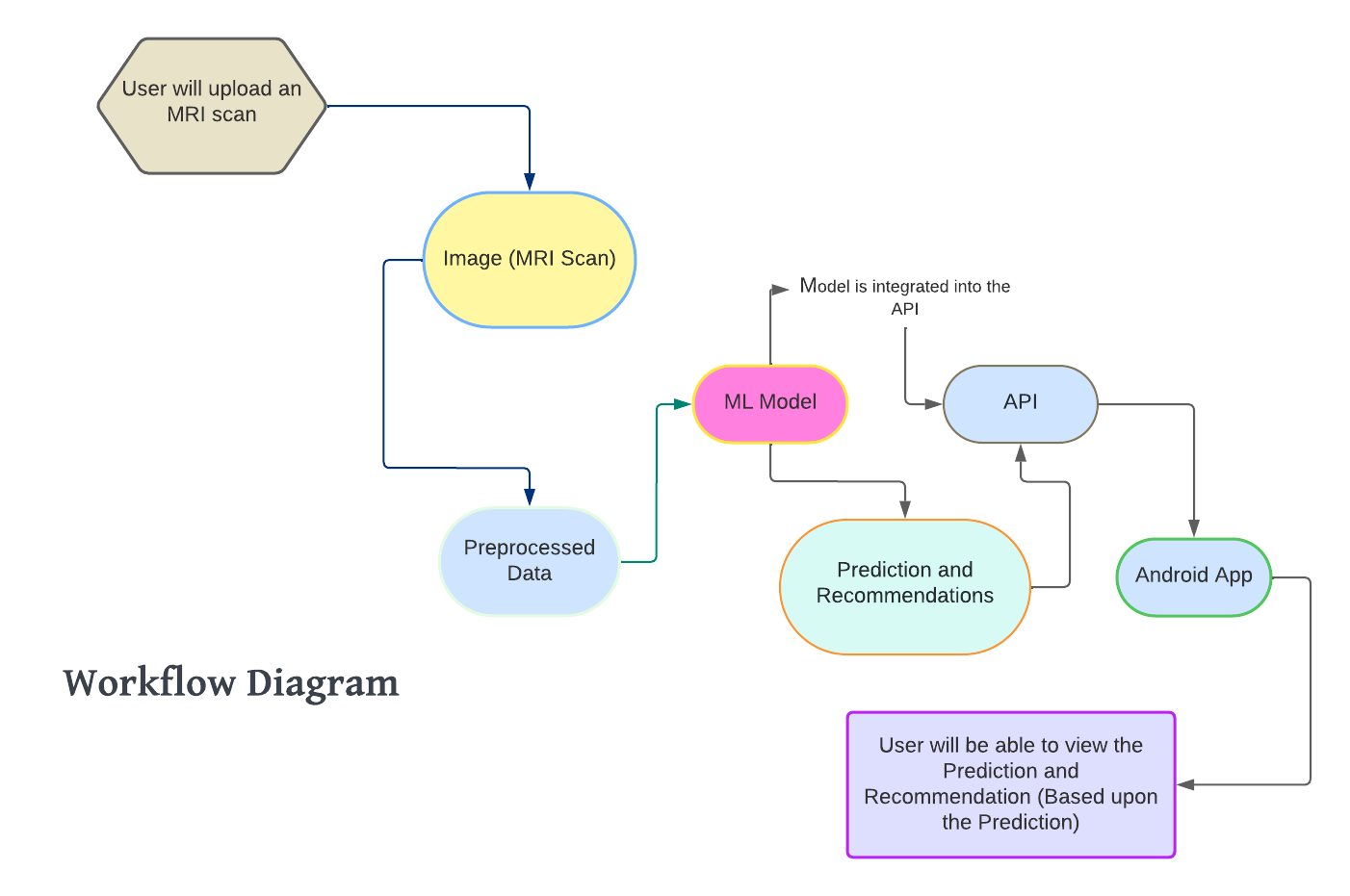
By employing ensemble feature selection techniques and comprehensive evaluation methods, we enhance the robustness and applicability of our proposed system.

Furthermore, our identification of neuroanatomical alterations associated with ADHD contributes to advancing our understanding of the disorder's underlying mechanisms.

**Chapter 2**

**Workflow, Architecture & Timeline**

**2.1 Basic Overview of working of our system**

****

**Explanation:**

**1.** The user uploads an MRI scan into the system.

**2.** The uploaded image (MRI Scan) undergoes preprocessing to prepare the data for analysis. This typically includes steps like noise reduction, normalization, and possibly feature extraction.

**3.** The pre-processed data is then fed into a machine learning (ML) model. The model processes the data to make a prediction, which could relate to diagnosing a condition, identifying anomalies, or any other task for which the model is trained.

**4.** The ML model is integrated into an application programming interface (API). This integration allows the model to be accessed by other software components or applications.

**5.** The API generates predictions and recommendations based on the input data processed by the ML model.

**6.** The predictions and recommendations are then made available to an Android application. This means the API sends the results to the Android app, where it can be viewed by the user.

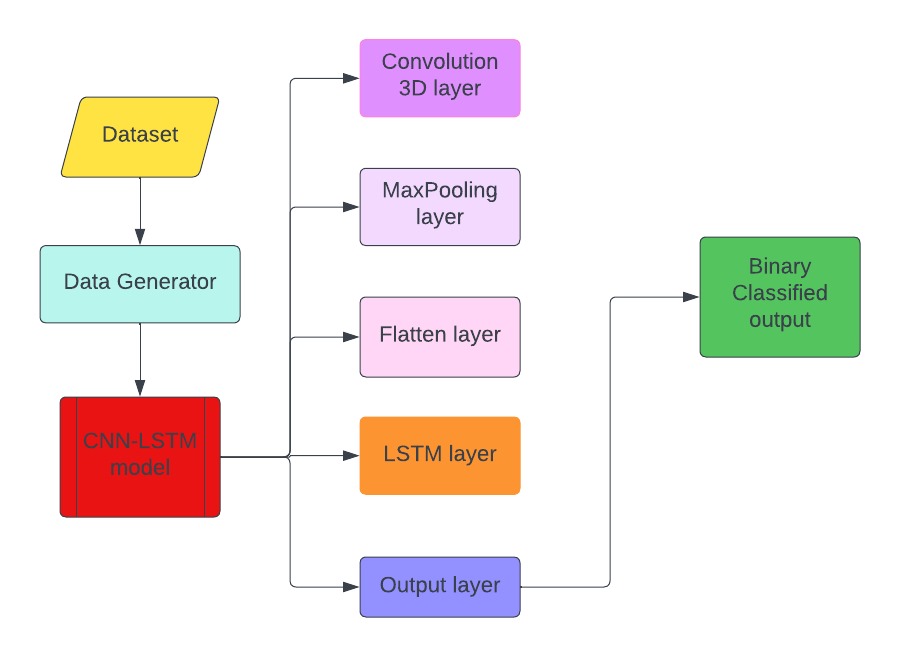
**7.** Finally, the user will be able to view the predictions and recommendations generated by the model on their Android device. These results are based upon the initial MRI scan data provided by the user.

**2.2 Detailed Overview of our Project**

Our project have mainly 3 components which are interconnected with each other.

1. **ML Model**
2. **API**
3. **Android App**

**2.2.1 ML Model Workflow**

****

**Explanation:**

**1. Dataset:** The process begins with a dataset that contains the data to be used for training the model.

**2. Data Generator:** A data generator takes the dataset and prepares the data for processing by the model. This may include data augmentation, normalization, and batching, which are essential for efficient training.

**3. CNN-LSTM model:** The prepared data is fed into a composite model that combines Convolutional Neural Network (CNN) layers with Long Short-Term Memory (LSTM) layers. **4. Convolution 3D layer:** Within the CNN-LSTM model, the first operation is a three-dimensional convolutional layer. This layer applies filters to the input data to extract features and is designed to process 3D data such as sequences of images or videos.

**5. MaxPooling layer:** Following the convolutional layer, a max pooling layer reduces the dimensionality of the data by down-sampling, which helps in reducing overfitting and computational load.

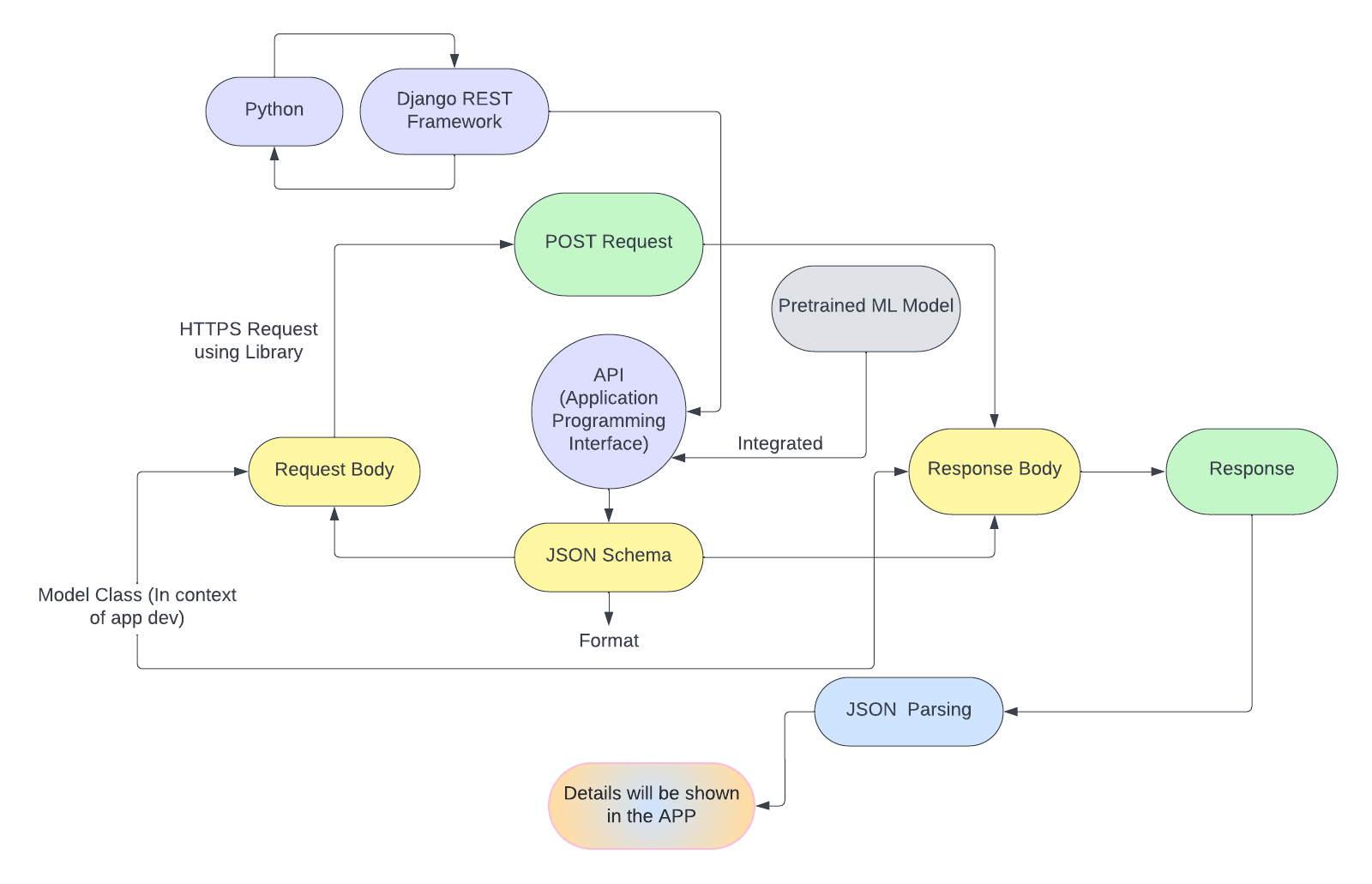
**6. Flatten layer:** After max pooling, a flatten layer transforms the pooled feature map into a single column that is passed to the next layers, making it possible to connect convolutional and pooling layers to dense layers.

**7. LSTM layer:** The flattened output is then fed into an LSTM layer. LSTM is a type of recurrent neural network capable of learning long-term dependencies. It processes the data sequentially and retains information over intervals with its memory cells.

**8. Output layer:** The final step in the CNN-LSTM model is the output layer, which typically consists of dense neurons with an activation function suited to the task; in this case, it would likely be a sigmoid activation function for binary classification.

**9. Binary Classified output:** The output from the model is a binary classified output, which means the model predicts one of two classes for each input sample, such as 'yes' or 'no', 'true' or 'false', '1' or '0', etc.

**2.2.2 API Workflow**

****

**Explanation:**

**1. Python:** The programming language used for developing the backend or server-side logic is Python.

**2. Django REST Framework:** This is a powerful and flexible toolkit built on top of the Django framework to create web APIs. It is used to handle HTTP requests and responses.

**3. HTTPS Request using Library:** An HTTPS request is made to the API using a library, which is not specified in the flowchart but is commonly a part of web development in Python (such as requests).

**4. Request Body:** The body of the HTTPS request, which contains the data sent to the API, presumably in JSON format.

**5. JSON Schema:** The request data is structured according to a JSON schema, which defines the format and type of data expected by the API.

**6. POST Request:** This is the method used to send the request body to the API. A POST request is typically used to create or update resources on the server.

**7. API (Application Programming Interface):** This is the interface through which the client and server communicate. It processes the request and interacts with the ML model.

**8. Model Class (In context of app development):** This refers to a class (in the object-oriented programming sense) that represents the model within the application's codebase, likely used to handle data and interact with the ML model.

**9. Pretrained ML Model:** This is the machine learning model that has already been trained on data and is ready to make predictions or analyses based on new input.

**10. Integrated:** The ML model is integrated with the API, meaning the API has direct access to the model for making predictions.

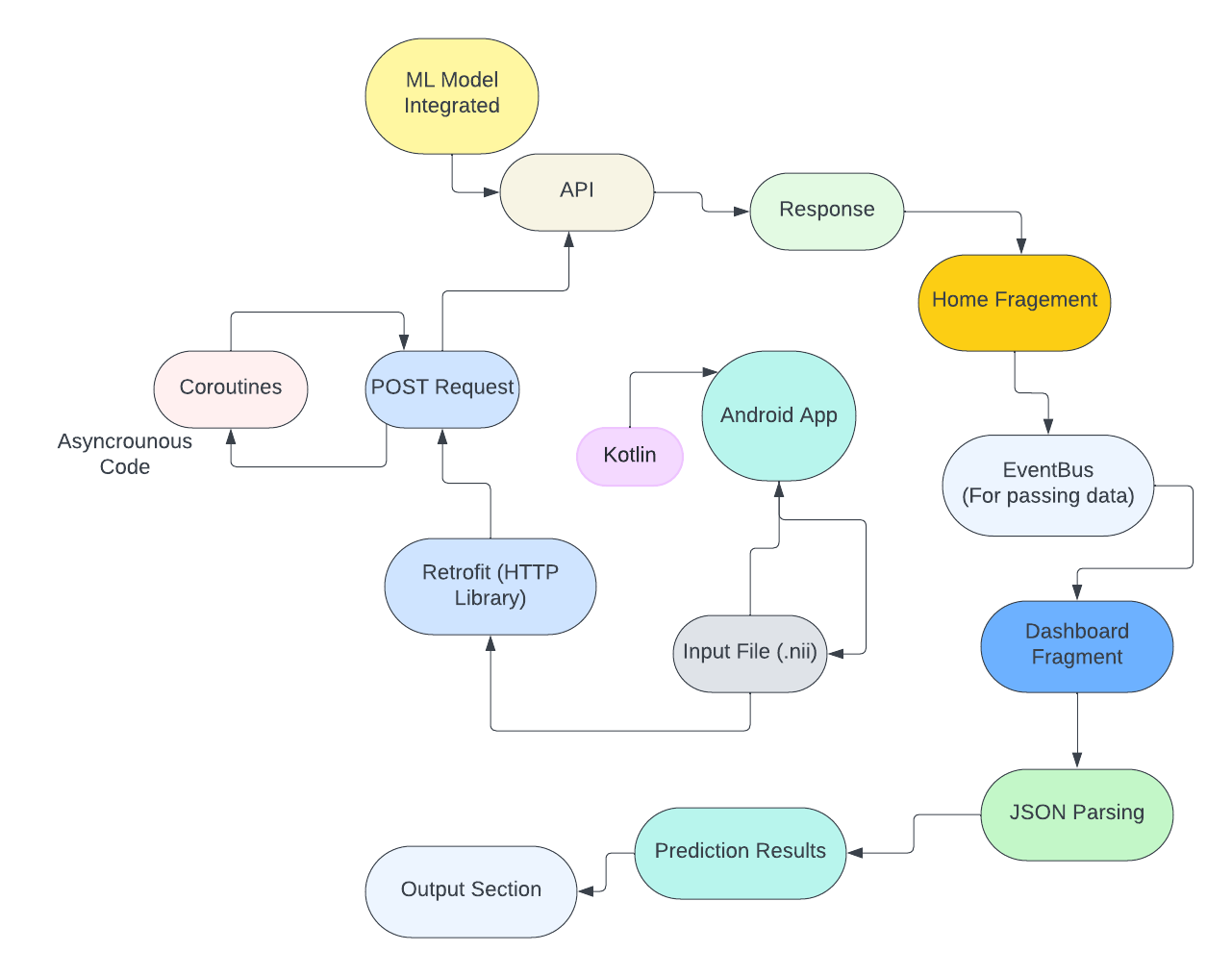
**11. Response Body:** After the ML model processes the input data, it sends back a response body, which is the result of the prediction or analysis.

**12. Response:** This is the full HTTP response sent back to the client from the server, which includes the response body and other HTTP headers.

**13. JSON Parsing:** The response body is parsed from JSON format to be processed or displayed within the app.

**14. Details will be shown in the APP:** The parsed data is then displayed to the user in the app's user interface.

**2.2.3 App Workflow**

****

**Explanation:**

**1. ML Model Integration:** The Android app has a machine learning model integrated into it. This model is likely used to make predictions or analyse data.

**2. API:** The app interacts with an API (Application Programming Interface), which could be a server-based web service. The interaction is done through a network call to send or receive data.

**3. POST Request:** To communicate with the API, the app makes a POST request. This is a type of HTTP request used to send data to a server.

**4. Retrofit (HTTP Library):** The app uses Retrofit, which is a type-safe HTTP client for Android and Java, to manage network requests and responses.

**5. Coroutines:** The network call is handled asynchronously using Kotlin coroutines. Coroutines allow for non-blocking network calls, which means the main thread isn't halted while the app waits for the network response.

**6. Android App (Kotlin):** The overall application is built using Kotlin, which is a modern programming language that is officially supported for Android development.

**7. Input File (.nii):** The input for the machine learning model appears to be a file with the extension .nii, which could be a neuroimaging file format.

**8. Prediction Results:** After the POST request is processed, the machine learning model returns prediction results.

**9. Output Section:** The results from the prediction are then sent to an output section within the app, which is likely a part of the UI where the results are displayed to the user.

**10. Home Fragment:** A fragment within the app called "Home Fragment" serves as a part of the app's home screen or initial user interface.

**11. EventBus (For passing data):** EventBus is a library that simplifies communication between different parts of the app. It is used here to pass data between fragments or components of the app.

**12. Dashboard Fragment:** Another fragment called "Dashboard Fragment" seems to be used for displaying a dashboard to the user, which could include the prediction results and other relevant information.

**13. JSON Parsing:** The data received from the prediction results is parsed from JSON format. JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.

**14. Response:** Finally, the API sends a response back to the app, which could be the result of the POST request, including any data or confirmation needed by the app to proceed.

**Individual Contribution of each member:**

**Mohammad Ammar (22BCE10394):** Collaborating in Training, configuration and Development of Machine Learning Model

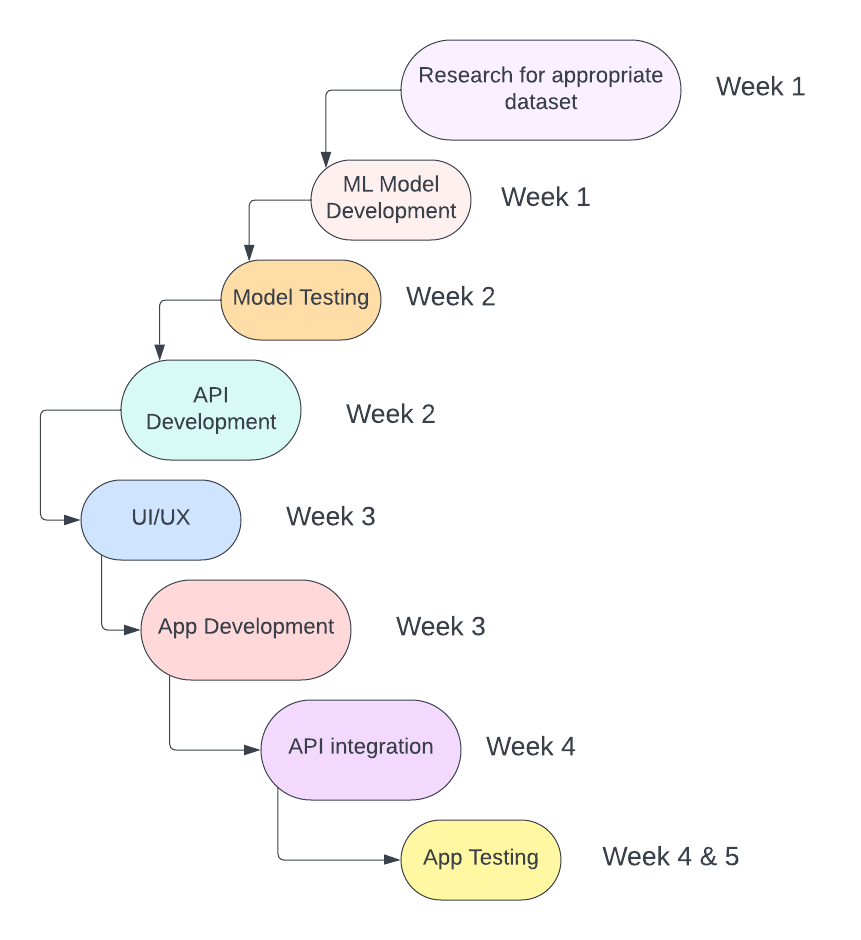
**Manoj Kumar Pradhan (22BCE11415):** Collaborating in Application Building, API Building

**SK Sahil Islam (22BCE10440):** Collaborating in Application Building

**Shishir Agarwal (22BCE10771):** Collaborating in Training, configuration and Development of Machine Learning Model

**Rushikesh Raut (22BCE10965):** Collaborating in Training, configuration and Development of Machine Learning Model

**Project Timeline (Expected) :**

****