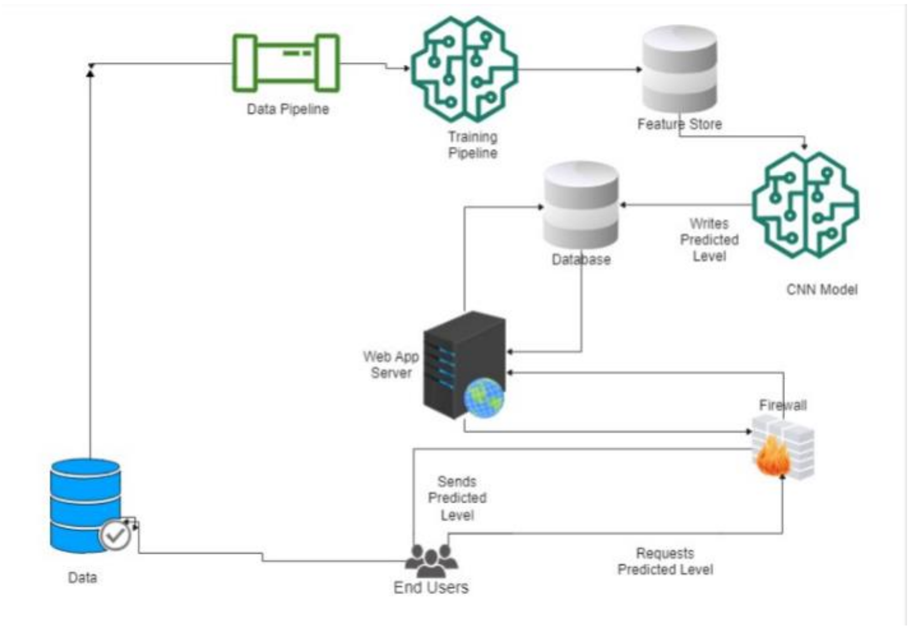
Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	14 November 2023
Team ID	Team-592664
Project Name	T20 Totalitarian: Mastering Score Predictions
Maximum Marks	4 Marks

Technical Architecture:

T20 Totalitarian: Mastering Score Predictions



Guidelines:

1. Use a variety of data sources to train the CNN model. This will help to improve the accuracy of the model.
2. Preprocess the data carefully to remove any noise or outliers.
3. Use a CNN architecture that is well-suited for the task of predicting T20 scores.
4. Train the model for a sufficient number of epochs to achieve good accuracy.
5. Save the trained model to a web server so that it can be accessed by the web app.
6. Develop a web app that is user-friendly and easy to use.
7. Deploy the web app to a production environment so that it is accessible to users.

Table-1 : Components & Technologies:

S.No	**Component**	**Description**	**Technology**
1.	Data Acquisition	**Data Collection:** Downloading historical T20 cricket match data from reputable sources like Kaggle	Kaggle
2.	Data Preprocessing	**Data Cleaning:** Removing duplicate or incomplete records, handling missing values, and ensuring data consistency.	Python, Pandas
3.	Data Exploration	**Exploratory Data Analysis (EDA):** Analyzing data distribution, identifying patterns, and visualizing relevant statistics.	Python, Matplotlib, Seaborn
4.	Model Training	**CNN Model Design:** Defining the CNN architecture, including layers, activation functions, and filter sizes.	Tensorflow/PyTorch
5.	Model Optimization	**Hyperparameter Tuning:** Optimizing hyperparameters to improve model performance.	Optuna, Hyperopt
6.	Model Evaluation	**Model Validation:** Evaluating model performance on unseen data to assess generalization ability.	K-fold cross-validation, confusion matrix, ROC curve
7.	Web App Development	**Web Framework Selection:** Choosing a suitable web framework like Flask for backend development.	Flask
8.	Frontend Design	**User Interface (UI) Design:** Creating an intuitive and user-friendly interface using HTML, CSS, and JavaScript.	HTML, CSS, JavaScript
9.	Backend Integration	**Backend Logic Implementation:** Implementing functions for data processing, model loading, and score prediction.	Python, web framework
10.	Deployment Strategy	**Deployment Platform Selection:** Choosing a deployment platform like cloud hosting or a self-hosted server.	Cloud hosting, self-hosted server
11.	Performance Monitoring	**Monitoring and Maintenance:** Monitoring web app performance, addressing user feedback, and updating the model with new data.	Web analytics tools, user feedback mechanism

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Embrace open-source frameworks like TensorFlow, PyTorch, and Flask for their flexibility, community support, and ongoing development.	TensorFlow, PyTorch,Flask
2.	Security Implementations	Prioritize user security by implementing robust authentication and authorization mechanisms, such as user roles, access control lists, and secure data storage protocols.	OAuth 2.0, HTTPS encryption
3.	Scalable Architecture	Leverage a 3-tier architecture with microservices to scale the web app horizontally, handling increasing user traffic and data volume seamlessly.	Docker, Kubernetes
4.	Additional Features	Enhance the web app with valuable features like team rankings, player statistics, match history, and real-time live score tracking for engaging user interactions.	APIs, WebSockets
5.	Availability	Implement strategies to ensure the web app is highly available, such as load balancing, fault tolerance, and disaster recovery.	Load balancers, redundant servers, disaster recovery plans
6.	Performance	Optimize the web app's performance to ensure fast response times and low latency, even under heavy load.	Caching, asynchronous processing, database optimization