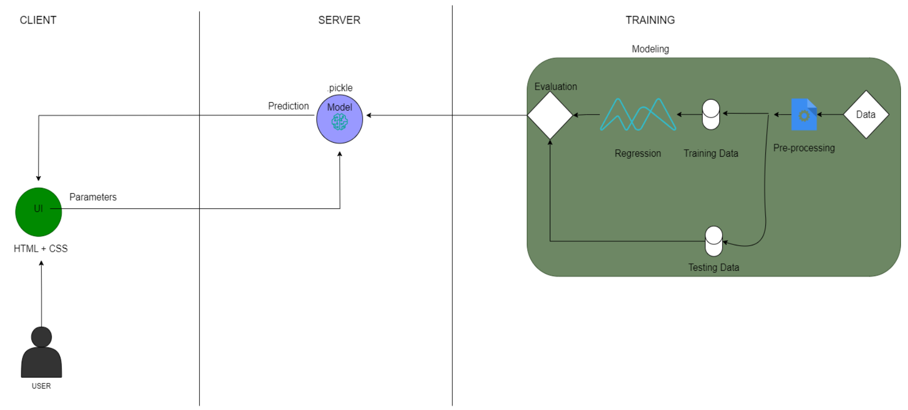
Project Design Phase-II

Technology Stack (Architecture & Stack)

Date	28 October 2022
Team ID	592514
Project Name	Machine Learning Approach for Predicting the Rainfall
Maximum Marks	4 Marks

Technical Architecture:



Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How the user interacts with the Web UI for rainfall prediction	HTML, CSS, JavaScript, React JS
2.	Application Logic	Dive into Datasets and Pattern Recognition	Python (Pandas, Scikit-Learn)
3.	Application Logic-2	Flask Web Application Development	Python (Flask)
4.	Application Logic-3	Data Visualisation	Python (Matplotlib, Seaborn)
5.	Database	Collecting dataset for analysis and prediction	Cloud/ Gdrive, MySQL,colab etc.
6.	File Storage	File storage requirements	Local File System
7.	Model Evaluation	Assessing model performance and accuracy	Accuracy score, Confusion matrix, Roc-Auc Curve
8.	External API-1	Weather Data API	Appropriate weather data API or Relevant external API for additional data
9.	Machine Learning Model	Rainfall Prediction (Regression) Model	Convolutional Neural Network (CNN)
10.	Infrastructure or Deployment	Application Deployment	Anaconda, Python, Local Host & Flask

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	React js, pandas, skit-learn, flask, matplotlib, seaborn, MySQL
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Cross-Site Scripting (XSS) Protection, Content Security Policy (CSP), HTTPS (TLS/SSL), CSRF Protection, CORS (Cross-Origin Resource Sharing)
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Elastic Load Balancing, EC2 instances, RDS Multi-AZ, Auto Scaling
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	ELB, Auto Scaling, Route 53
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Auto Scaling, Read replicas, ELB, Cloudfront

References:

https://c4model.com/

https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/

https://www.ibm.com/cloud/architecture

https://aws.amazon.com/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d