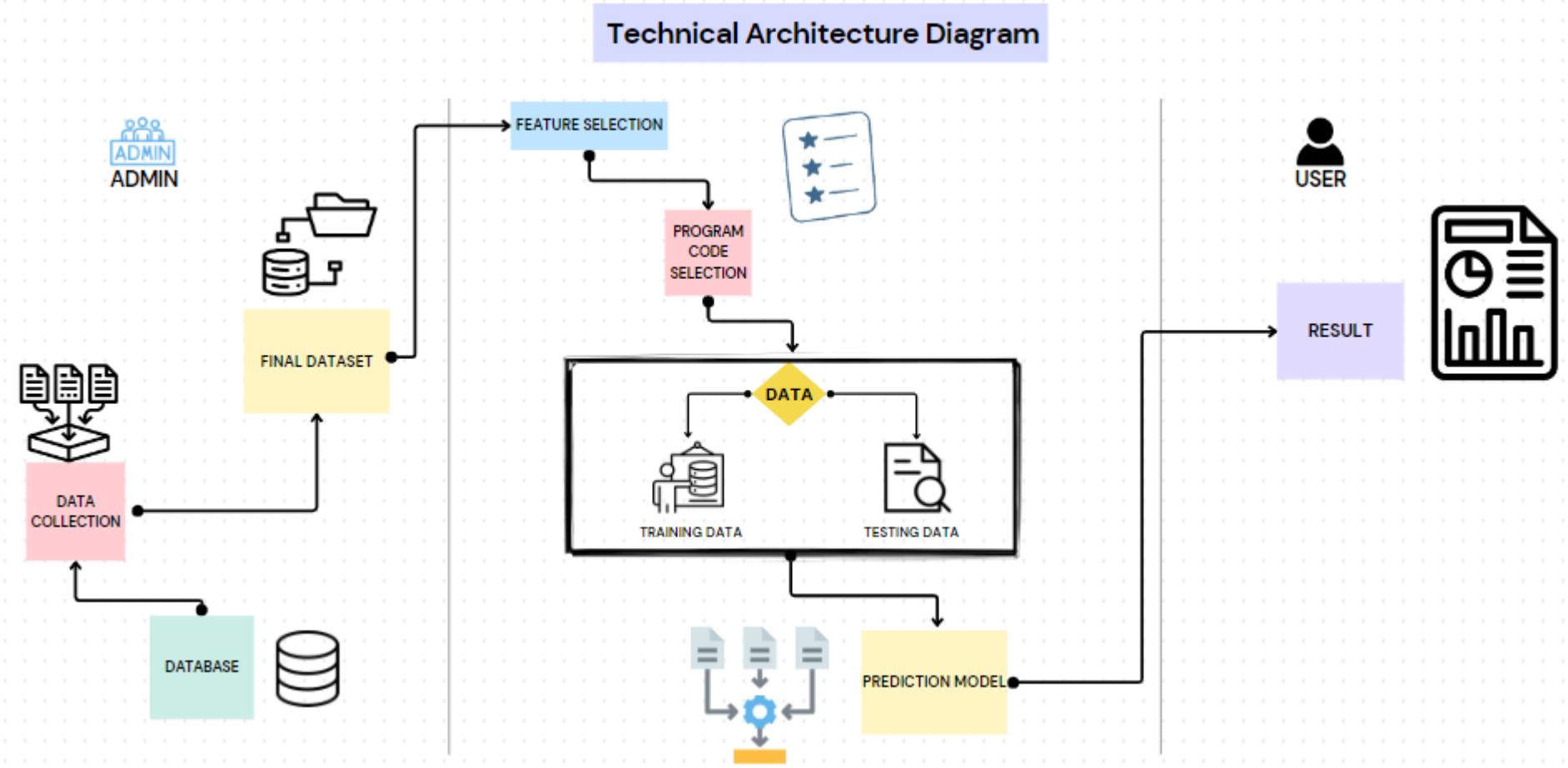
Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	**14 November 2023**
Team ID	**Team-591645**
Project Name	**Diabetes Prediction Using Machine Learning**
Maximum Marks	**4 Marks**

Technical Architecture:

Designing a technical architecture for diabetes prediction using machine learning involves several components and steps. Below is a high-level overview of the key components and their interactions. Keep in mind that the specifics may vary depending on the exact requirements, data availability, and the machine learning algorithms chosen.


Technical Architecture Diagram
ADMIN
ADMIN
FEATURE SELECTION
PROGRAM CODE SELECTION
USER
RESULT
FINAL DATASET
DATA COLLECTION
DATA
TRAINING DATA
TESTING DATA
DATABASE
PREDICTION MODEL

Guidelines:

1. Set Clear Goals:
   Clearly outline what you want the diabetes prediction system to achieve.
2. Clean Data, Protect Privacy:
   Make sure your data is clean and handle it responsibly to respect patient privacy.
3. Choose Helpful Features:
   Pick relevant details from the data to make your prediction model more accurate.
4. Pick the Right Model:
   Choose a machine learning approach that fits your problem and check how well it works.
5. Put Your Model to Work:
   Deploy your model in a way that makes it easy to use and connect with other systems.
6. Keep an Eye on Performance:
   Watch how your model is doing, and regularly update it with new information for better results.
7. Follow Rules, Stay Secure:
   Stick to healthcare rules, especially for patient data, and keep everything secure.

-

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Users can access the application through a web browser on their desktop or laptop.	HTML, CSS
2.	Application Logic-1	User Input Handling: When a user accesses the web application, display the homepage. Data Validation and Preprocessing: Preprocess the input data, such as normalizing and scaling numeric values, handling categorical data, and ensuring completeness	Python
3.	Application Logic-2	Machine Learning Model Integration: Load the trained machine learning model into the web application. Results Display: Display the prediction result to the user along with relevant information about the risk level.	HTML,CSS
4.	Application Logic-3	Built a web application using flask	Flask
5.	Database	A simple website is made using web development and any servers can be added based on requirement	MySQL or SQLite
6.	Cloud Database	We have the flexibility to choose for any cloud Databases	Amazon RDS (Relational Database Service etc
7.	File Storage	Dataset storage, Model Storage	Local Filesystem

8.	External API-1	Health data APIs act as bridges between applications and diverse health-related information sources, fostering the development of applications that support users in monitoring and improving their health and well-being.	Health Data API
9.	External API-2	Medical Data API provides access to relevant research findings, clinical trial data, or the latest information on diabetes treatments and medications. This additional data could enhance the project's accuracy and relevance, contributing to more informed predictions and recommendations.	Medical Data API
10.	Machine Learning Model	It uses machine learning models (K Nearest Neighbors, Support Vector Machine, and Naive Bayes) to analyze patterns in the data and make predictions. The goal is to identify people at high risk of diabetes early, allowing for timely intervention and prevention. The project contributes to healthcare by enhancing the early detection of diabetes, leading to better health outcomes for individuals and communities.	K Nearest Neighbors (KNN),Support Vector Machine (SVM) and Naive Bayes
11.	Infrastructure (Server)	Application Deployment on Local System Local Server Configuration: http://127.0.0.1:5000/	Local

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask:	HTML/CSS,Python,Medical Data API,MySQL or

		● Framework for building the web application backend. HTML/CSS: ● Standard technologies for creating the web application's user interface. Python (including Scikit-learn, Pandas, NumPy): ● Used for machine learning model integration, data processing, and implementing predictive algorithms. MySQL or SQLite: ● Open-source databases used for storing and managing application data.	SQLite,Pandas,NumPy.
2.	Security Implementations	User Authentication: ● Securing access to the application through user login with usernames/passwords or other secure authentication methods. Firewall Protection: ● Using firewalls to monitor and control incoming and outgoing traffic to prevent unauthorized access. Encryption for Data Protection: ● Encrypting sensitive data during transmission (SSL/TLS) and storage to prevent unauthorized access. Input Validation:	SHA-256 ,Web Application Firewalls (WAF),OpenSSL ,SSL/TLS,Logging Frameworks,SIEM Tools.

		• Checking and validating user inputs to prevent common security threats like injections or cross-site scripting. Regular Software Updates: • Ensuring all software components are updated with security patches to fix vulnerabilities. Server Security Measures: • Implementing server hardening techniques to secure the server environment and limit unnecessary access. Security Monitoring: • Using tools to monitor the system for security incidents and abnormal activities	
3.	Scalable Architecture	Three-Tier Architecture: • User Interface: Can handle more users by using smart techniques like caching and load balancing for a smooth user experience. • Backend Logic: Able to manage increased data processing demands by adding more servers to handle computations. • Data Storage: Can accommodate more data by expanding the database storage either vertically (more resources) or horizontally (across	HTML/CSS,Flask, Django,MySQL,HTTP,Pandas, NumPy,Docker,Kubernetes.

		multiple servers). **Microservices Scalability:** Service Decomposition: ● Different Lego Blocks: Each important task, like handling predictions or getting data, is like a different Lego block. ● Building Independently: If one block needs to get bigger (more predictions or more data), only that specific block gets extra pieces, while the rest of the blocks stay the same. Loose Coupling and Fault Isolation: ● Blocks Don't Stick Together: These Lego blocks are separate and don't affect each other's size. If one block needs more pieces, only that block gets them—others don't change.	

S.No	Characteristics	Description	Technology
4.	Availability	Load Balancers: ● Load balancers share website visitors across multiple servers, preventing any single server from getting	MySQL replication,Google Cloud,NGINX,AWS Route 53.

		overwhelmed. This keeps the site running smoothly, even during busy times. Distributed Servers: ● Why: Having the website on multiple servers in different places prevents the whole site from going down if one server fails. Redundancy and Backup Plans: ● Why: Backup systems are like spare parts. If something breaks, there's always another ready to step in and keep the site going. Cloud Services: ● Why: Cloud services provide a super reliable place to host websites with lots of safety features. Monitoring and Quick Fixes: ● Why: Keeping an eye on the website constantly helps spot any issues before they become big problems. Automated systems can fix these issues fast.	

5.	Performance	Using Cache and CDN: ● Save Frequently Used Stuff: Store commonly used data in a special place to quickly show it again without asking the main server (Cache). Handling Requests: ● Even Sharing: Use tools to share user requests evenly among different servers to prevent any server from getting too busy. Handling Requests: ● Even Sharing: Use tools to share user requests evenly among different servers to prevent any server from getting too busy. Compression and Smaller Files: ● Use Smaller Versions: Making website parts like code or images smaller to load them faster (Minification). Smart Resource Use: ● Group Things Together: Put similar things together to need fewer requests when loading a webpage (Resource	LazyLoad,GZIP,MySQL,Redis, NGINX.

		Bundling). Watching and Fixing: ● Keep an Eye on Speed: Always watch how fast the site is and fix things that make it slow (Performance Monitoring).	

References:

https://www.kaggle.com/code/ahmetcankaraolan/diabetes-prediction-using-machine-learning

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

https://aws.amazon.com/architecture

https://www.canva.com/templates/EAE20DscRr4-colorful-context-system-concept-map-infographic/

https://www.canva.com/templates/EAFSwXEEpEA-shopping-process-flow-graph/

https://www.canva.com/templates/EAFQNdypCxI-green-simple-ui-design-flowchart-infographic-graph/