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

Date	**1 November 2023**
Team ID	**Team-592006**
Project Name	**FetalAI: USING MACHINE LEARNING TO PREDICT AND MONITOR FETAL HEALTH**
Maximum Marks	**4 Marks**

Technical Architecture:

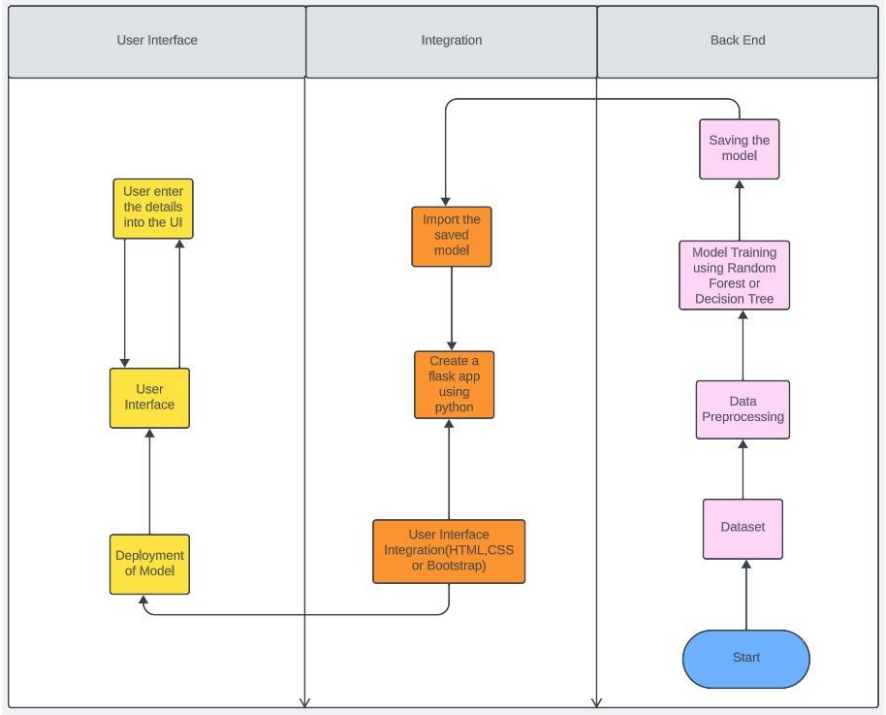

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	Data Collection Sensors	Fetal heart rate sensors, uterine activity sensors	Medical-grade sensors and transducers
2.	Data Processing Module	Data pre-processing, signal processing, feature extraction	Python, MATLAB
3.	Machine Learning Model	Fetal distress prediction model	TensorFlow, scikit-learn
4.	Database	Data Type, Configurations, etc.	Relational or NoSQL database for storing monitoring data
5.	Cloud Integration	Cloud storage for remote access and data backup	Amazon Web Services (AWS), Google Cloud Platform (GCP)
6.	User Interface	Patient and healthcare professional interfaces	Web-based dashboard, mobile application
7.	Alerting System	Real-time alerts for abnormal fetal conditions	Email, SMS, push notifications
8.	Security	Data encryption, user authentication	SSL, OAuth
9.	Reporting	Generation of reports and charts	Aadhar API, etc.
10.	Machine Learning Model	Purpose of Machine Learning Model	Reporting libraries, data visualization tools
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used in the application.	Technology stack of the open-source framework, e.g., Django (Python), Angular (JavaScript), Spring Boot (Java), etc.
2.	Security Implementations	List all the security and access controls implemented, including the use of firewalls, encryption techniques, IAM (Identity and Access Management) controls, and adherence to OWASP (Open Web Application Security Project) guidelines.	Technology or cryptographic methods used, e.g., SHA-256 for data hashing, SSL for encryption, Role-Based Access Control (RBAC), and security libraries..
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Technology stack and architectural patterns that enable scalability, e.g., Docker and Kubernetes for containerization and orchestration in microservices, or load balancers for horizontal scaling in a 3-tier architecture.
4.	Availability	Justify the availability of the application, such as the use of load balancers, distributed server configurations, or other redundancy mechanisms.	load balancers like NGINX or HAProxy, distributed databases like Cassandra or MongoDB, or failover mechanisms.
5.	Performance	Design considerations for the application's performance, including metrics like the number of requests per second, the use of caching mechanisms, and Content Delivery Networks (CDNs).	Redis for caching, CDN services like Cloudflare or Akamai, load testing frameworks like Apache JMeter, or optimization libraries for database queries.