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

Date	27th October 2023
Team ID	592463
Project Name	Psycare
Maximum Marks	4 Marks

Technical Architecture:

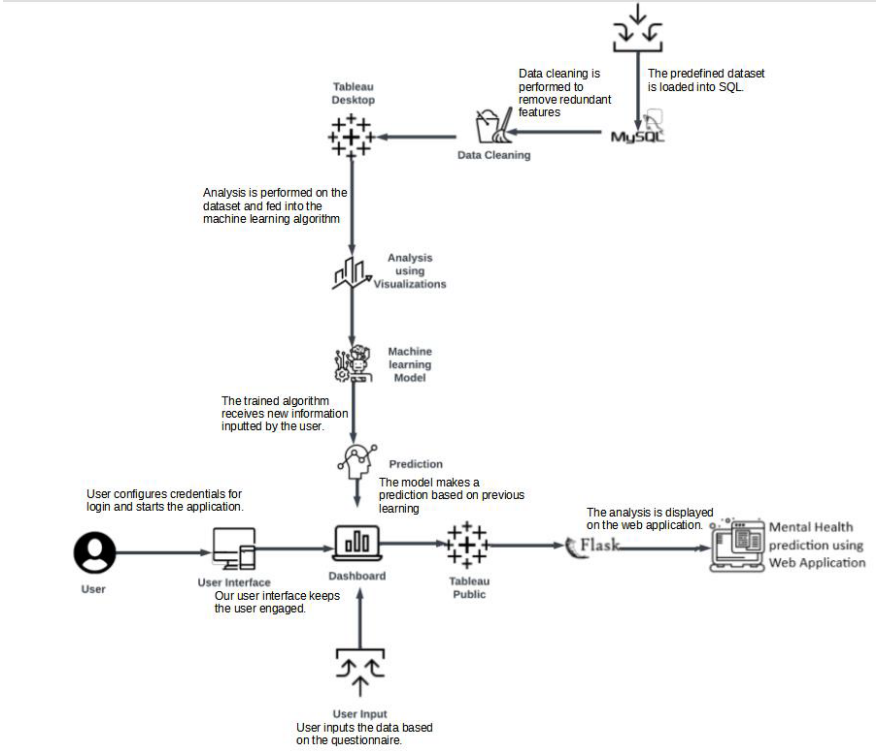

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How the user interacts with web application.	HTML, CSS, JavaScript, React Js etc.
2.	User Authentication	Authentication and security	Firebase, OAuth2
3.	Database	Data Type, Configurations etc.	MySQL
4.	Infrastructure (Server)	Application Deployment on Local System	Local, Cloud Foundry, Kubernetes, etc.
5.	Data collection	Gather a diverse dataset of 1200 employees from different regions of the country across various age groups.	Dataset collection
6.	Data preprocessing	Cleaning, visualization, removal of nulls and redundant data. Removing stop words, tokenizing data and handling special characters.	NumPy, seaborn, etc.
7.	Model Architecture	Utilize word embeddings to convert words into numerical representations suitable for deep learning. Implement an RNN-based model which can capture sequential information in the text.	Word2Vec, GloVe, PyTorch
8.	Training	Train the model on the preprocessed dataset, utilizing GPU/CPU resources, and fine-tune model weights to minimize classification errors.	GPU/CPU, Deep Learning Frameworks
9.	Evaluation	Assess the model's performance and accuracy through metrics.	Metrics such as F1 score/ correlation
10.	Deployment	Integrate the trained model into applications for solutions, involving web app development for user accessibility.	Flask, python

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	React JS, Node JS, flask
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Firebase, OAuth2
3.	Scalable Architecture	Designing the app's architecture for scalability, easy maintenance, and future updates.	Model-View-ViewModel (MVVM) architecture: Separation of concerns for easier scalability. Repository pattern: Isolating data access for flexibility and testability. Dependency Injection: Dagger Hilt for managing dependencies and scalability
4.	Availability	Ensuring the web app is available to users with minimal downtime	Load Balancing: Distributing incoming network traffic to maintain availability. Server redundancy: Multiple server instances to ensure availability in case of server failures.
5.	Performance	Optimizing the web app's speed and responsiveness	Caching: Use of local caching to reduce network requests. Image Compression: Compressing and optimizing images for faster loading. Background Tasks: Implementing background processing to keep the app responsive. Profiling and optimization tools: Android Profiler and tools for performance analysis and improvement