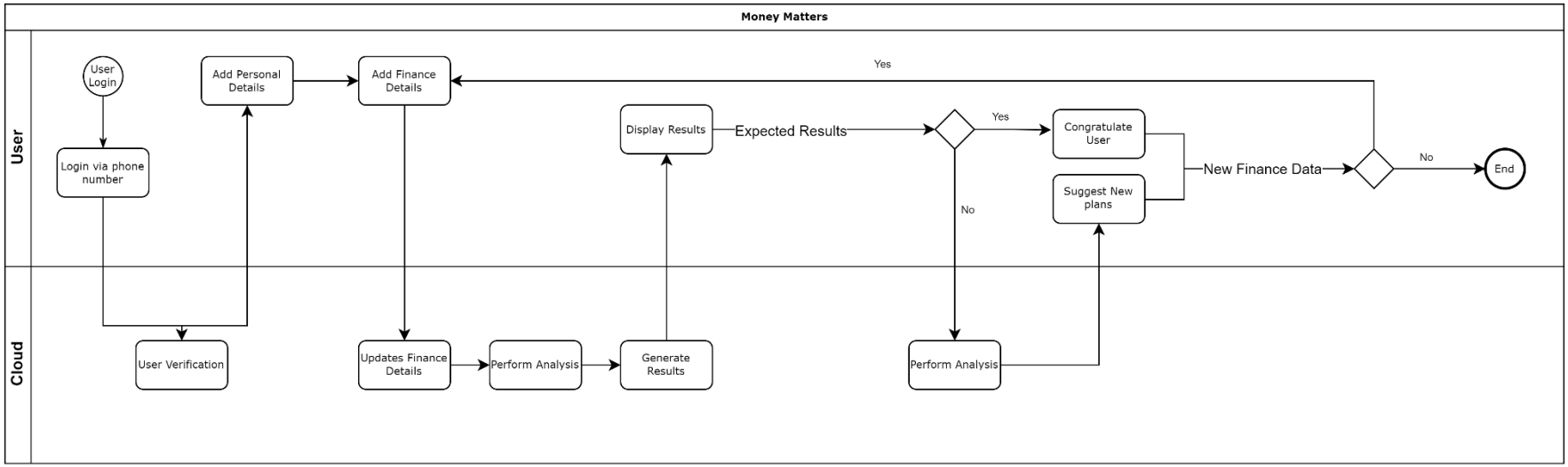
Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	27th October 2023
Team ID	Team-591008
Project Name	Money Matters
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



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:

Sr. No.	Component	Description	Technology
1	User Interface	Mobile App	Kotlin Programming Language, Android Studio, Gradle, Android Jetpack, Firebase, Retrofit, Glide/Picasso, Koin/Dagger 2, Coroutines, JUnit and Mockito
2	Encryption Algorithm	Securing sensitive Data	RSA, AES
3	Data Mining Algorithm	Discover patterns, insights from collected Data	Apriori, K-Means
4	Graph Algorithm	Model relationships between different financial entities, such as transactions, expenses and income sources	Dijkstra's Algorithm, Breadth-First Search
5	Hashing Algorithm	Data integrity and security by generating unique hash codes for data sets	SHA-256, MD5
6	Database	Data type, Configurations, etc.	RDBMS like MySQL, PostgreSQL, or SQLite
7	Cloud Database	Database service on cloud	Amazon RDS, Google Cloud SQL, or Azure SQL Database

8	File Storage	File storage requirements	IBM Block Storage
9	External API - 1	Payment Gateway API (for transactions and payments)	Stripe API
10	External API - 2	SMS and Email API (for notifications and alerts)	Twilio API
11	Machine Learning Model	For learning spending, saving and target analysis and preparing plans	Regression Models, Classification Models, Clustering, etc
12	Infrastructure (Server/Cloud)	Application Deployment on Local System / Cloud Local Server Configuration, Cloud Server Configuration	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

Sr. No.	Characteristics	Description	Technology
1	Open-Source Frameworks	Open-source frameworks allow for the development of the Money Matters app using community-supported tools, libraries, and frameworks, promoting transparency and collaboration within the development process	Kotlin programming language, Retrofit for networking, Koin or Dagger 2 for dependency injection, and Android Jetpack libraries
2	Security Implementations	Security implementations are crucial for protecting sensitive user financial data, ensuring data integrity, and preventing unauthorized access to user accounts and transaction details within the Money Matters app	Encryption algorithms like AES for data encryption, SSL/TLS for secure communication, Firebase Authentication for user authentication, and secure storage options such as encrypted databases and secure cloud storage services

3	Scalable Architecture	enables handling varying user loads and data processing requirements effectively, ensuring a seamless user experience even during periods of high traffic and increased data processing demands	Cloud-based infrastructure (e.g., AWS, GCP, Azure) for scalable compute resources, auto-scaling mechanisms for dynamic resource allocation, and cloud-based databases (e.g., DynamoDB, Firestore) for scalable and flexible data storage.
4	Availability	Availability is essential to ensure that the Money Matters app remains accessible to users without interruptions, minimizing downtime and service disruptions that could impact the user experience and financial management processes	oad balancing for distributing traffic, fault-tolerant design for ensuring continuous operation, redundant systems for high availability, and automated monitoring and alerting systems for proactive issue detection and resolution
5	Performance	Performance optimizations are crucial for ensuring that the Money Matters app delivers fast response times, smooth user interactions, and efficient data processing, enhancing the overall user experience and satisfaction	Caching mechanisms (e.g., Redis, Memcached) for improving data retrieval speeds, optimized database queries for efficient data retrieval, and use of asynchronous programming (e.g., Coroutines) for handling long-running tasks without blocking the main application thread