Technical Specification Document

Single Face Pay Application

Prepared by: Poulomi Saha

Date: 13.08.2024

AV

Introduction to the Single Face Pay Application MVP

Purpose:

The Single Face Pay Application MVP is designed for the demonstration of the core functionalities of a secure , seamless and technologically enhanced facial recognition payment system. This document outlines the entire development life cycle of this minimum viable product(MVP) and focusses on key areas like intended audience and key objectives . This MVP will be the early prototype of the main product and will be developed by only incorporating the essential features which are necessary to get the users to experience the potential of the actual facial payment technology which is our main product in real-world.

The primary objective of this MVP is to validate the use of facial recognition along with multi-biometric verification for secure and touchless payments. User feedback will be gathered which will help and guide in development of the prototype and resources will be used efficiently . This MVP is introduced into the market quickly to establish an initial user base, collected insights from the early users which will help in future enhancement of the main product and also to plan for future investment needs.

Scope:

The Single Face Pay Application will mainly focus on the following the following objectives:

a) Validation of Core Functionalities: The efficiency of the facial recognition and multi-biometric verification for secure and touchless payments thereby ensuring a fast and convenient user experience will be tested.
b) Collection of User feedback: Engagement of the early users to gather their insights and feedback on the MVP , which will then be used to refine and enhance the actual product making it fit for the market.
c) Resource Efficiency : During the MVP development the main focus will be on its key features using minimum resources , hence reducing the cost.
d) Strategy for Market entry: The MVP will be introduced quickly to the market potentially without a subscription fee for the use of initial features , thereby attracting users and creating a strong foundation for future growth of user base for the actual product.

AV

Constraints:

The main constraints of this MVP is to maintain high standard to security and privacy while storing and processing user data, ensuring seamless integration with bank and merchant systems and to deliver a intuitive and user-friendly interface for both end-users and merchants.

Background:

The Single Face Pay application project aims to revolutionize the payment systems by removing the need for physical interaction making it user friendly with enhanced security . The development of this MVP is important as the current focus for our startup is to attract investment, risk reduction for final product and focused development of the product. By developing a fully functioning prototype , the Single Face Pay Application MVP will provide the stakeholders a chance to experience the fully functioning product in action thereby helping them to make informed decision along showing the path for enhancements need to the product in future.

This product MVP is mainly based on machine learning, deep learning , neural network and computer vision technologies. Here various libraries such as OpenCV and face_recognition is used to implement facial recognition , pretrained neural network models are also used along with that to enhance the performance. The application will then be integrated on flutter so that its functional in different environments and that will be the user interface for user registration and live image capturing . Amazon RDS is used for securely storing and managing user data only if the application scales else initially SQLite database will be used , secure payment gateways will be integrated to that along with merchant and bank APIs. At the forefront of this product will be to maintain privacy and security at every point.

In conclusion ,the MVP for the Single Face Pay Application represents a signification enhancement in the evolution of payment methods, which gives a glimpse into a future of touchless, secure transactions with the help of facial recognition along with multi-biometric verification technology.

AV

Project Overview

Goals: The AI Mobile Shopping App (MVP) mainly focuses on revolutionizing the user shopping experience by integration of advanced security features into the mobile application from the user as well as merchant side system. The primary goal of the project is to include:

1) Multi-biometric verification: Development of a robust multi-biometric verification system that will enhance security during user verification and transaction processes.

2)AI-integration: Using AI to enhance security features within the application , such as detecting fraudulent activities, verifying users on real-time and providing real-time security updates.

3)Secure Transactions: Ensuring that the payment processing is secure through integration with trusted gateways like Stripe and PayPal which are further safeguarded by multi-biometric verification system.

4)Scalable Architecture : Building a plan to make the backend of the system robust and scalable so that it can handle the user based on growth , thereby ensuring multi-biometric verification and transaction process remain secure and efficient even if the application scales in the future.

Hence, this project mainly focuses of the security of the user's payment transaction, particularly with the integration of the multi-biometric verification system that is central to ensure safe and reliable payment transactions.

Proposed timeline for the Single face pay application(MVP) using Agile methodology:

Here, the project is divided into sprints that align with agile methodology. Each sprint is designed to cover a specific task within the project timeline:

Sprint 1: Planning and Research (2 Weeks)

Timeline: 9/30/2024 - 10/13/2024

Focus:

- Finalize project scope, objectives, and requirements.

AV

- Conduct research on AI ,ML , Computer Vision and face recognition models along with user needs and market trends.
- Initial brainstorming on design approaches.

Deliverables:

- Project plan document.
- Research report.
- Initial requirements and design ideas.

2)**Sprint 2: Design (3 Weeks)**

Timeline: 11/2/2024 - 11/23/2024

Focus:

- Develop wireframes and prototypes for the single face pay application interface.
- Define data flows and user experiences.
- Finalize design guidelines and UI/UX elements.

Deliverables:

- Low-fidelity and high-fidelity prototypes.
- Design documentation and style guide.

3) **Sprint 3: Execution Phase 1 (4 Weeks)**

Timeline: 11/23/2024 - 12/21/2024

Focus:

- Start frontend development using React Native (Typescript/JavaScript).
- Begin backend setup with Python, Flask and SQLite integration.
- Implement initial AI features for the MVP.

Deliverables:

- Basic working version of the mobile app.
- Initial API and database setup.
- Core AI functionalities integrated.

AV

4) **Sprint 4: Execution Phase 2 (4 Weeks)**

Timeline: 12/22/2024 - 1/19/2025

Focus:

- Complete frontend and backend development.
- Refine AI/ML models and algorithms based on research insights.
- Integrate payment gateways (Stripe/PayPal) and secure user authentication.

Deliverables:

- Fully functional mobile app.
- Payment and authentication modules.
- Finalized AI/ML features.

5) **Sprint 5: Testing (3 Weeks)**

Timeline: 1/21/2025 - 2/9/2025

Focus:

- Conduct thorough testing (functional, integration, security) according to the testing plan.
- Debugging and resolving issues.

Deliverables:

- Tested and debugged application.
- UAT feedback.
- Final preparations for deployment as per deployment plan.

6) **Sprint 6: Completion and Pre-Launch (1 Week)**

Timeline: 2/9/2025 - 2/16/2025

Focus:

- Finalize app deployment and prepare support and maintenance according to plan.
- Prepare marketing and launch materials.
- Ensure all systems are operational and secure.

AV

Deliverables:

- Ready-to-launch application.
- Marketing and support materials.
- Deployment plan.

7) **Sprint 7: Commercialization and Launch (4 Weeks)**

Timeline: 2/20/2025 - 3/20/2025

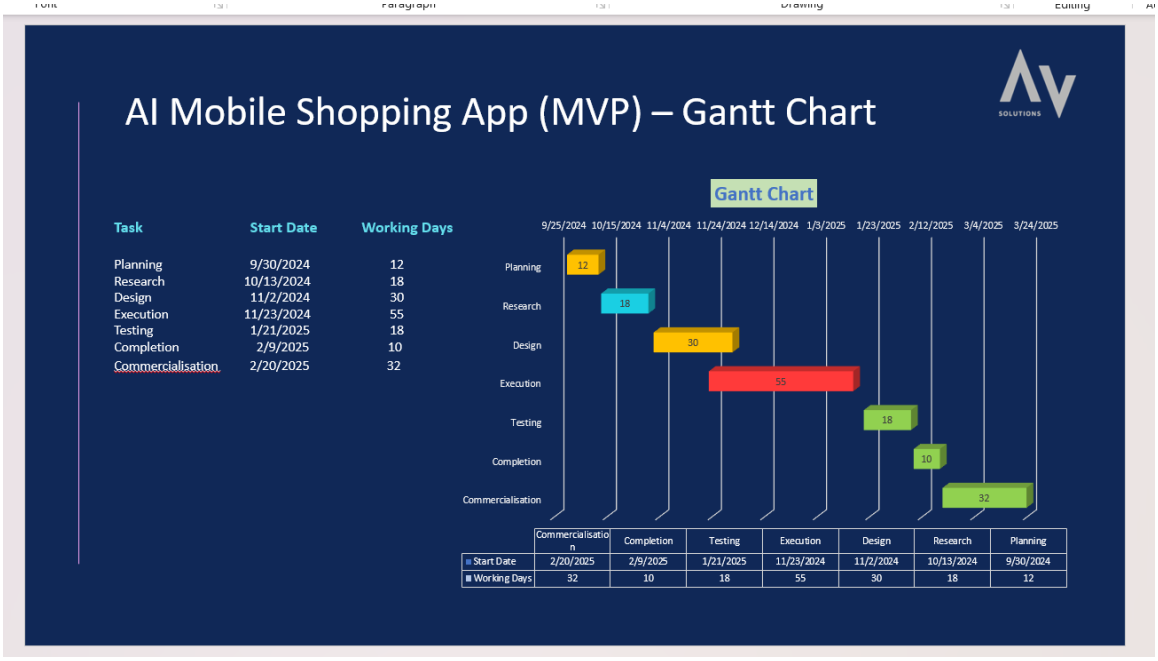
Focus:

- Launch the app on App Store and Google Play.
- Monitor early user feedback and performance.
- Address any post-launch issues or bugs and plan on fixing them according to the support and maintenance plan.

Deliverables:

- Live app on marketplaces.
- Initial user metrics and feedback.
- Post-launch updates if needed

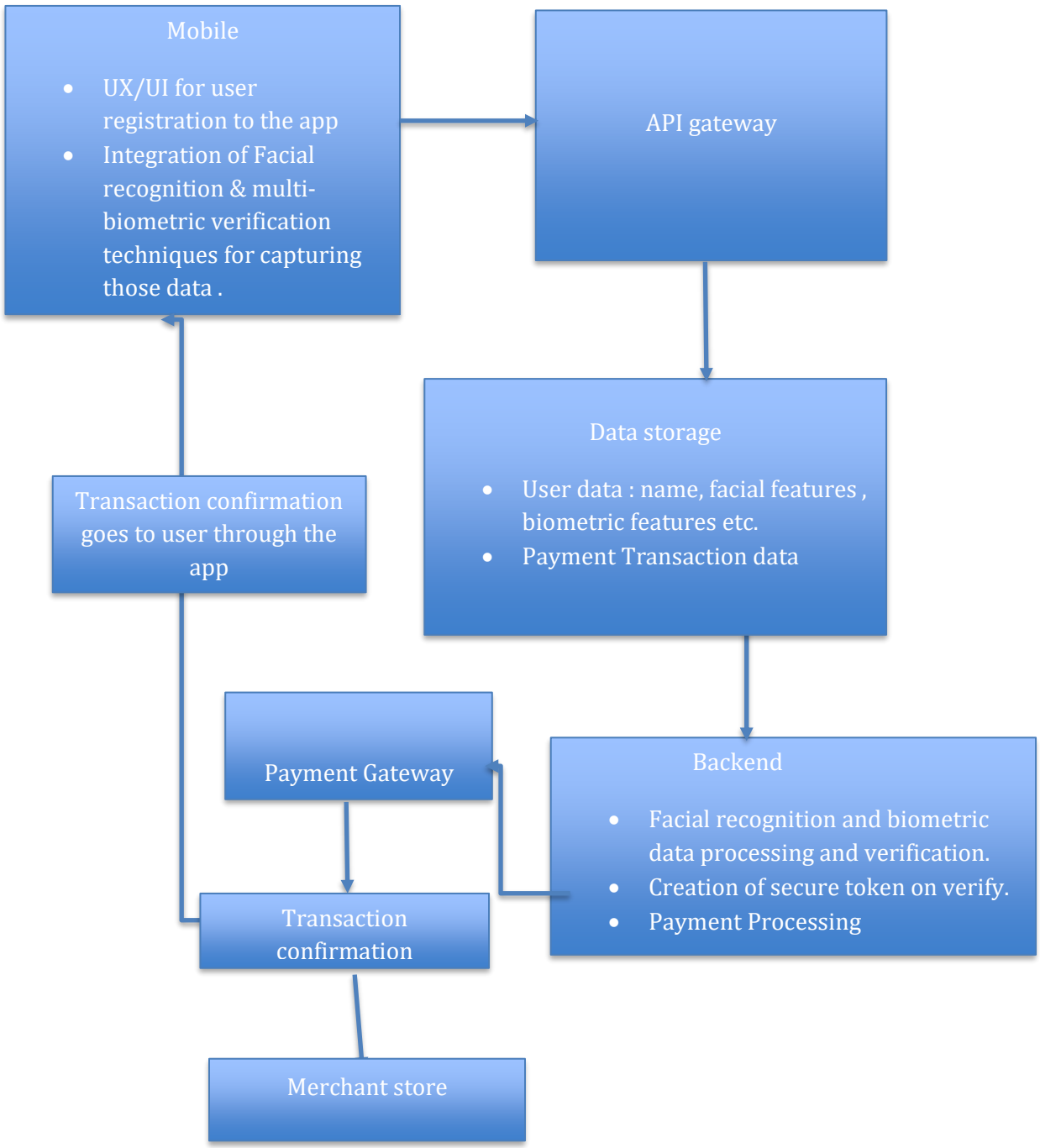
Proposed timeline for the Single face pay application(MVP) using Grantt chart:



AV

System Architecture

High Level overview of the system architecture:



AV

Components of the System:

1) Mobile : The mobile is basically the end-user interface for our app. Its functionalities include user registration where user details and biometric data is captured during first time user registration . This app integrates multi-biometric verification technique to capture and transmit the data in a secure way. It also deals with verifying user in real time ,handling transaction confirmation of verified users, providing users with real-time update on their transaction .

2) API Gateway: The API Gateway here acts a connecting point between the mobile app and the backend system. It helps to route incoming requests from the mobile app to either the databases or the requested backend services as directed by the end-user. It ensures secure and efficient communication between client (mobile app) and server(backend services) and helps to handle various tasks .It is one of the critical component in the application as it centrally manages all the incoming requests and streamline the interaction between the frontend and the backend effectively.

3) Backend: The backend is responsible for multiple functionalities, the most important one being handling the facial and multi-biometric verification process by triggering the front-end to initially capture the facial and biometric data during initial registration and after which it generates a passcode for that user , which the user uses to login to the application after , it helps to capture the live image and biometrics of the user to carry out multi-biometric authentication. Its, also responsible for processing payments by interacting with the payment gateways and ensuring that the transactions are completed successfully for the verified users.

4) Data Storage: It is responsible for securely storing all user-related information , that includes their personal data, facial and other biometric data which are used for the id verification . It is designed to ensure highest level of security with encryption and provide access control to protect sensitive information. This component is very crucial for maintaining confidentiality and integrity of user data in the systems.

5) Payment Gateway: The payment gateway is mainly responsible for verifying the transaction details, processing payments by communicating with the financial institutions , confirming the completion of the verified transactions. Once a payment is processed , a confirmation is sent to the mobile app and the merchant store is also notified.

AV

6) Merchant Store: The merchant store receives the confirmed transaction details and updates its records. This component is there to make sure that the merchant is notified of all the successful transactions. The merchant store is the final endpoint in the transaction process from where the user proceeds the checkout.

Technical Stack:

1) Languages :

- Javascript/Typescript is mainly used for developing the mobile app with React Native Framework.
- Python: Used for development of various backend services which includes multi-biometric verification, API development along with database management using OpenCV, face recognition library and Flask.
- SQL: It will be used for managing SQLite databases and for handling database queries with python in the backend.

Implementation:

- JavaScript/Typescript will be used to build the frontend of the application in React Native to ensure a unified and responsive UX across both IOS and android platforms.
- In this app, Python will be the primary language for all the backend services , such as capturing user biometric details, managing user authentication using multi-biometric verification, handling multiple API routes with Flask and interaction with the SQLite databases.
- SQL will be used to manage SQLite databases where all the user details , transaction details and biometric verification details are stored as its lightweight and suited for small to medium scale application.

2) Frameworks and Libraries:
    - React Native: This framework is used for developing the front-end of the application with cross-platform support.
    - Flask: This lightweight python framework is used for development of the backend services.
    - Flask Restful: This is an extension of the Flask for developing Restful APIs. SQLite: This is a Lightweight database and is file-based that can be

integrated with the help of python and is used for handling various database operations securely.

AV

- OpenCV: This library is mainly for image processing and computer vision tasks .
- face_recognition: This is a python library built on top of dlib for facial recognition tasks with OpenCV
- Stripe/Paypal SDK: These libraries will be used for payment processing integration into this app
- AWS SDK: For integration of optional AWS services.

Implementation:

- React Native will be used for developing the front-end of the mobile application that supports cross-platform and provides a user-friendly and consistent interface.
- Flask serves as the framework that will be used in the backend development for handling user authentication by multi-biometric verification ,API routing and communication with the database.
- Flask-Restful will mainly be responsible to manage the Restful APIs that will interact with the frontend.
- SQLite is the database management system to be used , and will manage storing and retrieval of user data, biometric information , transaction details and other necessary details in a safe way.
- SQLAlchemy can be used to interact with the SQLite database for easy ORM-based queries and transactions.
- OpenCV , face_recognition and facenet will be used for implementation of multi-biometric authentication .
- Stripe/Paypal SDK will be integrated to manage payment processing through the app thereby ensuring safe and reliable transactions
- Bank APIs(Plaid or Bank APIs) will be used to link user's bank account to the app, in case the user wants direct bank transfers.
- AWS SDK maybe optionally used for integrating additional AWS services if needed.

AV

3) Tools:

- Visual Studio Code: This IDE will be used for writing and managing both Python and javascript code.
- Git: Git will be used for version control for managing the codebase and collaborating with other members.
- Postman: Its a API Testing tool used to test the endpoint APIs during development.
- Docker: It will be used for containerizing the Flask backend application thereby maintaining consistency in different environments.
- Jenkins or GitHub Actions: Its a CI/CD tool and mainly used for automating build, test and deployments.
- Firebase TestLab : Testing tool to test the mobile application on real devices.
- AWS CloudFormation: This is mainly used for managing AWS infrastructure(optional)

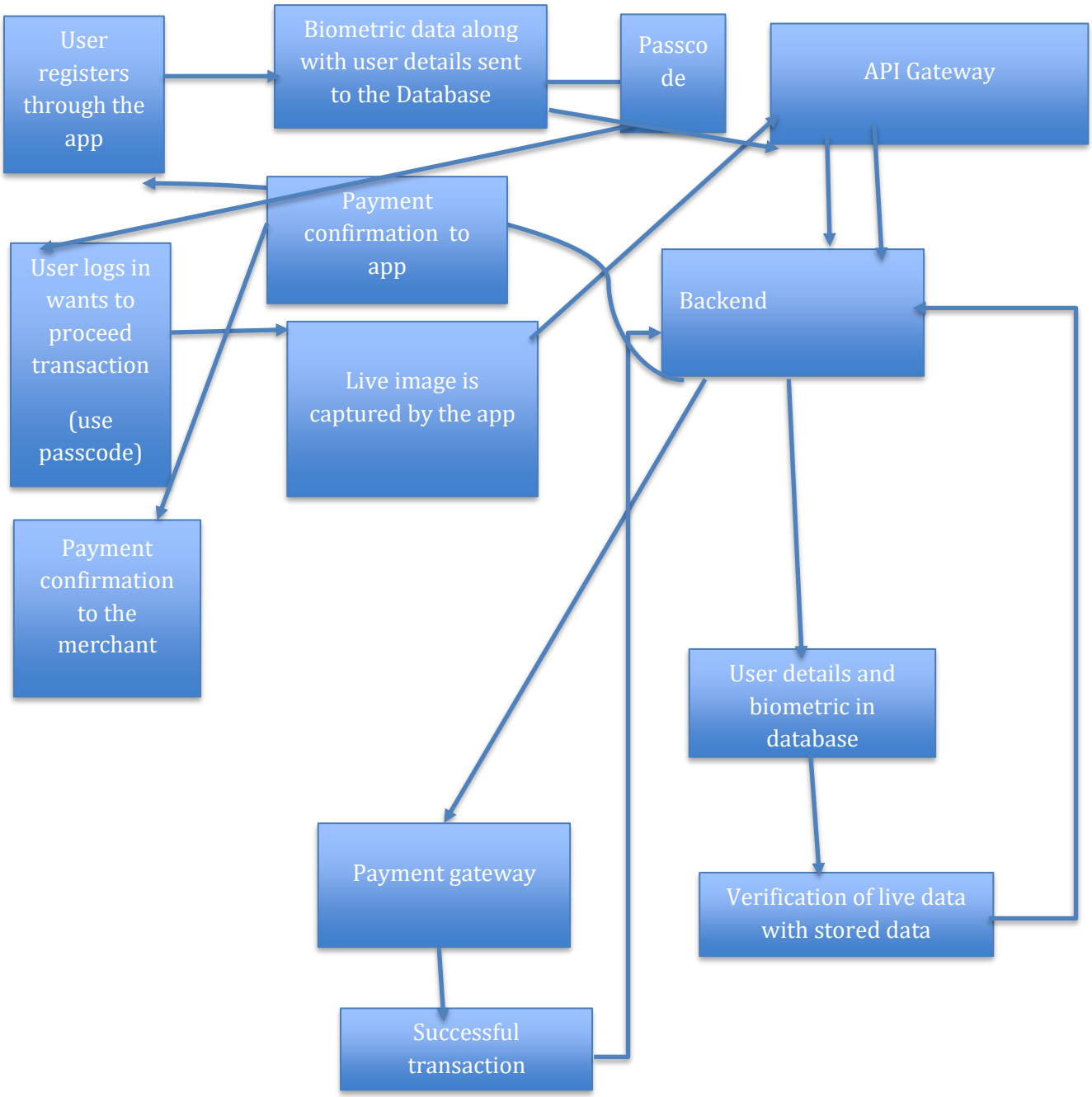
Implementation:

- Visual Studio Code will be used as the primary IDE for development of both the frontend and the backend, thereby providing a comprehensive environment for code development, debugging and version control.
- Git will be used as version control system which enables it to maintain the history of code changes along with collaborative development that will take place.
- Postman will be used to test Flask APIs during development thereby ensuring that the APIs work correctly before being integrated to the frontend of the application.
- Docker will be mainly used to containerize flask application which will allow for consistent deployment across different environments.
- Jenkins or GitHub will automate the test, build and deployment processes which will make the workflow more efficient.
- Firebase Test Lab will mainly be used to test the mobile application on different devices .
- AWS cloudFormation is optional and will only be used to manage AWS infrastructure components in case the project expands beyond its capabilities of SQLite and require cloud integration.

AV

Data Flow and Security

Data Flow diagram for the application:



AV

Data Flow Description for the MVP of the application:

1) User Registration and biometric enrollment initially:
    - Process: Initially the users will have to register through the mobile app and enroll their biometric information and receives a passcode for later login.
    - Data Flow: All these user information is then sent to the API Gateway.

2) API Gateway:
    - Process: The user information goes to the API gateway after registration.
    - Data Flow: The API Gateway then forwards this information to the backend of the application from where it is sent to the database for storage.

3) a) Backend Processing (Initial registration):
    - Process : The backend processes the user biometric information data that it receives and stores them securely in the database.
    - Data Storage : User details and biometrics are stored in secure database.

    b) Backend Processing (User Authentication and verification on subsequent logins):
    - Process: Now , as the user tries to log in to the app using the passcode and make a transaction , a live image (along with other biometric data) is captured through the app.
    - Data Flow: This live captured data is then sent to the API Gateway which is then forwarded by the API gateway to the backend.

    c) Backend Processing (Verification):

    - Process: As the live captured data is sent to the backend , it then compares that data with the stored data in the database.
    - Verification: If a match is found then only payment processing starts.
    - Data Flow: If the user is verified then only the payment processing through the payment gateway starts.

4) Payment Processing:

    - Process: The backend interacts with the payment gateway to handle secure transactions.
    - Data Flow: As soon as the payment is processed , the payment gateway sends a transaction confirmation to the backend.

AV

5) Transaction confirmation:

- Process: After the backend receives the transaction confirmation, it then sends the confirmation to the mobile app via the API Gateway.
- Data Flow: The user receives the transaction confirmation through the app and simultaneously the transaction confirmation is also sent to the merchant store for their record.

AV

Integration and API Specifications

APIs :

1.**User Registration API:**

- Endpoint:/api/register
- Method: POST
- Data Format: User biometric and other details is submitted through the API gateway.
- Description: This API handles the registration of new users into the app and processing of the user data to the backend which then is securely stored in SQLite database.

2. **User Verification API:**

- Endpoint:/api/authenticate
- Method: POST
- Data Format: Requests containing the live biometric data is sent to the back for id verification.
- Description: When users login later after registration, a live biometric data is captured which is compared with stored data, this API handles this verification process. Upon successful verification the payment transaction starts.

3. **Payment Processing API:**

- Endpoint:/api/payment
- Method: POST
- Data Format: User's payment details and payment methods are transmitted for processing the payment
- Description: This API handles payment transactions by interacting with the secured payment gateways.

4. **Transaction confirmation API:**

- Endpoint:/api/transaction-confirmation
- Method:POST

AV

- Data Format: Once the transaction is successful, the transaction confirmation details are sent back to the mobile app and the merchant store through the backend server.
- Description: This API is responsible for sending the transaction confirmation details to the mobile application and the merchant stores from the backend server after it gets indication from the payment gateways of successful transaction.

Integration Points:

1)Integration of Payment Gateways:

- Tools: Stripe/PayPal SDKs
- Purpose: Once a user is verified , the backend server interacts with the payment gateways to handle secure payment transactions.

2) Integration of Bank API:

- Tools: Plaid / direct Bank APIs
- Purpose: This API integration allows the registered user to link their bank accounts directly to the app for direct bank transfers , account verification etc.

3**) Integration of Biometric capture and verification**

- Tools: OpenCV,face_recognition libraries, facenet
- Purpose: To capture, process and verify the biometric information of the users, to ensure that the biometric data of the live user is the same as that of the stored biometric during authentication.

4)Optional AWS Integration:

- Tools: AWS SDK
- Purpose: This maybe needed if additional services such as secure storage or infrastructure management is needed that is if the application scales out beyond its scope.

Data Exchange:

1) Protocols:

- **HTTP/HTTPS**: The data exchanges within the mobile application, API Gateway and the backend will take place over HTTPs thereby ensuring secure communication.
- **RESTful API**: APIs will be Restful and use appropriate HTTP methods for different requests.

AV

2)Data Formats:

- JSON: Data that will be exchanged between the frontend and the backend will be formatted in JSON format for consistency and ease of integration.

3) Authentication:

- JWT Tokens: Secure JSON Web Tokens will be used for authorization of API requests after a user is verified successfully.

4) Encryption:

- SSL/TLS: The transmitted data will be encrypted using SSL/TLS protocols so as to ensure that the sensitive information such as biometric data and payment details are protected during in transit.

- Data Encryption: Sensitive data such as biometric information of the users, payment details that are stored in the databases, they will be encrypted to ensure integrity and confidentiality.

AV

Testing and Quality Assurance

Overall Testing Approach:

1)**Unit Testing:**

- Purpose: This phase is used to validate if each individual component in the application works as expected.
- Focus Area: biometric data capture, API endpoints, biometric data processing function, database interaction, secure token generation, user verification, secure payment transaction.
- Tools: Testing frameworks like PyTest, JUnit etc will be used.

2) **Integration Testing:**

- Purpose: To ensure the data flow between different components in the application is going on seamlessly.
- Focus Area: This phase will be done to check the data flow from the mobile app through the API gateway to the backend and the database followed by payment gateways.
- Tools: Postman for API testing and other integration testing tool.

3)**System Testing:**

- Purpose: This phase of testing is to make sure that the end-to-end functionality of the entire application is working as per expected, that is to ensure that all components work together to meet the requirements.
- Focus Area: User registration, capturing biometric data, passcode generation, capture of live data and user authentication, processing transaction and confirmation of payment.
- Tools: Selenium for UI Testing and JMeter for performance testing.

4) **Security Testing:**

- Purpose: This phase of testing is to ensure that any vulnerabilities in the system is identified and the sensitive information is stored and maintained securely.
- Focus Area: Data encryption, API security , security of the stored biometric and user data.
- Tools: OWASP ZAP, Burp Suite

6) Performance Testing:

- Purpose: This phase of testing is mainly for evaluating the performance of the application under different condition such as higher number of user or higher volume of transaction.
- Focus Area: API response time, transaction processing time
- Tools: JMeter, LoadRunner

6)**User Acceptance Testing(UAT):**

- Purpose: This phase of testing will make sure that the application meets the requirements of the end-users and works as expected and is fit for the market .
- Focus Area: Reliability, usability and functionality from user perspective.
- Tools: Manually conducted user feedback.

AV

Deployment Plan

Environment

1)Hardware:

- Mobile Devices: This application will be deployed on both IOS and android devices thereby ensuring cross platform compatibility.
- Servers: AWS EC2 will be used to host the backend services which will ensure security and scalability ,AWS EC2 or AWS S3 can be used to host the front-end of the application and the database can also be hosted on AWS EC2 as our MVP application will be small scale .To maintain consistency across different environments Docker containers will be used.
- Networking: AWS network components will be used to ensure secure and seamless communication.

2)Software:

- Frontend: React Native with JavaScript/Typescript will be used for development thereby ensuring a unified user experience across platforms.
- Backend: Python and Flask will be used with Flask-Restful API for API management and SQLite for database .
- Database: The primary database will be SQLite for storing user details, transaction details , biometric details and is ,managed by Python through SQLAlchemy for ORM-based queries.
- Security: OpenCV, face_recognition libraries and facenet will be integrated with the backend for multi-biometric authentication.

Deployment Process:

a)Step 1: Preparation

- Codebase Management : Visual studio code will be used as the primary IDE for both frontend and backend development.
- Version Control: For managing the codebase, Git will be used thereby ensuring collaborative development and maintainence of history of changes.
- Containerizations: For containerization the Flask backend , docker will be used ensuring consistency across development, staging and production environment.
- Build Automation: GitHub Actions will be used to automate the build, test and deployment for frontend as well as backend of the application.

AV

b)Step 2: Deployment to Staging Environment

- Backend Development: Docker will be used to containerize flask based backend services and deployment will be done on AWS EC2 instance in the staging environment.
- Database Setup: SQLite database configuration will be done in the staging environment thereby ensuring seamless data storage and retrieval.
- Frontend Deployment: The frontend will be built on React Native application and distributed to testers internally through Firebase Test Lab.
- API Testing : The testing of the Flask Restful APIs will be done by Postman, thereby ensuring they function correctly before their integration with the frontend.

c)Step 3: Staging Testing

- Functional Testing: Comprehensive and through testing will be conducted in the staging environment to verify the application features which includes biometric authentication and payment processing are working as intended.
- Performance Testing : Load and stress testing will be carried out to make sure that the backend services are able to handle the expected traffic.
- Security Testing : Testing of privacy and security will be carried out which will focus on multi-biometric authentication system, data storage and payment processing.

d)Step 4:Production Deployment

- Backend Services: Flask backend services will be deployed to production AWS EC2 instance thereby ensuring that all configurations are suited for a live environment.
- Database Deployment: SQLite databases will be deployed and configured in the production environment. If the application scales in later stages, AWS RDS might be considered for enhanced database management.
- DNS and SSL configuration: DNS setting will be updated and the SSL certificates will be applied for securing communications between the frontend and the backend.
- Monitoring Setup: AWS CloudWatch will be used and set up to monitor the performance of the application and to send alerts in case of any issues that may occur.

e)Step 5:Post Deployment Verification

- Smoke Testing: Smoke tests will be conducted to make sure that the deployment was successful and all the critical functions are operational.
- User Acceptance Testing(UAT): A selected group of users will be engaged to validate the performance of the application in real world environment.

AV

Rollback Plan

a)Step 1: Monitoring and Detection

- Monitoring: AWS CloudWatch will be used to monitor the application performance continuously and its alerts will be configured in case of any critical issues.
- Feedback Collection: User feedback and bug reports from the app stores and other sources will be monitored actively.

b)Step 2: Issue Identification

- Log Analysis: If any issue occurs , to identify its root cause , application and server logs will be analyzed.
- Severity Assessment: Then the severity of the issue caused will be assessed to figure out if a rollback is needed.

c)Step 3: Rollback Execution

- Backend Rollback: If any issue related to backend services arise, then revert the Flask backend to the previous stable Docker image.
- Database Rollback: If any data issue arise , restore the SQLite database from the most recent backup.
- Mobile App Rollback: Stop the distribution of the version on the application with issues and provide users with a stable version or a patch.

d)Step 4:Post- Rollback Verification

- Testing: The application must be re-tested after rollback to ensure stability.
- User Communication: Users are to be notified about the rollback and updates about the resolution will be provided.

e)Step 5:Root Cause Analysis and Fix

- Resolution: The issue that caused the rollback is to be addressed and then the fixed version need to be re-deployed with the staging environment.

AV

Maintenance and Support

1. Support Model

Structure:

- Tier 1: Designed to handle basic user queries and common issues.
- Tier 2: It will manage the complex technical problems related to backend services and APIs.
- Tier 3: Here, the development team is responsible to resolve any critical issue that require code change or any update.

Responsibilities:

- Tier 1: Customer support will be needed for general enquiry assistance.
- Tier 2: Technical support will be needed here with expertise in the tech stack.
- Tier 3: Support from the developers maintaining the codebase and deploying the updates will be needed.

Channels:

- Email/Helpdesk, In-App Support, Live Chat: Accessible for user support and issue reporting from user side and resolution update from tech side.

2. Maintenance Plan

Routine Activities:

- Codebase Updates: Updates and security patches for the frontend and backend on regular basis.
- Database Maintenance: Optimize the SQLite database and ensure data integrity and security.
- Third-Party Integrations: Updates for (Stripe/PayPal) and AWS services on regular basis.

Scheduled Maintenance:

- Monthly: Minor updates and required security patches.
- Quaterly: Comprehensive code reviews.
- Annually: Full security audit and optimization

Downtime Management:

- Planned: It will happen during off-peak hours and users will be notified.
- Unplanned: Immediate response with real-time user updates.

3.**Issue Tracking**

System:

- JIRA/GitHub Issues: Track bugs, feature requests and maintain tasks.
- Automated Monitoring: AWS CloudWatch for real-time issue logging.

Process:

- Identification and Prioritization: Triage issues by severity.
- Resolution: Appropriate support tier is assigned for fixing and testing.
- Verification: Regression testing is carried out to confirm resolution.
- User Communication: Inform users of the resolution state through the support channels.