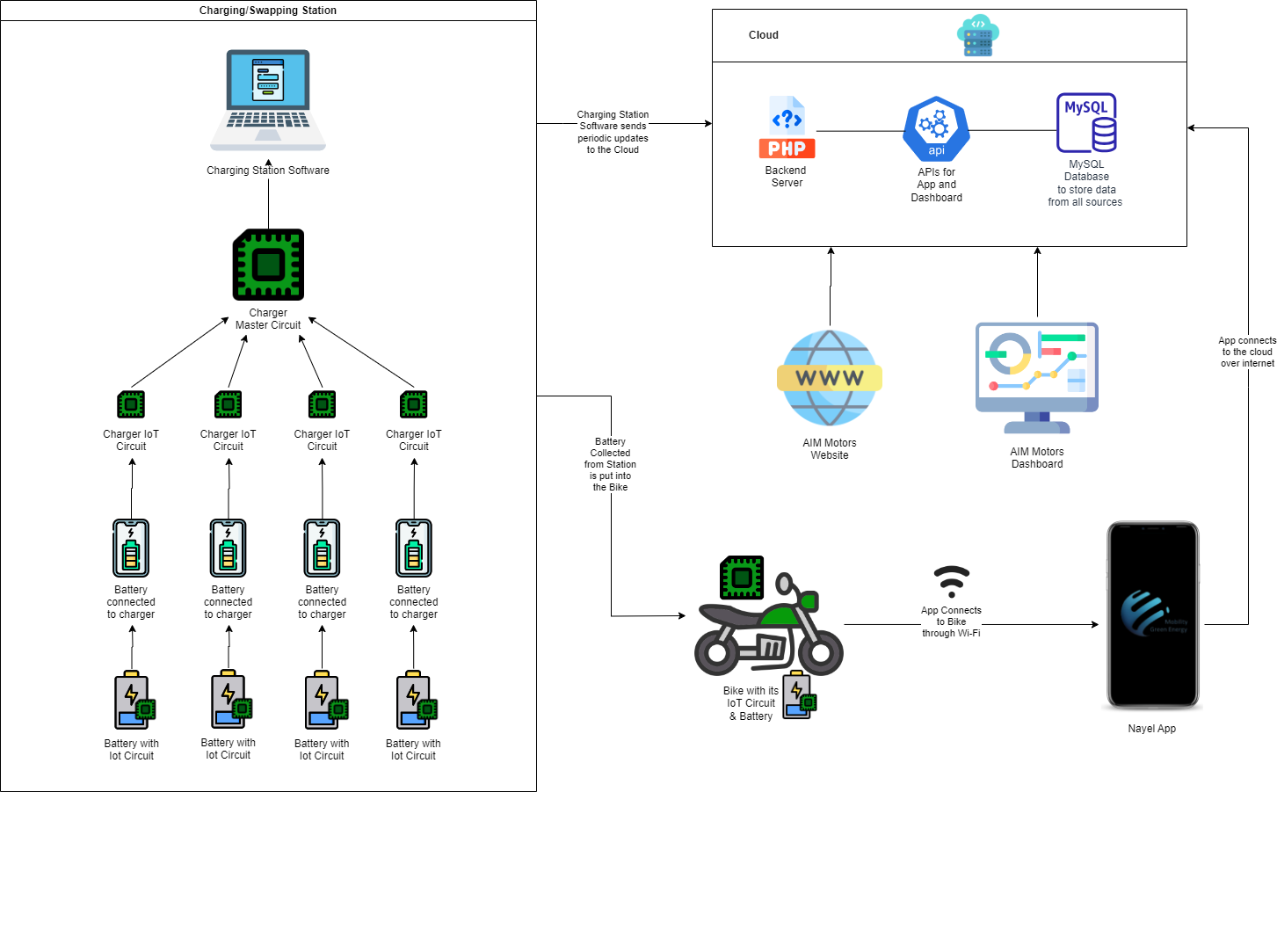
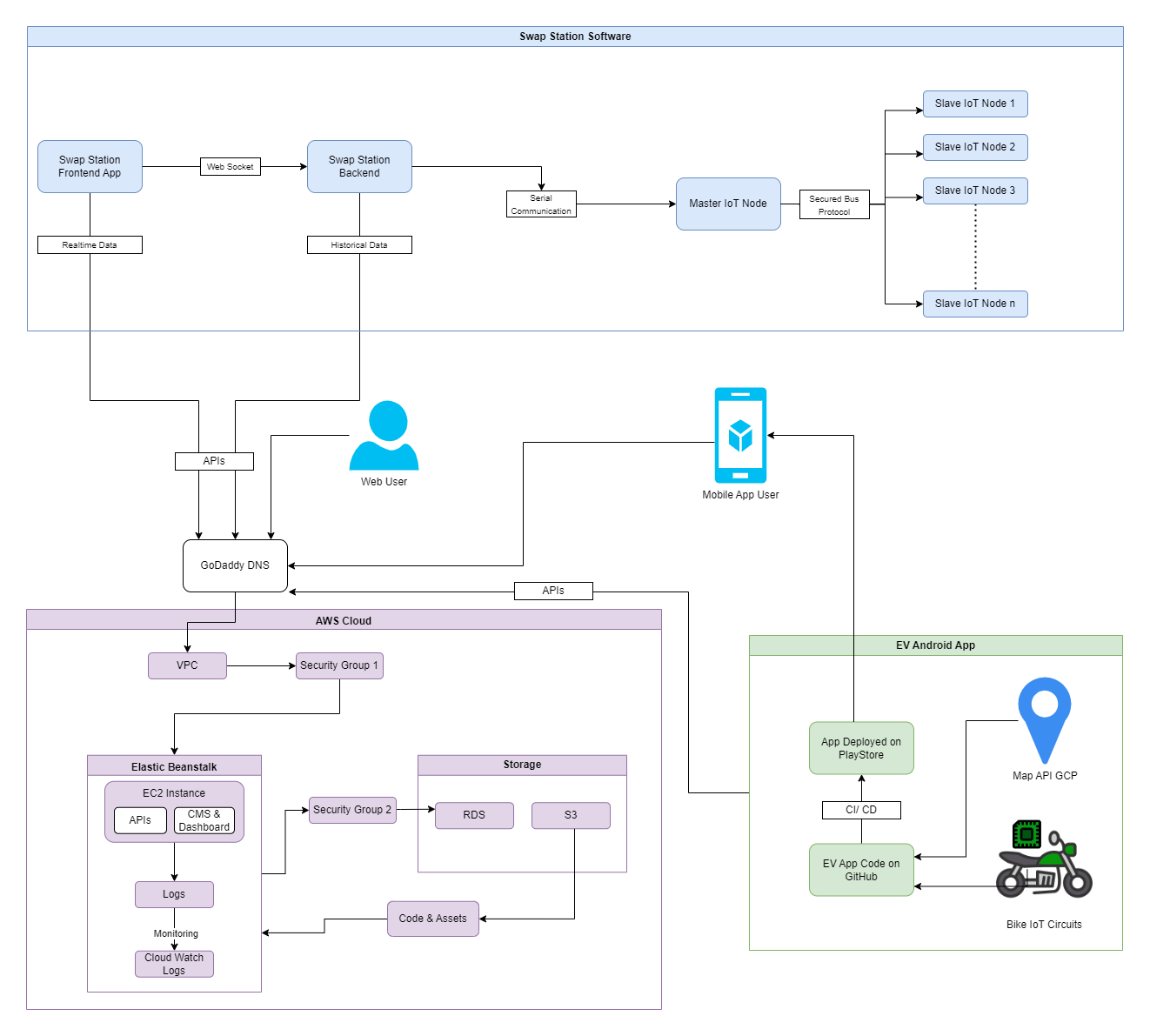
AIM Software Documentation

## High-Level Architecture Diagram:

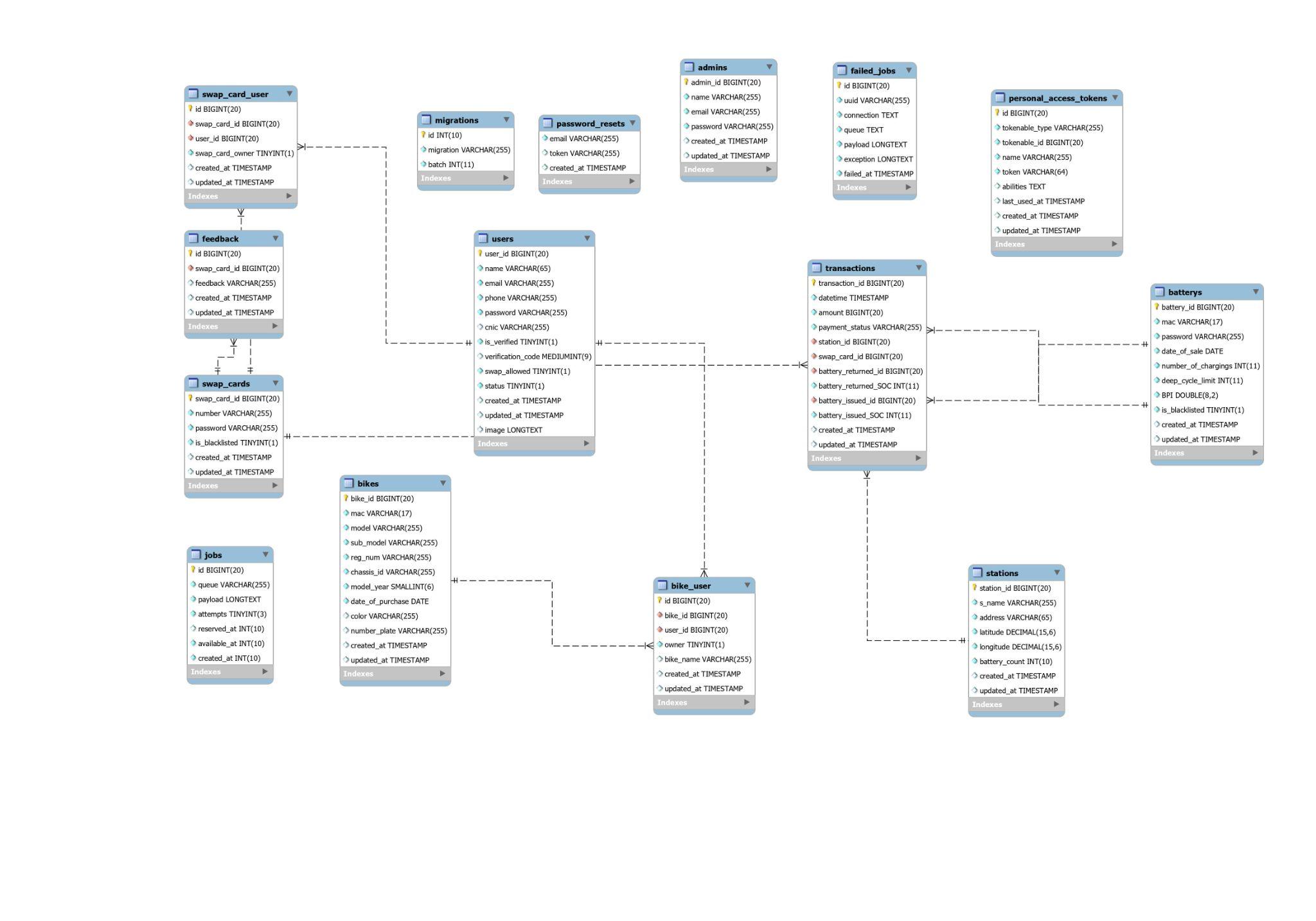
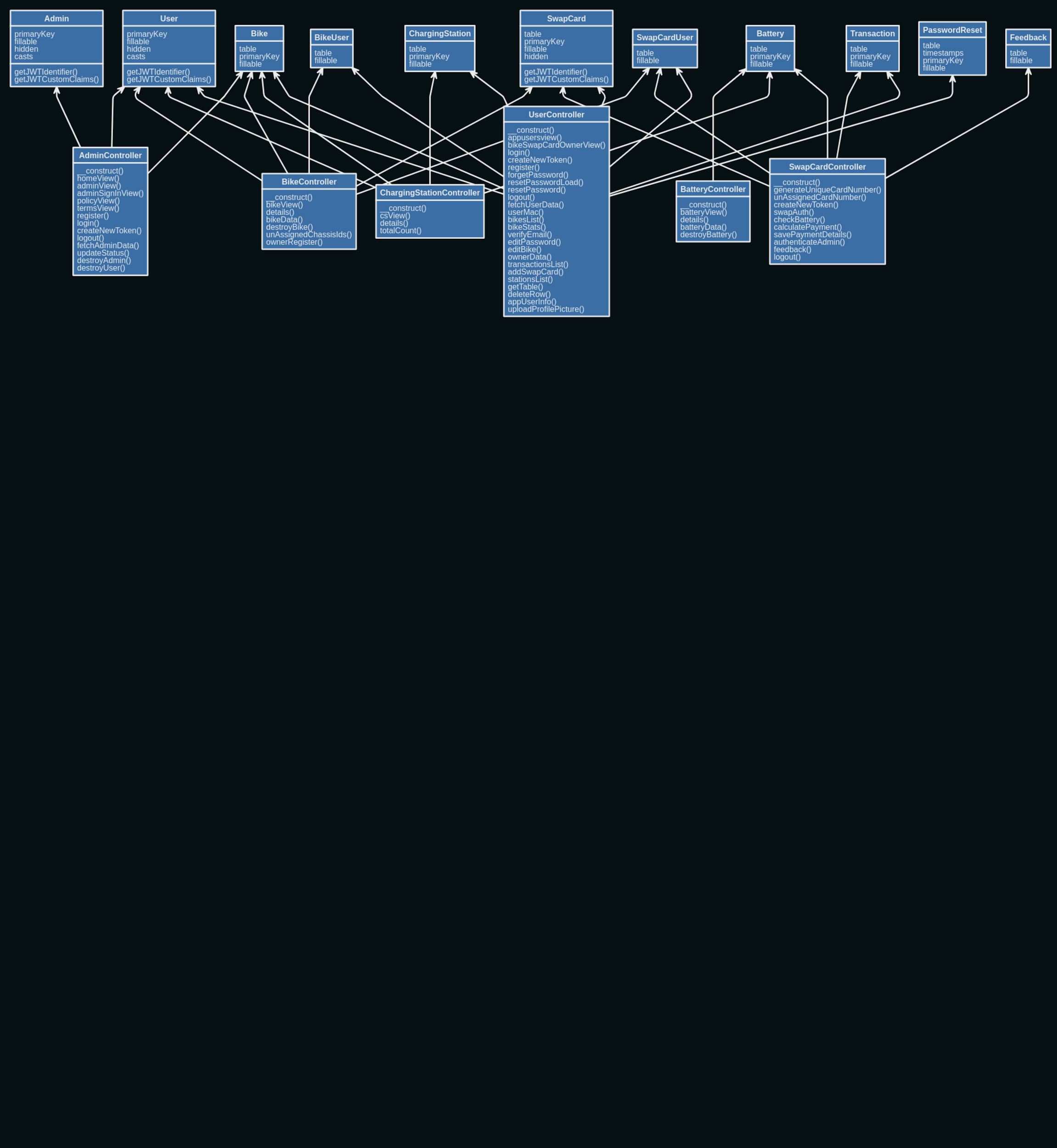
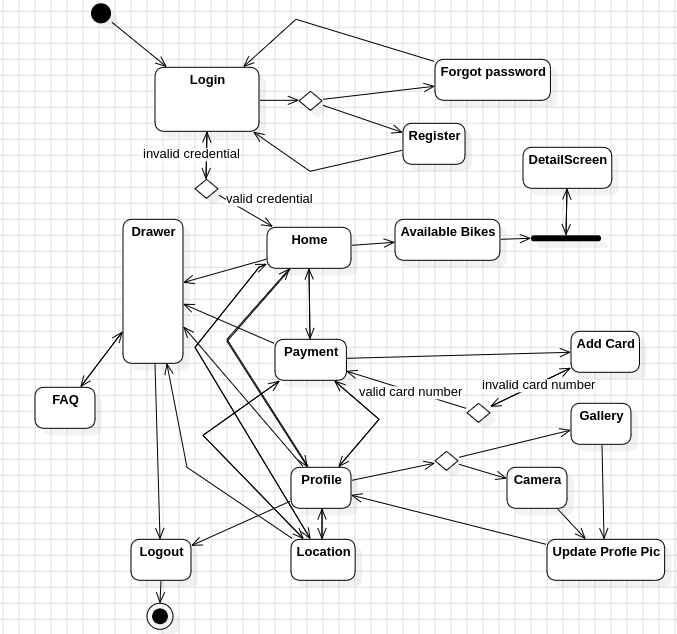
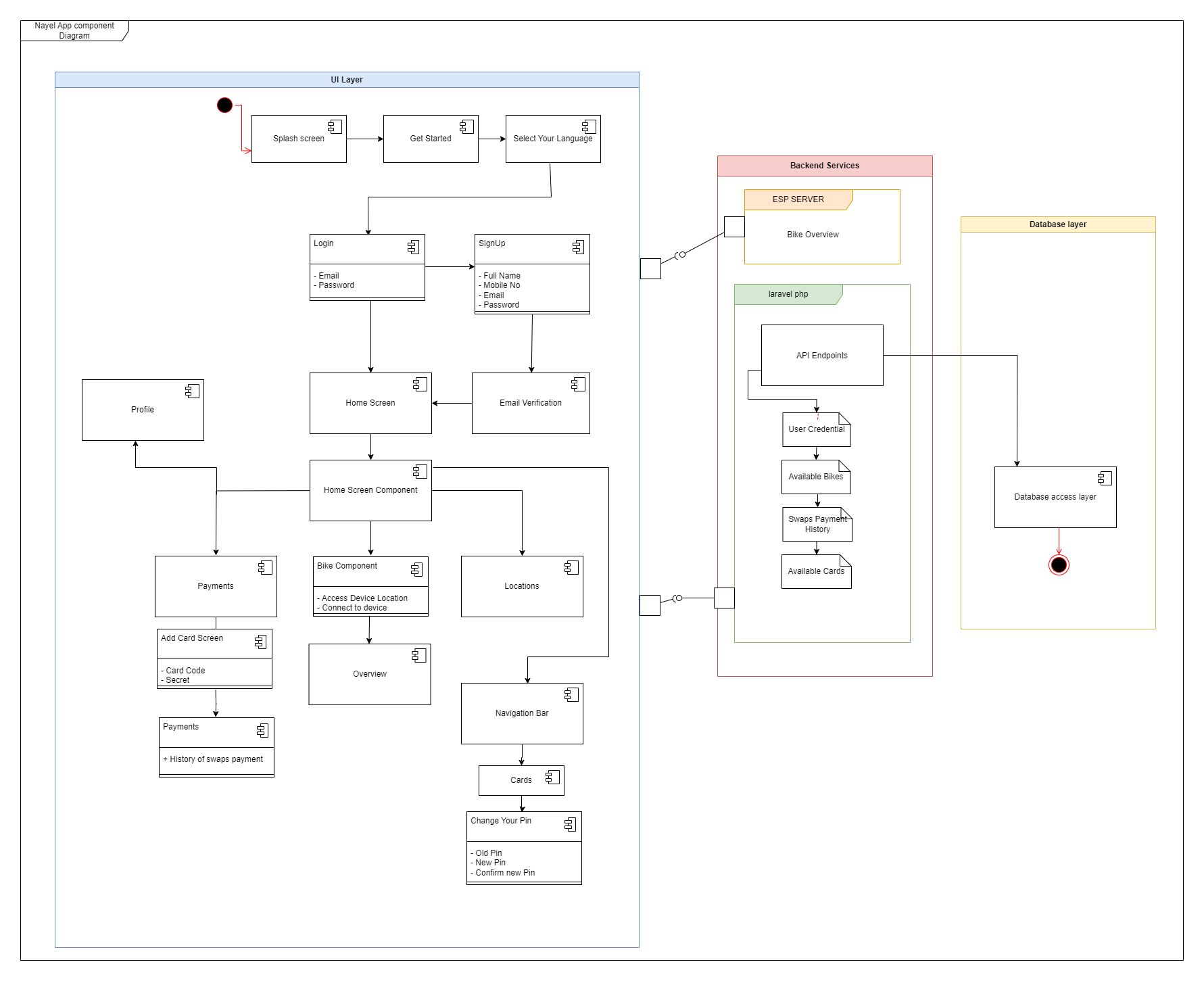
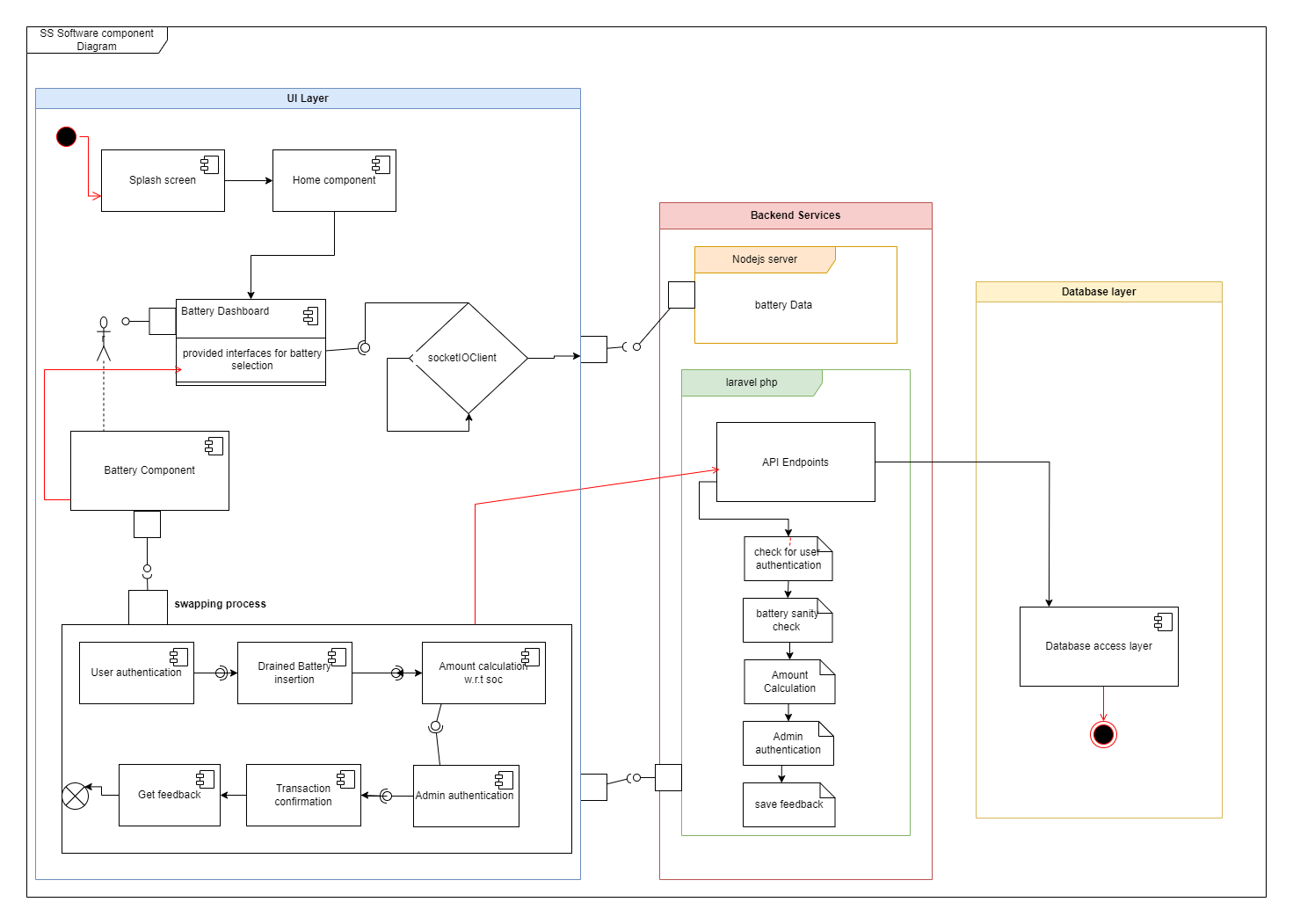
Following are the High- Level Architecture Diagrams:

* **System Diagram of all Software Components:**
* **High Level architecture Diagram:**

## 

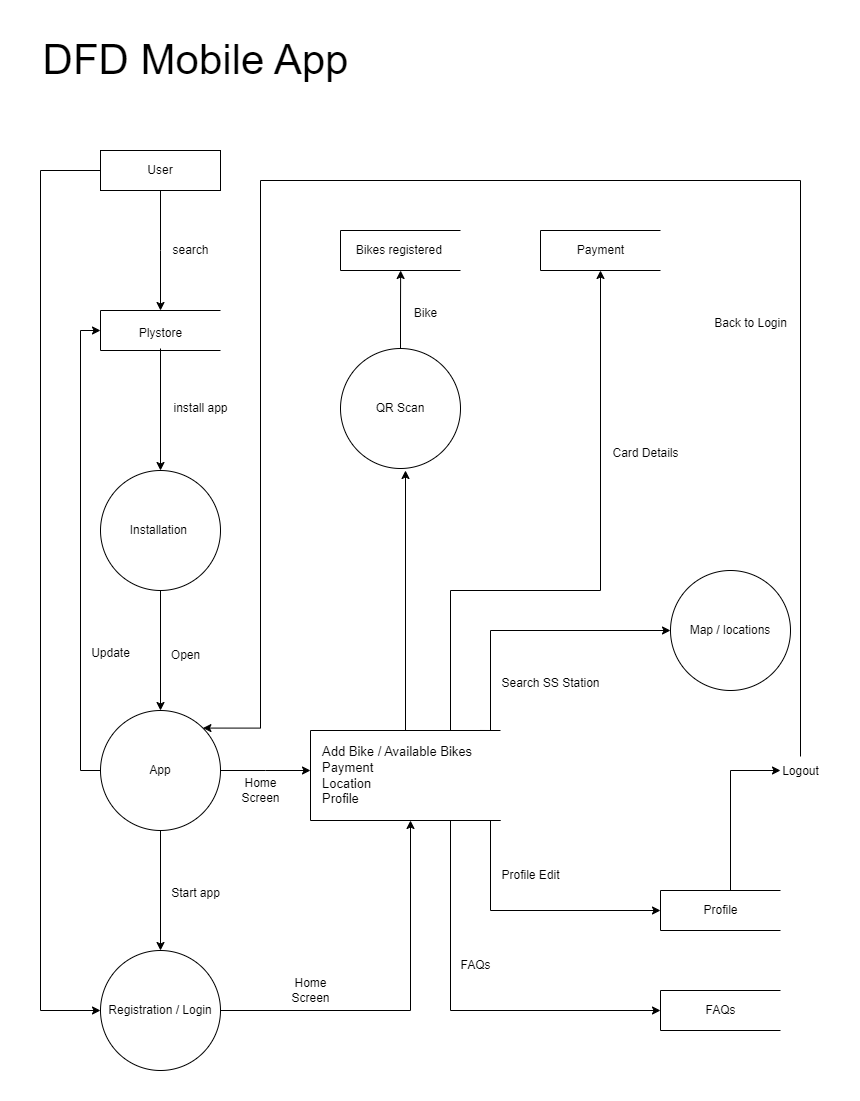
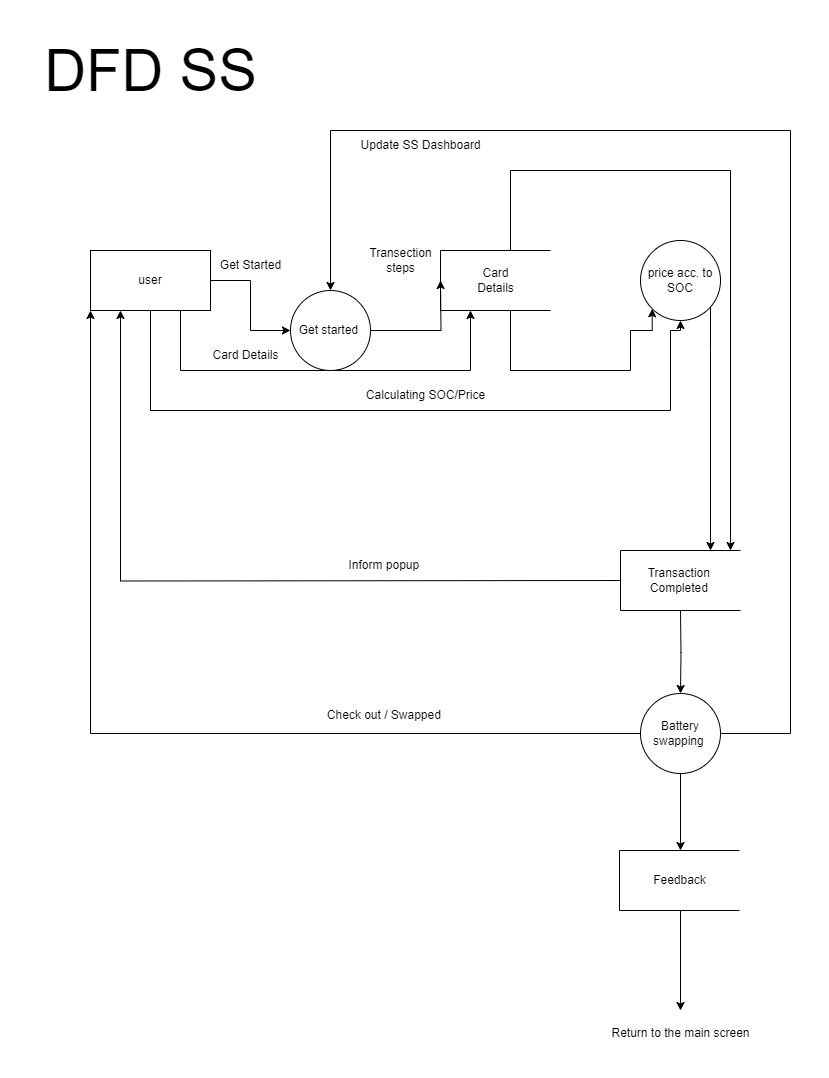
## Component Descriptions:

Following are the Component Description Diagrams:

* **Entity Relationship Diagram:**
* **Class Diagram of CMS, Dashboard & Backend APIs:**
* **Activity Diagram of Nayel App:**
* **Component Diagram of Nayel App:**
* **Component Diagram of Swapping Station Software:**

## 

## Data Flow and Interaction:

* **Data Flow Diagram of Nayel App:**
* **Data Flow Diagram of Swapping Station Software:**

## 

## API Documentation:

API Documentation can be found at the following link:

<https://aim-motors.com/docs>

To login use following credentials:

UserId: [admin@aim-motors.com](mailto:admin@aim-motors.com)

Password: Hello1234\*

## 

## Deployment and Infrastructure:

### Deployment Process:

Our deployment process follows a structured workflow ensuring code integrity and reliability. It encompasses several stages from code development to production deployment.

* **Code Development and Review:**
  + Developers work on features or bug fixes in feature branches.
  + Once development is complete, a pull request is created to merge the changes into the main branch.
  + Code undergoes thorough review by peers before merging to ensure quality and adherence to standards.
* **Main Branch Deployment:**
  + Upon approval, code changes are merged into the main branch.
  + The main branch serves as the source of truth for the production-ready codebase.
* **Automated Deployment with GitHub Actions:**
  + GitHub Actions is utilized for automated deployment.
  + When changes are pushed to the production branch, GitHub Actions triggers deployment to the production environment.
* **Deployment Logs:**
  + Deployment logs are available on AWS Elastic Beanstalk.
  + Detailed logs provide visibility into the deployment process, facilitating troubleshooting and monitoring.
* **App Deployment to Play Store:**
  + Similarly, the mobile application follows a similar deployment process.
  + Changes are pushed to the main branch, which are merged in the production branch triggering deployment to the Google Play Store.

### Cloud Resources Utilized:

* **AWS Elastic Beanstalk:**
  + Used for deploying and managing web applications at scale.
  + Provides automatic scaling, load balancing, and application health monitoring.
* **Amazon EC2:**
  + Virtual servers hosting our application and supporting services.
  + Allows for flexible scaling and customization of computing resources.
* **Amazon S3:**
  + Object storage service utilized for storing and retrieving static assets e.g. code different versions in the form of zip files.
* **Amazon RDS (Relational Database Service):**
  + Managed database service for hosting relational databases like MySQL.
  + Provides high availability, automatic backups, and scaling capabilities.
* **AWS Security Groups and VPC (Virtual Private Cloud):**
  + Security groups define firewall rules controlling traffic to and from EC2 instances.
  + VPC enables isolation and secure connectivity within the AWS cloud environment.
* **GoDaddy DNS Service:**
  + Utilized for managing domain names and DNS records, facilitating access to our web application.
* **Google Maps Service (from Google Cloud Platform):**
  + Integrated for location-based services and mapping functionalities within our application.
* **Google Play Store:**
  + Platform for distributing our Android application to users.
  + Provides a streamlined process for app deployment, updates, and management.

## Security Considerations:

**Two main mechanisms for authentication are used:**

* Session-based Authentication
* Token-based Authentication (API Authentication)

**For authorization, following laravel mechanisms are used:**

* Middleware
* JWT (JSON Web Tokens)

**Encryption in Transit is applied through:**

* HTTPS (SSL/TLS)
* Encryption Libraries of Laravel

**General Encryption Details are as follows:**

* Encryption Configuration in Laravel .env file
* Hash passwords stored in MySQL
* Secure Sessions for CMS and Dashboard

## 

## Monitoring and Logging:

For Monitoring and Logging following services are used:

**Monitoring and Logging in AWS**

* AWS CloudWatch

**Monitoring and Logging in Elastic Beanstalk:**

* Elastic Beanstalk Logs
* Elastic Beanstalk Monitoring

**Play Store Deployment Logs:**

* Deployment Status
* Version Information
* Deployment Timestamps
* Errors and Warnings
* Distribution Details

## Scalability and Resilience:

* Not Applicable

## Version Control and Change Management:

**Main Branch (main)**

* + Represents the stable production-ready codebase.
  + All feature branches are merged into this branch after review and approval.

**Feature Branches**

* + Created from the main branch for implementing new features or bug fixes.
  + Developers work on isolated changes within these branches.
  + Named descriptively (e.g., feature/add-authentication, bugfix/fix-payment-gateway).

**Release Branches**

* + Branched off from the main branch to prepare for a release.
  + Used for final testing, bug fixing, and preparing release notes.
  + Named based on the release version (e.g., release/1.0.0).

**Tracking Changes:**

* **Version Control:** Changes are tracked using a version control system like Git.
* **Commits:** Developers make changes in their local environment and commit them with descriptive messages.
* **Branches:** Changes are isolated in feature branches for development, ensuring separation from the main codebase until they're ready.

**Reviewing Changes:**

* **Pull Requests (PRs):** Developers submit PRs to merge their feature branches into the main branch.
* **Code Review:** Peers review the code changes in PRs, providing feedback on code quality, best practices, and potential issues.
* **Approval Process:** PRs require approval from designated reviewers before merging.

**Deploying Changes:**

* **Deployment Automation:** Automated deployment pipelines (CI/CD) handle the deployment process.
* **Environment Setup:** Separate environments (e.g., development, staging, production) are set up for testing and deployment.
* **Continuous Integration (CI):** CI pipelines automatically build, test, and deploy changes after they pass review and approval.
* **Rollback Mechanism:** A rollback plan is in place to revert changes in case of deployment failures or issues.
* **Monitoring:** Monitoring systems track deployment metrics, performance, and errors to ensure successful deployments.
* **Manual Verification:** Critical changes undergo manual verification in the deployed environment to ensure functionality and performance.
* **Post-Deployment Tasks:** Tasks like database migrations and cache clearing are performed post-deployment to maintain application integrity.

## Collaboration Tools:

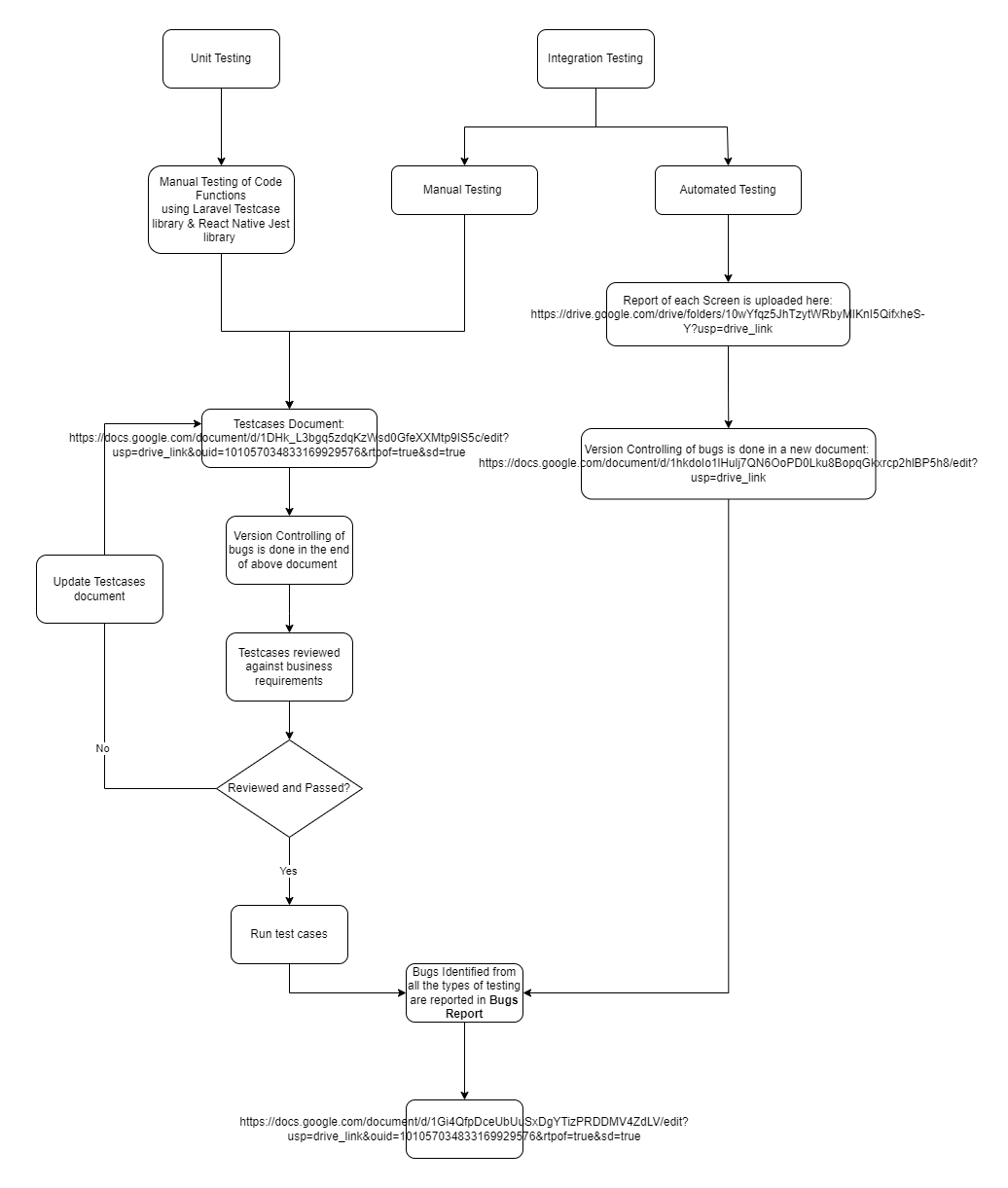
**Collaboration Tools Used by the Team**

* GitLab/GitHub
* Jira
* Google Meet
* Google Drive

## V&V Examples and Tutorials:

**Test Plan**

* Objective
* Scope
* Approach
* Test Types
* Tools
* Reporting

**Testing Cycle:**

**Integration Process**

* Branching Strategy
* Code Review
* Testing Environment
* Deployment Strategy
* CI/CD Pipelines

**Setting Up Development Environment**

* Cloning Repository
* Installing Dependencies
* Environment Configuration
* Generating Application Key
* Running Migrations and Seeding

# Clone the repository

git clone <repository\_url>

# Install dependencies

composer install

npm install

# Copy environment configuration

cp .env.example .env

# Generate application key

php artisan key:generate

# Run migrations and seed database

php artisan migrate --seed

**Running the Application Locally**

* Starting Development Server
* Accessing the Application in Browser

# Start the development server

php artisan serve

# Access the application in your browser

http://localhost:8000

**Common Use Cases and Workflows (PHP Laravel)**

* User Authentication
* CRUD Operations
* Form Validation
* API Development
* Middleware Usage
* Scheduled Tasks
* Queues for emails