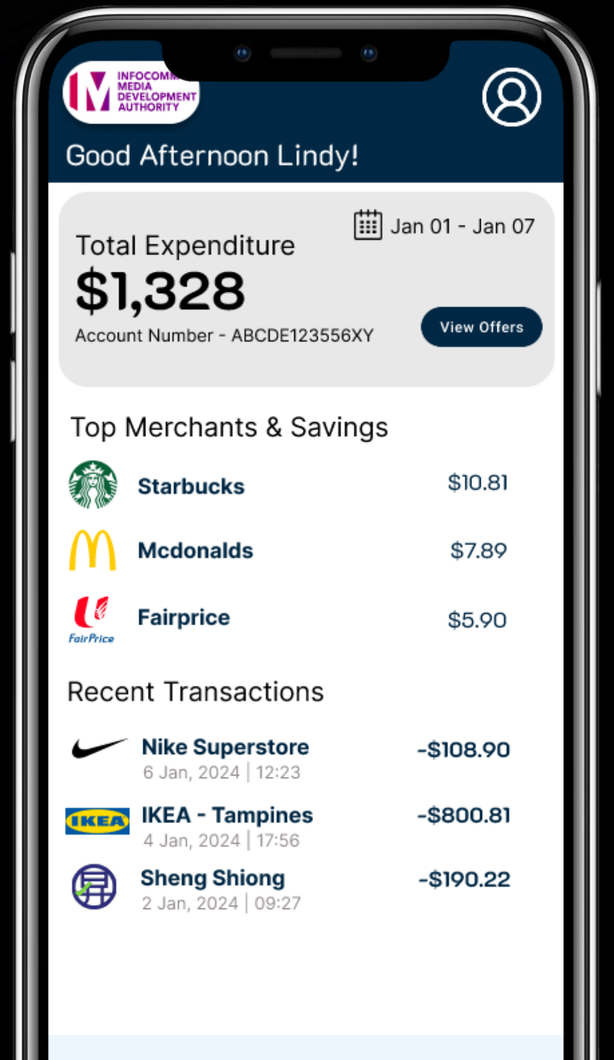
sigma tech

Sigma Tech

JAMISON (SMU) JINUK(NUS) SIVA(NUS) GERALD(NUS) SHOBHIT(NUS)



Problem statement

To create a sample app that **stores payment transactions** on a **user's personal data pod** which reads data and **recommends personalised merchant offers**.

VALUE PROPOSITION

Customer profile

B2C Businesses

Gain

Usage of data from PODS for insights into consumer's preference

Usage of PODS

Users

Concerns about data security

Pain

Value proposition

Gain creators

Using ML methods, personalize product recommendation for customers

sigma tech

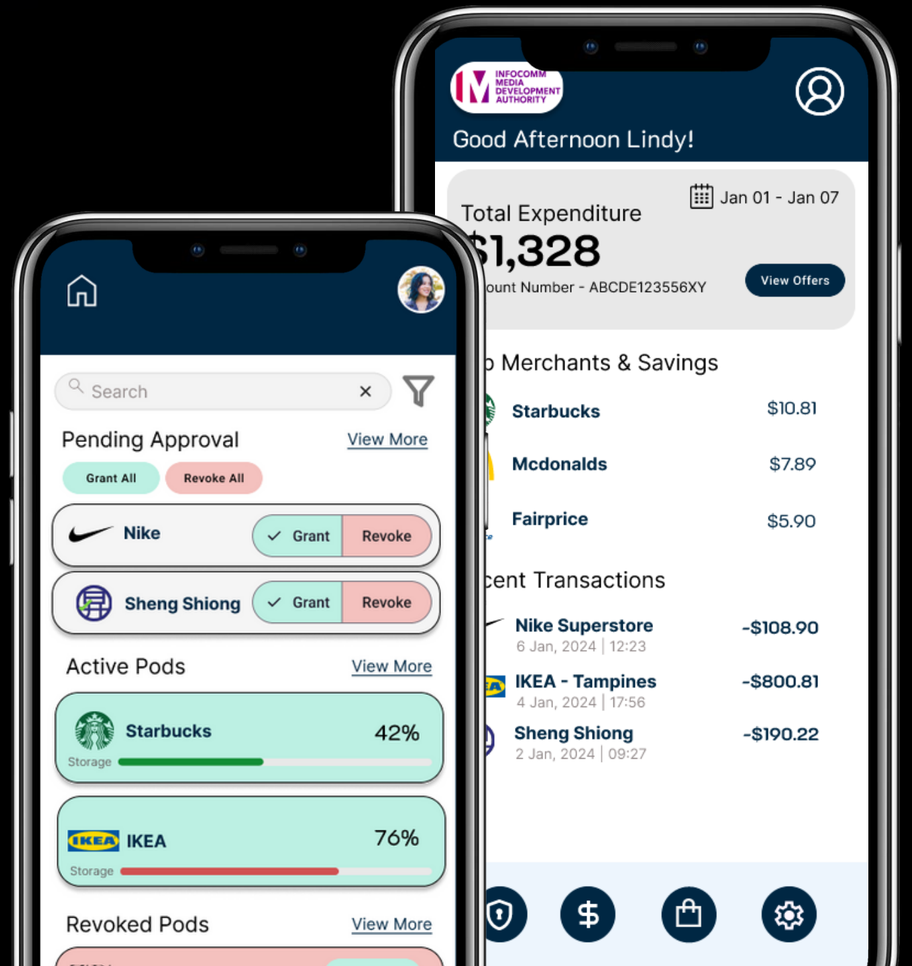
Usage of common cybersecurity measures to ensure data security

Products and services

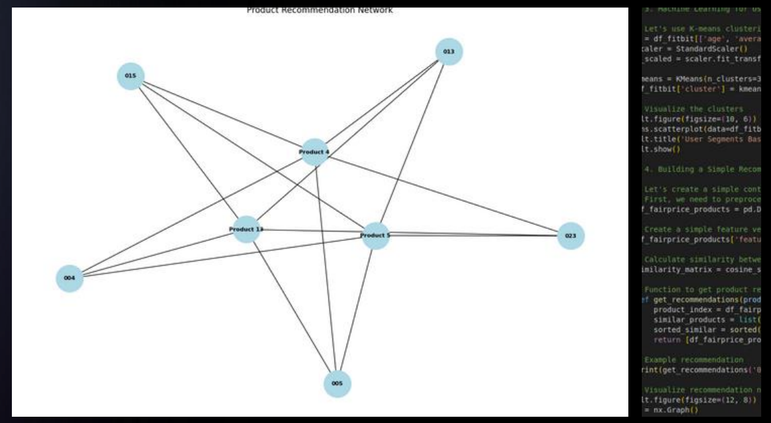
Pain relievers

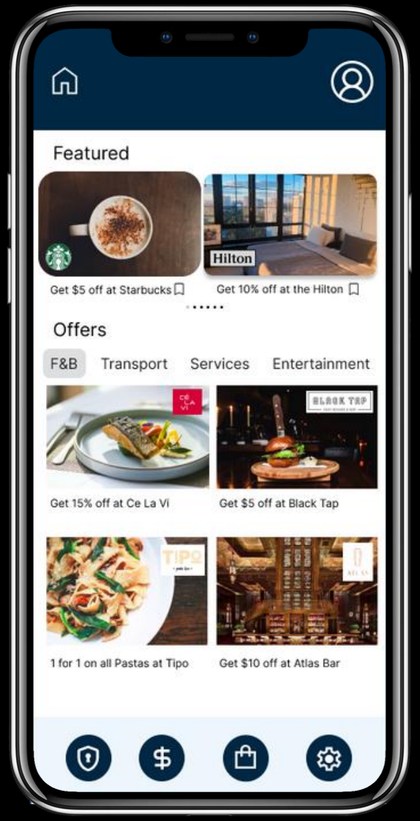
Sigma Integrated Pod

Clean and intuitive user interface
Easy to read data visualization for users and firms
Personalized recommendations based on lifestyle



Sigma Integrated Pod

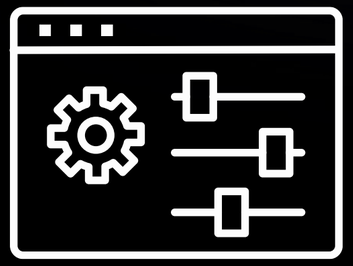




Full frontend and backend integration:

Backend:
Anonymised, aggregated user data gives merchants insight into their users' spending habits. This increases recommendation accuracy

Frontend:
Users get personalized recommendations and offers from stores they like and have previously purchased from



Pod Management

Your Data, Your Choice
Users can choose what type of data from each service goes in

Changed Your Mind?
Revoke access in an instant

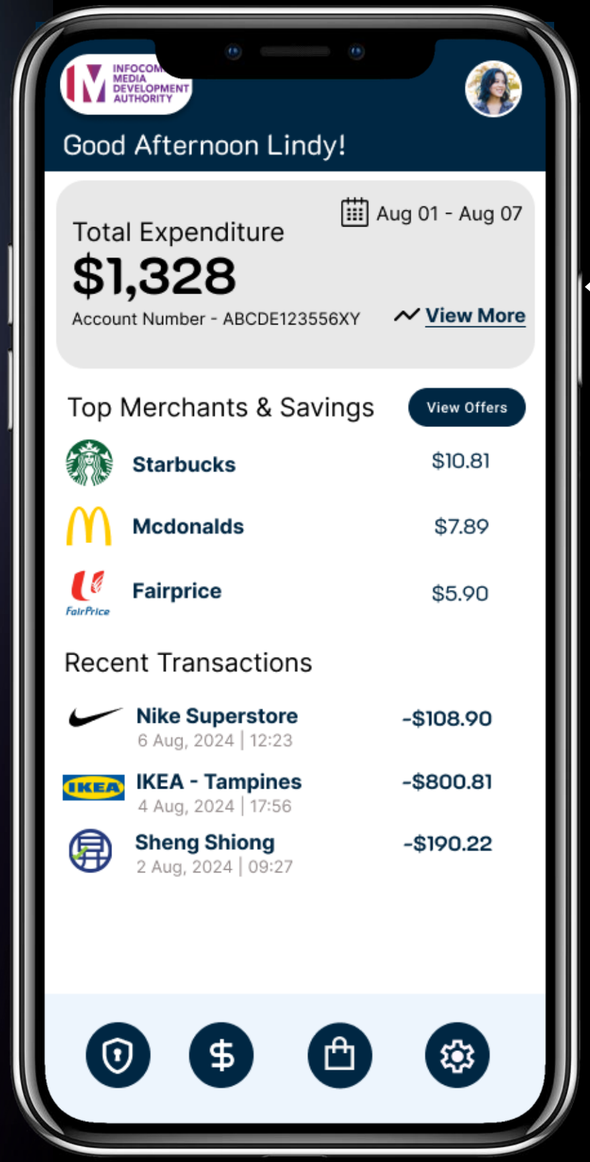
Spending Analysis

Understand Yourself
Users can view their data at a glance, and gain insight on expenditures

Better Recommendations

Your Preference, Our Priority
Backend identifies personalised offers users will enjoy and use

Dashboard



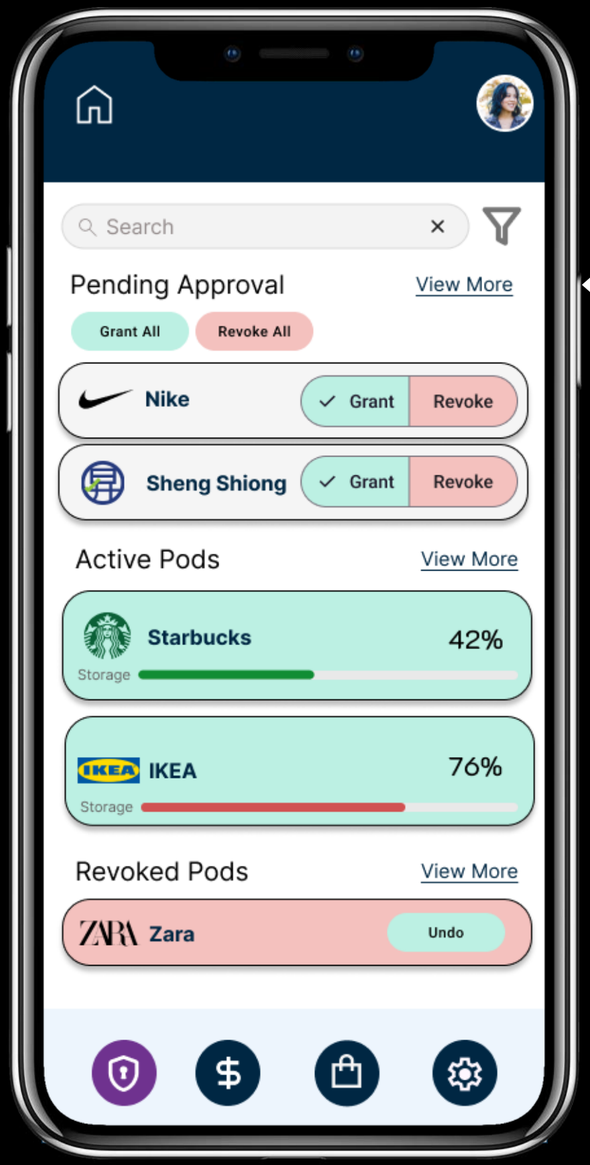
Understand Yourself

Users can see how much they've spent, as well as their favourite stores

Break it Down

Users can get a detailed view of their expenditure across all services

Pod Management

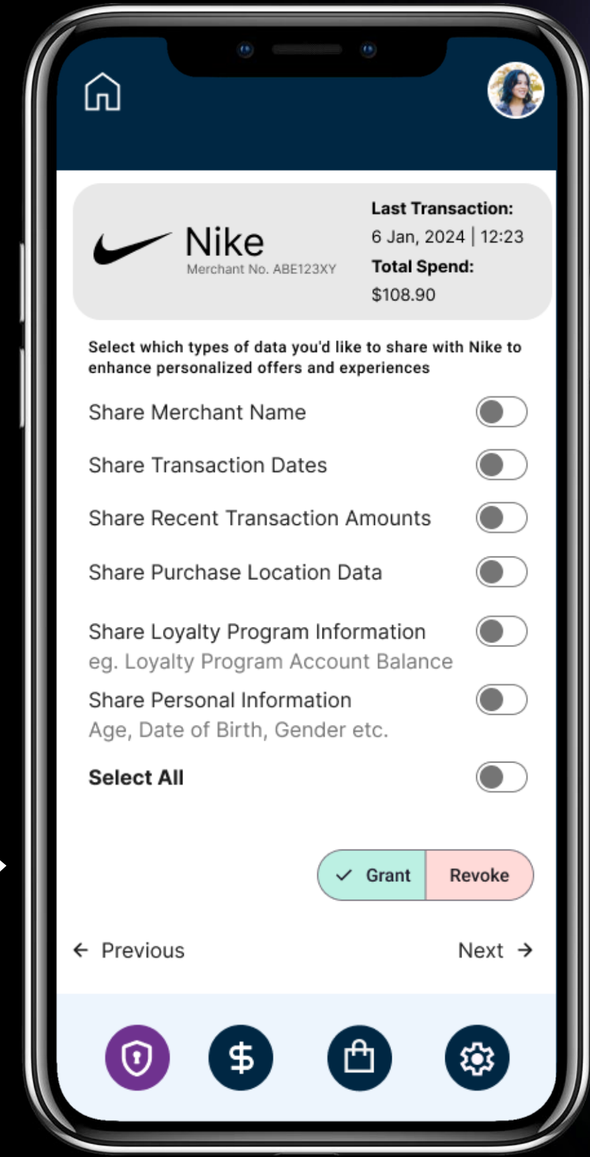


Streamlined Consent

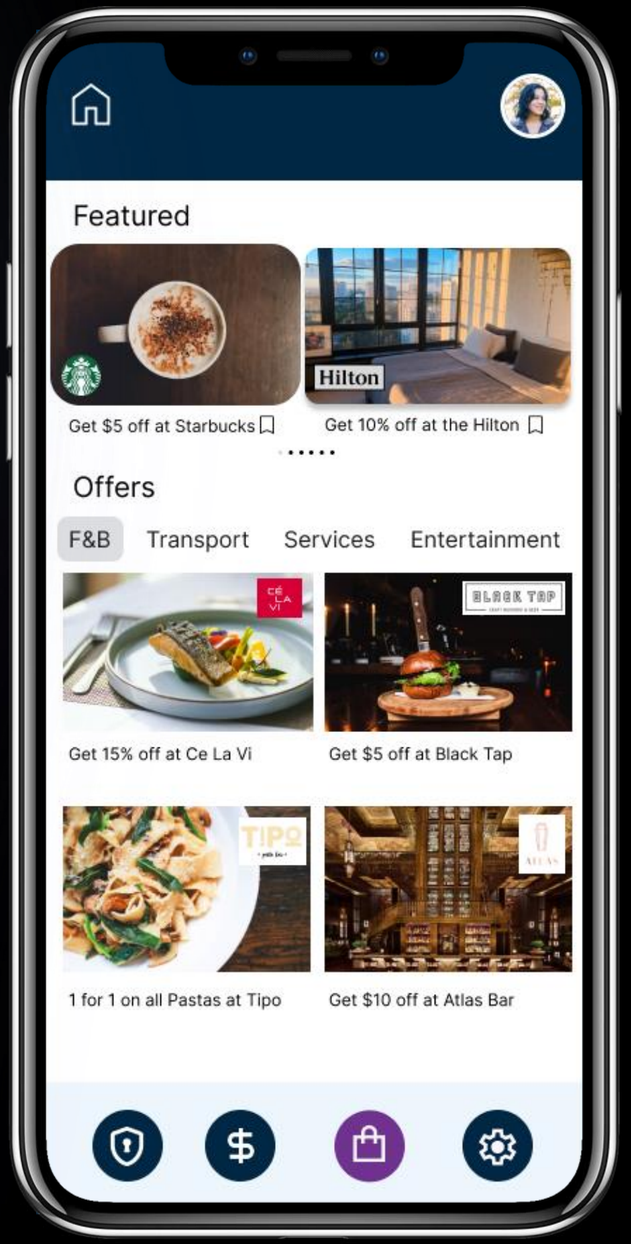
At a glance, users can view the data being shared from each service, and grant / revoke access in a heartbeat

Your Data, Your Rules

Users control what, how much, and when data is shared



Offers Page



Make Your Data Work For You

Users can get recommendations and offers for their favourite stores based on their purchase history

Organisation is Key

Offers are neatly separated by category for users' convenience

App Live Demo

Architecture

Customer data

Pre-processing

Machine learning

Recommendation to customer

Backend live demo

Google Colab Demo

Cybersecurity measures

1

RBAC & DID Hybridization

Leverage the strengths of both systems for stronger protection.

2

Tokenisation

Replaces sensitive data with a unique token, making it difficult for unauthorized parties to understand or misuse.

3

Hashing/ SQL Injection Prevention

Implement query parameterization
Use ORM (Object-Relational Mapping) tools

4

Secure Token Handling for Third-Party Services

5

Load Balancing and Rate Limiting

6

Database Isolation and Transaction Management

Feature	RBAC (Role-Based Access Control)	DID (Decentralized Identity)	Differences	Similarities
Purpose	Manages access control based on roles.	Provides a unique, verifiable, and decentralized identifier for individuals.	RBAC focuses on controlling access, while DID focuses on identity management.	Both aim to improve security and control access.
Hierarchy	Hierarchical structure with roles and permissions.	Decentralized and self-sovereign.	RBAC has a defined hierarchy, while DID is more flexible and user-controlled.	Both involve assigning permissions or access rights.
Identity Management	Relies on traditional user accounts and credentials.	Uses DIDs as unique identifiers.	RBAC uses centralized identity management, while DID uses decentralized identity.	Both involve managing user identities.
Access Control	Grants access based on roles and permissions.	Can be integrated with RBAC to provide more granular control.	RBAC is the primary mechanism for access control, while DID can enhance it.	Both aim to control who can access what resources.

Feature	RBAC (Role-Based Access Control)	DID (Decentralized Identity)	Differences	Similarities
Flexibility	Can be customized to fit different organizational needs.	More flexible due to its decentralized nature.	RBAC can be adapted, but DID offers more dynamic possibilities.	Both can be adapted to various scenarios.
Privacy	Relies on traditional security measures.	Provides greater privacy by giving users control over their identity data.	RBAC may have limitations in privacy, while DID offers enhanced privacy.	Both aim to protect sensitive information.
Security	Depends on the strength of underlying security measures.	Can improve security by reducing reliance on centralized systems.	RBAC's security depends on traditional methods, while DID offers a more decentralized approach.	Both aim to prevent unauthorized access.

DID and Security:

- **Reduced Attack Surface: DID can reduce the attack surface by eliminating the need for centralized identity databases. This makes it more difficult for attackers to target and compromise identity information.**
- **Resilience: DID systems are generally more resilient to attacks, as there is no single point of failure. If one component of the system is compromised, the overall system may still be able to function.**

1 Modify DID Generation:

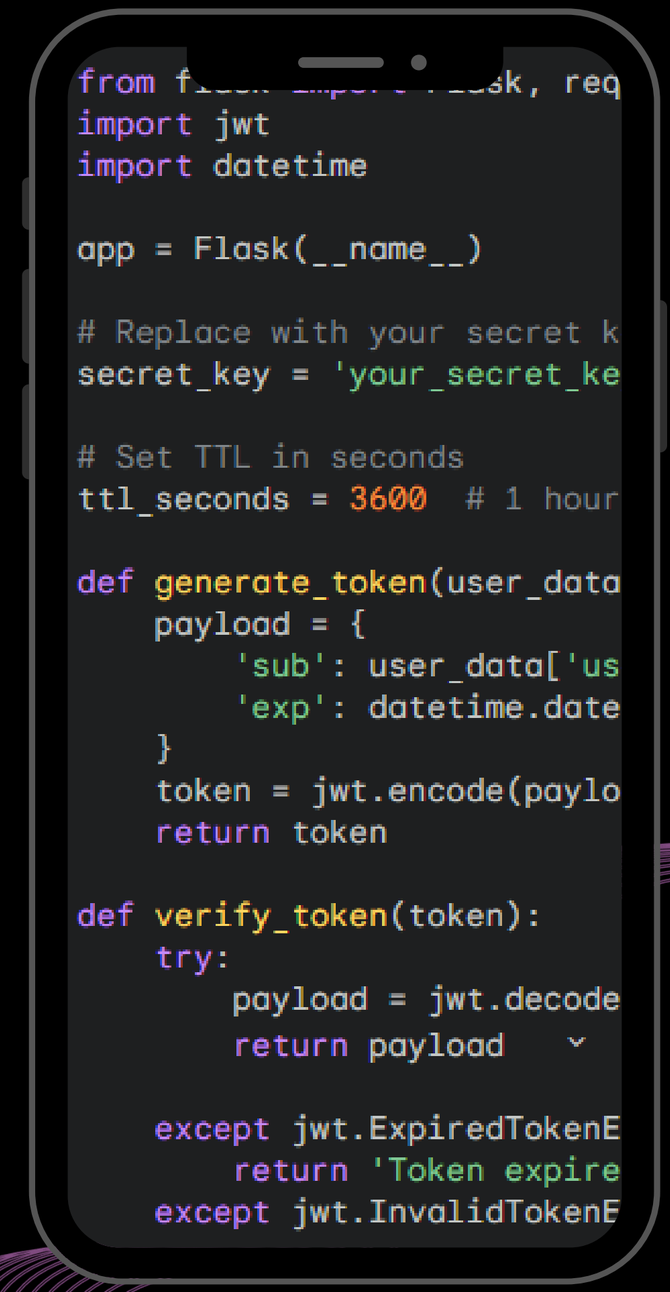
- Include a jwt claim in the DID document, containing the generated JWT token.
- Set the exp claim in the JWT to the desired TTL value.

2 Verify JWT on Access Control:

- Before granting access to a resource, verify the JWT token included in the DID document.
- Check the exp claim to ensure the token hasn't expired.

3 Implement Token Refresh:

- Add a mechanism to allow users to refresh their JWT tokens without having to re-authenticate.
- When a token is nearing expiration, the user can request a new token using their DID.
- The server can verify the user's DID and generate a new JWT with a refreshed expiration time.



1

DID Integration

- The DID document includes the JWT token.

2

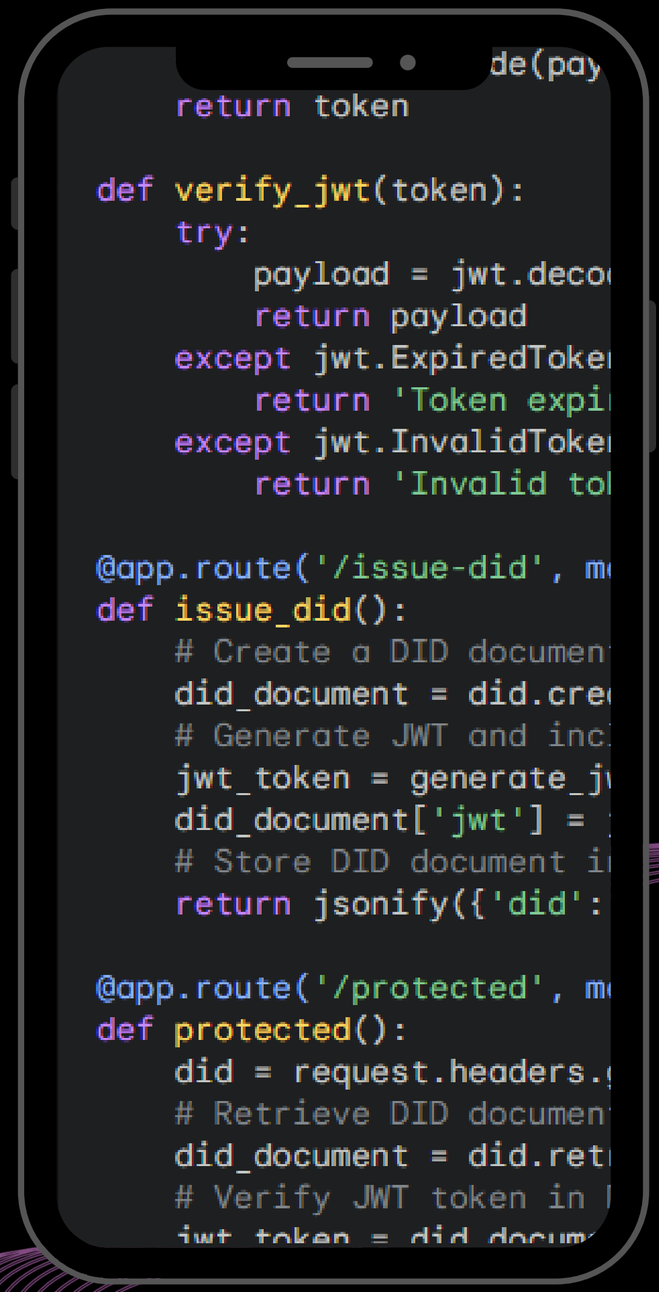
JWT Verification and Token Refresh

- JWT verification is performed before granting access
- Implement a mechanism to refresh JWT tokens as needed.

3

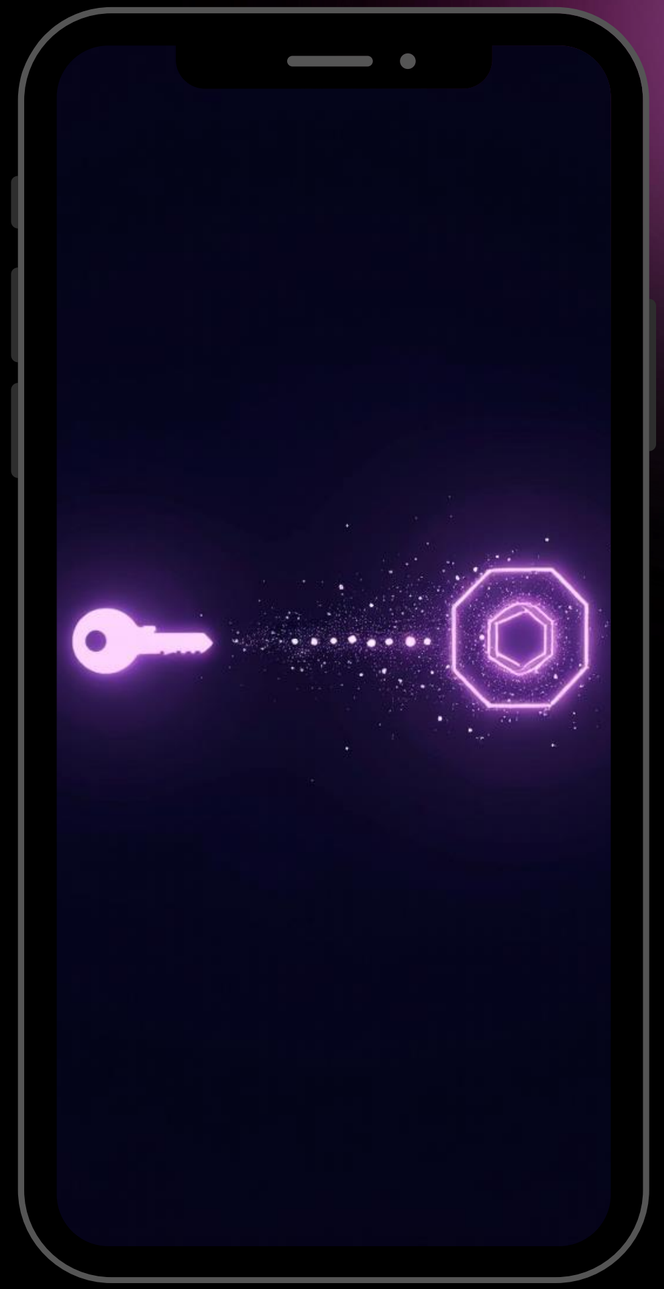
RBAC Integration

- Integrate RBAC rules to control access based on the DID and associated roles.



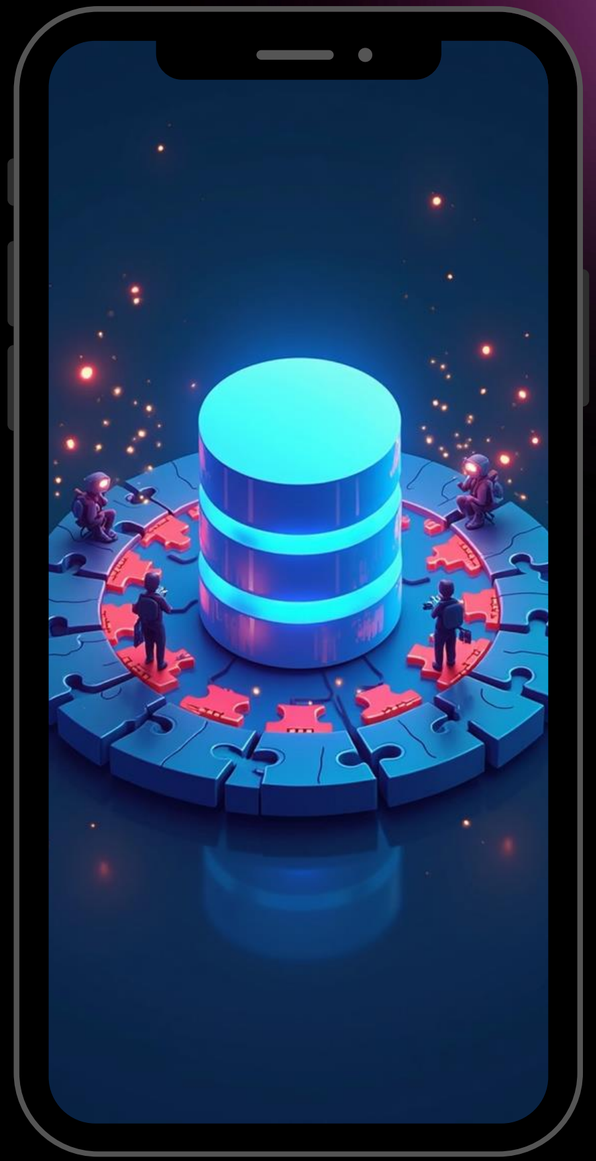
HASHING

Feature	Benefits
Data Encryption	Protects sensitive data from unauthorized access, ensuring confidentiality.
Salted Hashing	Makes it extremely difficult to recover original passwords, even if the database is compromised. - instead of passwords; maybe the most key information



SQL INJECTION PREVENTION

Feature	Benefits
Query Parameterization	Prevents malicious code from being injected into SQL queries, protecting the database from unauthorized access.
ORM Tools	Simplify database interactions and reduce the risk of SQL injection vulnerabilities.

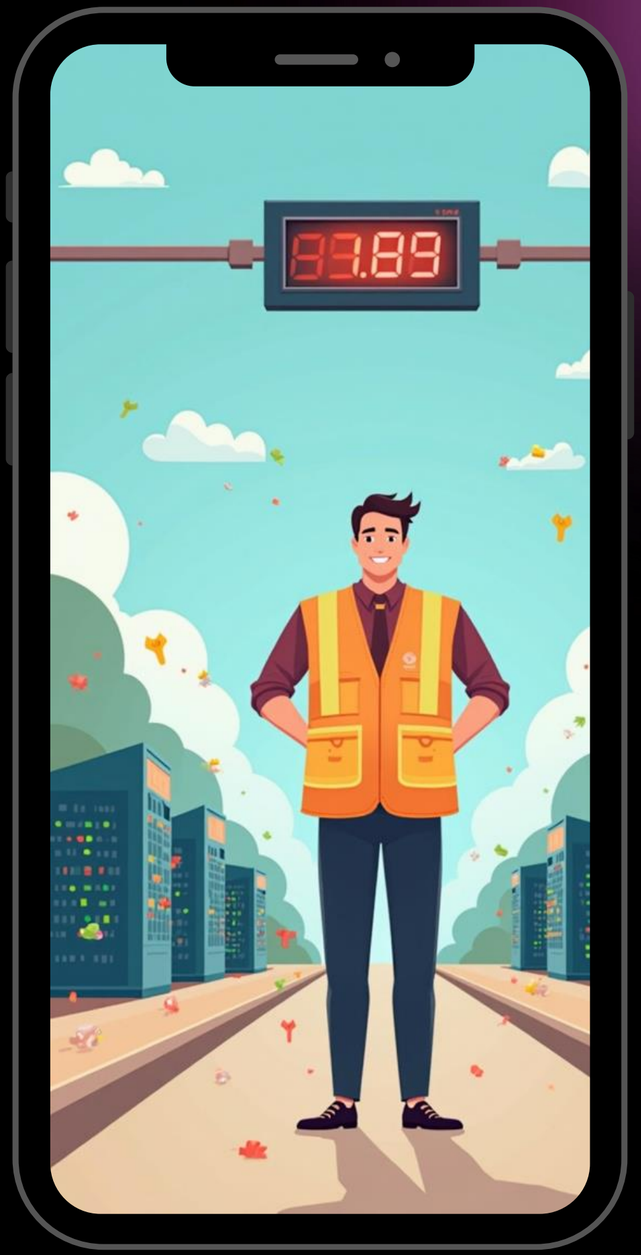


SECURE TOKEN HANDLING

Feature	Benefits
OAuth 2.0	Provides a standardized framework for authorization, allowing users to grant access to their data without sharing their credentials.
Request Origin Validation	Prevents unauthorized applications from accessing user data.

LOAD BALANCING & RATE LIMITING

Feature	Benefits
Load Balancing	Distributes traffic across multiple servers, improving performance and availability.
Rate Limiting	Prevents abuse and protects against DDoS attacks by limiting the number of requests that can be made within a specific time period.



DATABASE ISOLATION AND TRANSACTION MANAGEMENT

Feature	Benefits
Isolation Levels	Control how transactions interact with each other, preventing data inconsistencies.
Atomicity	Ensures that transactions are either fully committed or fully rolled back, maintaining data integrity.

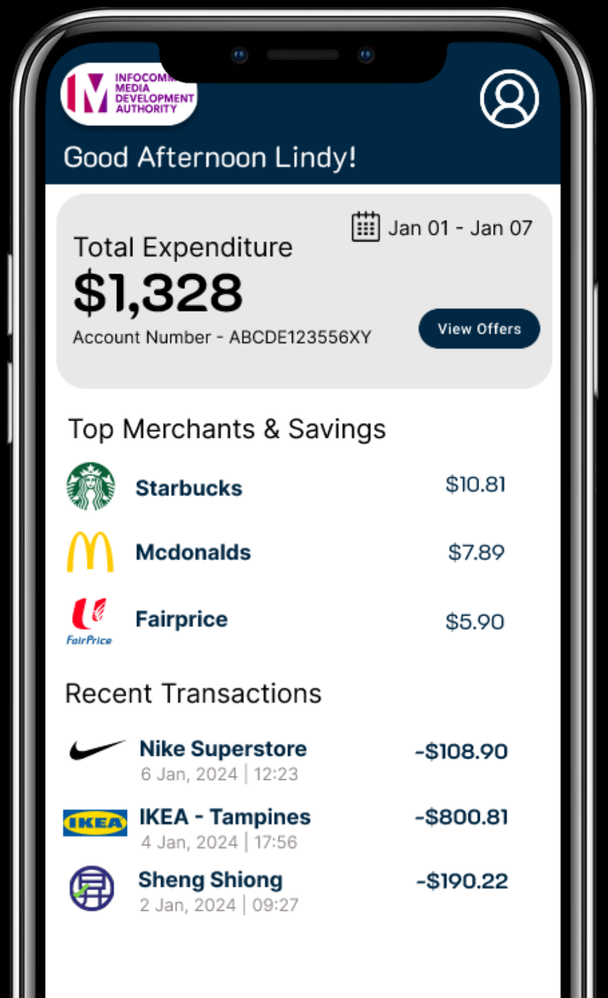
DATABASE ISOLATION LEVELS

1. Database Isolation Levels:

These prevent issues like dirty reads, non-repeatable reads, and phantom reads.
For example, the "SERIALIZABLE" isolation level provides the highest consistency but at the cost of performance.
Choosing the right isolation level is crucial to balance data consistency and system performance.

2. Transaction Atomicity:

This ensures that a transaction is treated as a single, indivisible unit of work.
It's part of the ACID properties (Atomicity, Consistency, Isolation, Durability).
If any part of a transaction fails, the entire transaction is rolled back, maintaining data integrity.



SigmaTech

Thank You