Don'sPay Requirements Document

Group 1: Anshul Abrol, Vijayagiridharan Subramanian, Sebastian Michael

Version 1.1

Document Control

Project Title:	Don'sPay Mobile Payment System
Version:	1.1
Last Updated:	October 10, 2024
Status:	Draft
Document Owner:	Group 1

Version History

Version	Date	Author	Changes
1.0	Initial	Group 1	Initial version
1.1	October 10, 2024	Group 1	Enhanced requirements, added detailed specifications

Contents

1 Introduction

Don'sPay is a mobile app designed to facilitate seamless and secure payments on campus using Don Dollars and Meal Swipes. The app offers a fast and convenient payment method through QR code scanning at various campus locations, allowing students to make purchases, check balances, and view transaction histories in real-time. Don'sPay also provides account management features, enabling users to update their profiles, reset passwords, and link additional payment methods.

The primary beneficiaries of the app are students, who use it for everyday transactions and account management, and campus staff, who monitor payments, issue refunds, and audit transaction records. The app handles data related to user accounts, transaction histories, payment information, and user profiles, ensuring robust data management and secure access for the campus community.

The primary users of the app are:

- **Students**: To make payments, check balances, and view transaction histories.
- **Campus Staff**: To manage incoming payments, issue refunds, and monitor transaction records for auditing.

The app's features are designed to streamline the payment process, ensuring a user-friendly experience while providing robust data management and secure transactions for the campus community.

2 Glossary

- **Don Dollars**: Virtual currency used by students on campus for payments at dining and retail locations.
- **Meal Swipes**: Allocated meal credits that can be used at campus dining locations for meal purchases.
- **QR Code**: A two-dimensional scannable barcode used to initiate and process payments quickly.
- **Authentication**: The process of verifying a user's identity to ensure secure access to their account and prevent unauthorized usage. This is achieved through university credentials.
- **API (Application Programming Interface)**: A set of protocols and tools that allows different software components to communicate with each other. In this case, it enables Don'sPay to interact with the university's payment systems.
- **UI (User Interface)**: The visual elements and layout of the app that users interact with, designed to be intuitive and user-friendly.
- **Backend**: The server-side component responsible for processing data, managing transactions, and securely storing information.
- **Frontend**: The client-side component where the app's design and user experience come together, allowing users to interact with the system.
- **Real-time Tracking**: The capability of the app to provide immediate updates on account balances and transaction statuses as they occur, ensuring users are always aware of their current financial status.
- **Data Security**: Measures taken to protect user information and transaction data from unauthorized access, breaches, or theft through encryption, secure protocols, and access controls.

3 User Requirements

This section details the features and functionalities that Don'sPay must provide to meet the needs of its users.

1. **Payment Functionality**: The app shall allow students to make payments using QR codes at designated campus locations for dining and retail, providing a quick and efficient payment method.

2. **Balance Viewing**: The app shall enable users to view their current Don Dollars and Meal Swipes balances in real-time, ensuring they are aware of their available funds before making transactions.

3. **Transaction History**: The app shall provide a detailed transaction history for users, including the date, amount, and payment location for each transaction, facilitating easy tracking of expenses.

4. **Profile Management**: The app shall allow users to update their profile information (e.g., name, contact details, payment preferences) to ensure accurate records.

5. **Low Balance Notifications**: The app shall notify users when their balance is low, prompting them to recharge their accounts in a timely manner.

6. **Account Management**: The app shall support account management features, such as password resets, updating account settings, and linking with additional payment methods (e.g., credit cards).

7. **Transaction Record Access**: The app shall allow campus staff to access transaction records for auditing purposes, ensuring transparency and accountability.

8. **Integration with Payment System**: The app shall integrate with the university's existing payment system for real-time balance updates, ensuring users have the most accurate information.

9. **Linking Payment Methods**: The app shall allow users to link their accounts with other payment methods (e.g., credit cards) for added convenience when funding their accounts.

10. **Secure Login Mechanism**: The app shall provide a secure login mechanism using university credentials to protect user accounts from unauthorized access.

11. **Refund Processing**: The app shall enable campus staff to issue refunds to students if needed, providing a straightforward process for handling payment discrepancies.

12. **Admin Dashboard**: The app shall include an admin dashboard for campus staff to monitor app usage, transactions, and overall performance, allowing for better management and decision-making.

4 System Requirements

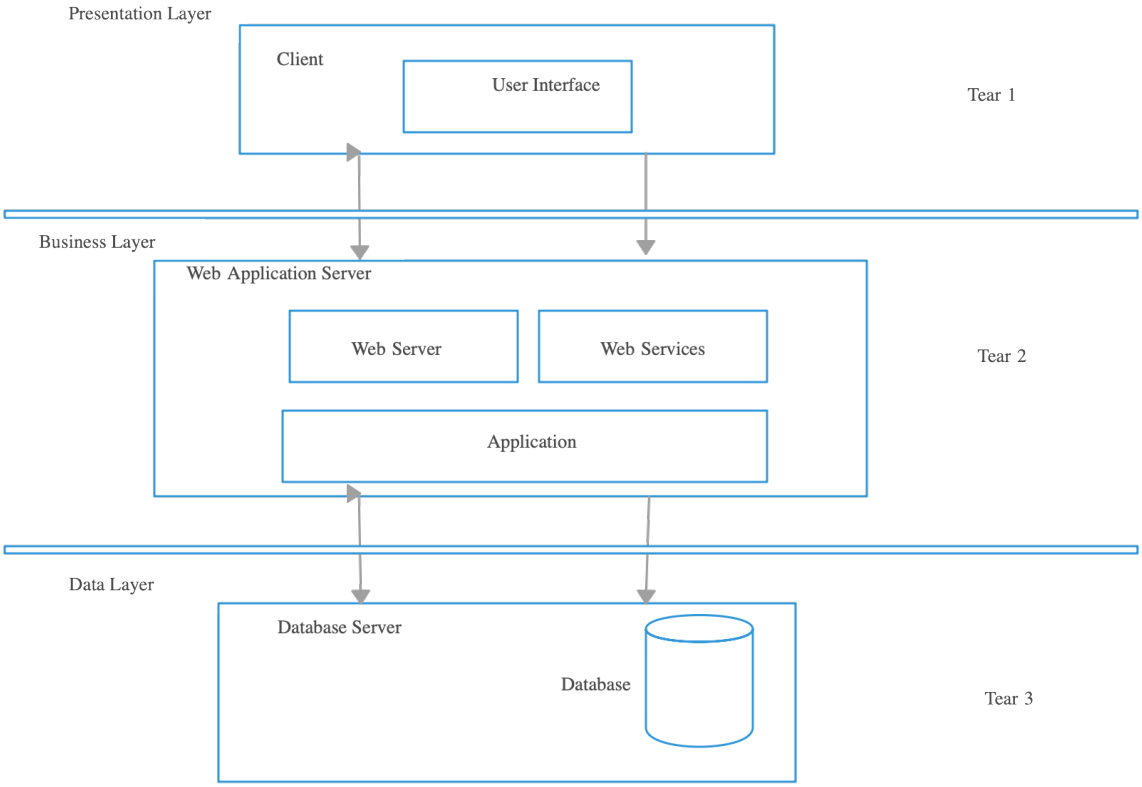
The system requirements define the technical specifications and constraints that the app must meet.

1. The app should process payments in under 2 seconds to ensure a quick and smooth user experience.
2. The system shall employ secure communication protocols (e.g., HTTPS) for all data transmission to protect user information.
3. The app should be compatible with both iOS and Android platforms to reach the entire student body.
4. The system shall automatically synchronize with the university's database to keep balances, user information, and transaction history up to date.
5. The app should be capable of supporting at least 10,000 concurrent users without noticeable performance degradation, ensuring scalability.
6. The app shall provide data encryption for sensitive information stored locally on devices and in transit.
7. The app shall use two-factor authentication for added security, especially for administrative functions and account management.

5 System Architecture

This section describes the architectural design of Don'sPay, including various layers and patterns employed to ensure modularity, scalability, and maintainability.

5.1 Layered Architecture Diagram

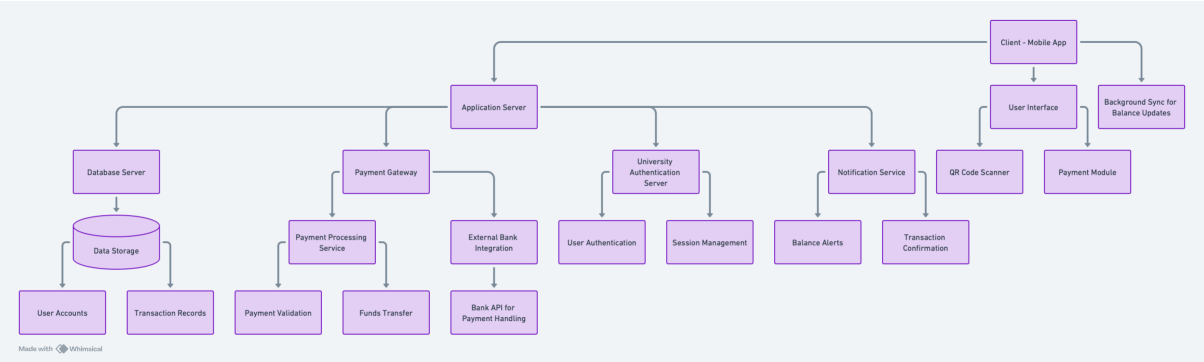


The layered architecture diagram divides the system into multiple layers, each with specific responsibilities:

- **Presentation Layer**: Manages the user interface, displaying account balances, transaction history, and payment options. It serves as the main point of interaction between the user and the system.
- **Business Logic Layer**: Contains the core functionality of the app, such as payment processing, balance updates, notifications, and managing user data. It ensures that all operations adhere to the business rules.
- **Data Access Layer**: Responsible for retrieving and storing user information, balances, and transaction records in the database. It abstracts data access details from the rest of the application, ensuring loose coupling.

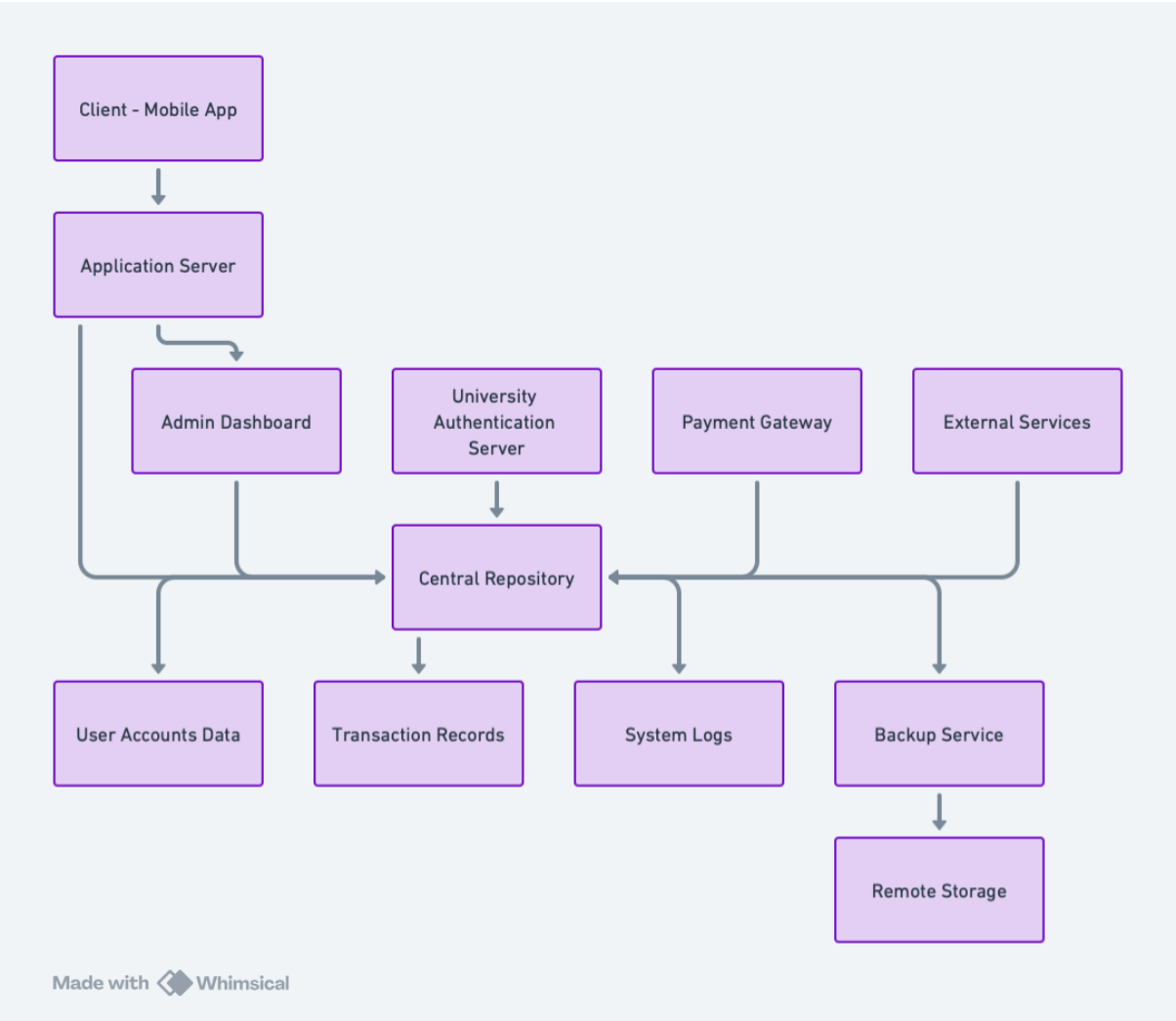
This structure promotes separation of concerns, making the app more maintainable and easier to scale.

5.2 Client-Server Architecture Diagram



The client-server architecture shows how the app's frontend (client) interacts with the backend (server). The client sends requests (e.g., payment processing, balance checks) to the server, which performs the necessary operations, accesses the database if needed, and returns responses. This architecture supports centralized data management, ensuring data consistency and providing secure access to sensitive information.

5.3 Repository Pattern Architecture Diagram

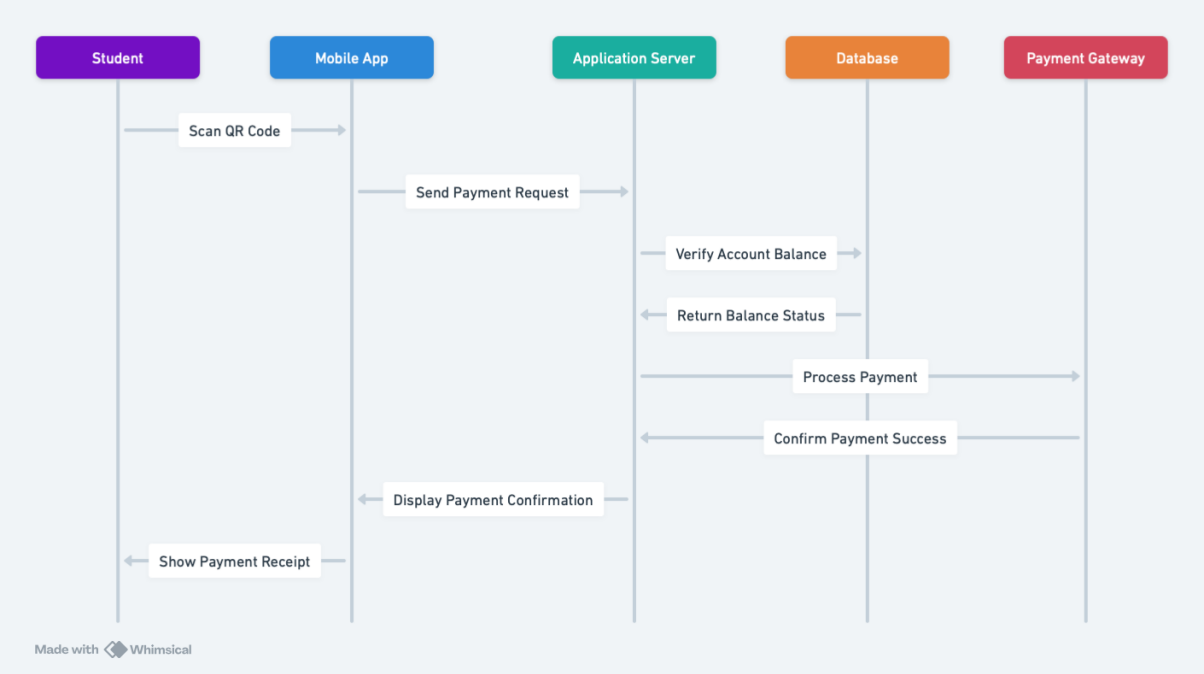


The repository pattern is used to manage data access by serving as an intermediary between the business logic and the database. It abstracts data retrieval, providing a clean

API for the business logic layer to interact with data sources. This pattern supports easy data source replacement (e.g., switching from SQL to NoSQL databases) without affecting the business logic.

6 System Models

6.1 Sequence Diagram

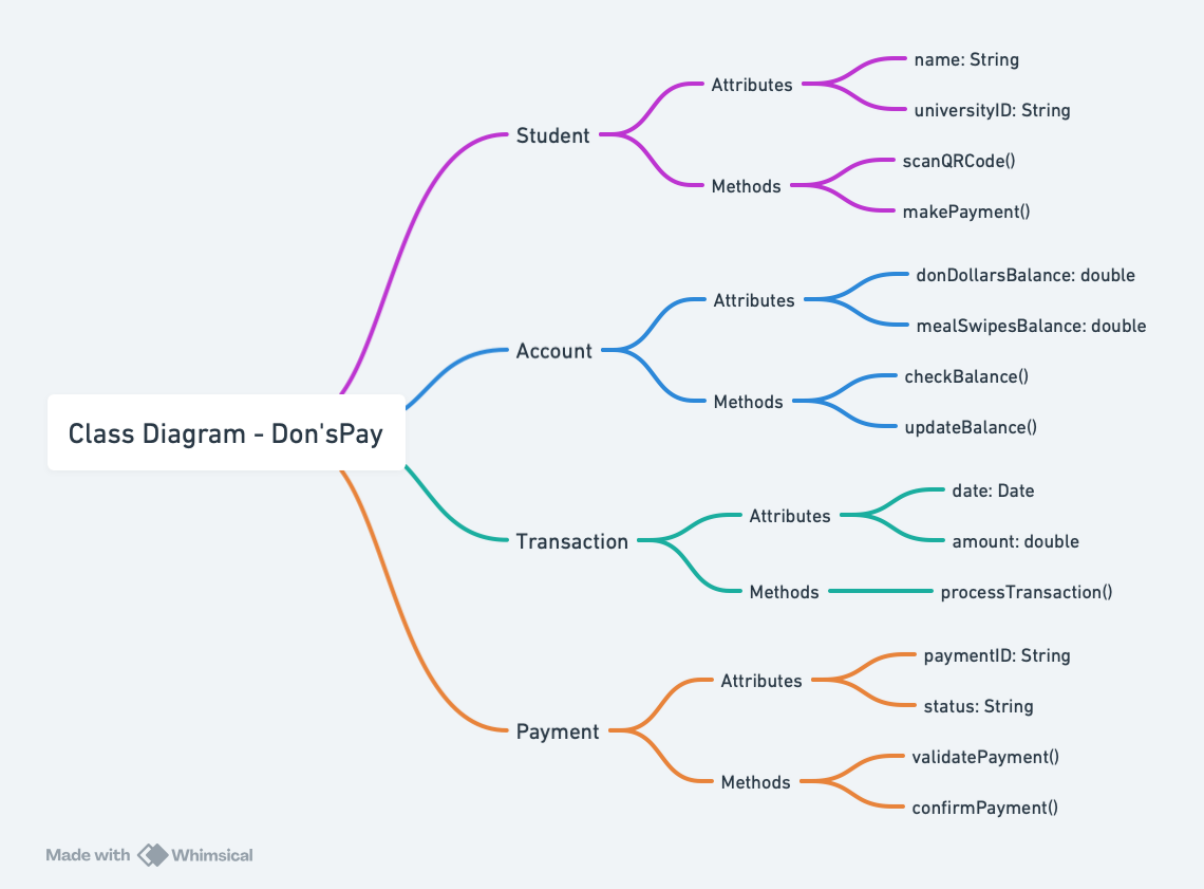


The sequence diagram illustrates the interaction between system components during a payment process:

- The user initiates the payment by scanning a QR code, which triggers a payment request.
- The app communicates with the server, which verifies the user's balance.
- If the balance is sufficient, the server processes the payment and updates the database.
- The server returns the payment status to the app, which then displays a confirmation or error message.

This diagram provides a clear representation of the workflow, helping to identify potential points of failure and optimize the system's response to errors.

6.2 Class Diagram



The class diagram illustrates the system's structure, depicting classes such as User, Payment, Transaction, and QRCode. It shows their attributes, methods, and relationships:

- **Inheritance**: Different types of payments may extend a base Payment class.
- **Composition**: The User class may have a Profile, containing personal information.
- **Associations**: The Payment class is linked to the Transaction history, tracking all user payments.

This diagram helps developers understand the system's structure and implement the code accordingly.

6.3 Use Case Template

Use Case Name	Make Payment
Actors	Student, Campus Staff
Description	Allows a student to make a payment for a purchase using Don Dollars or Meal Swipes by scanning a QR code at the payment terminal.
Preconditions	The student must have a sufficient balance in their account. The payment terminal must be connected to the system.
Postconditions	The payment is processed, and the student's account balance is updated. A receipt is generated and displayed.
Normal Flow	1. The student scans the QR code at the payment terminal. 2. The mobile app sends a payment request to the application server. 3. The application server verifies the balance with the database. 4. If the balance is sufficient, the payment is processed, and the balance is updated. 5. The app confirms the payment success to the student and displays a receipt.
Alternative Flows	- If the balance is insufficient, the payment is declined, and an error message is shown to the student. - If there is a network error, the transaction fails, and the student is prompted to retry.
Exceptions	- System maintenance mode, preventing payments. - QR code is invalid or expired.

7 Conclusion

7.1 Summary of the Document

This document presents a detailed overview of Don'sPay's requirements, including user needs, system specifications, architectural considerations, and use case scenarios. The app is designed to enhance the campus payment experience by offering a secure, fast, and user-friendly platform.

7.2 Future Work and Enhancements

Future enhancements may include:

- Integration with additional payment methods, such as mobile wallets and wearables.
- Expansion of payment options to off-campus vendors.
- Advanced analytics features for administrators to track spending trends and optimize campus services.
- Support for voice-activated payments for improved accessibility.