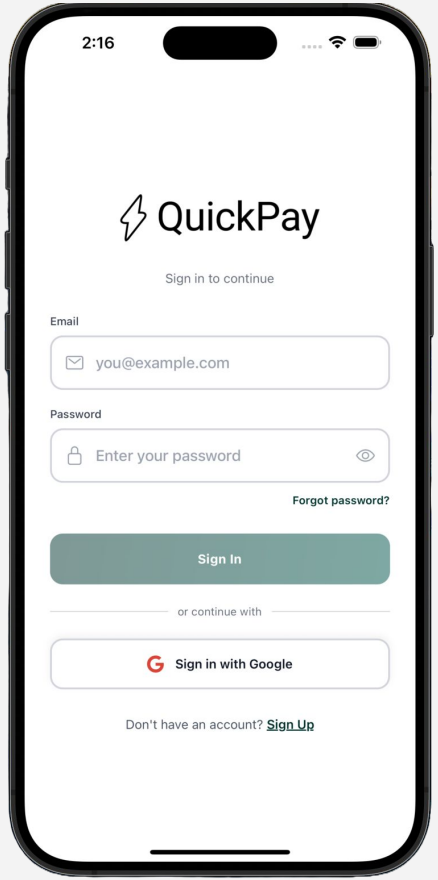
QuickPay

Team B – CSCI 441 Software Engineering

Members:

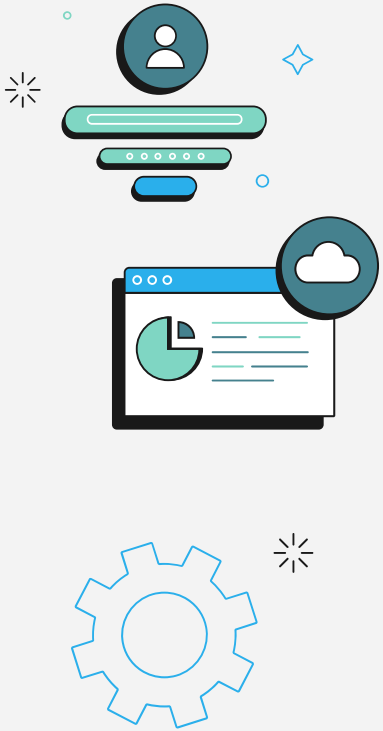
- Chanrattnak Mong - Frontend & Authentication Engineer
- Seanglong Lim - Backend Integration Engineer
- Seth Tharo Hour - Backend Integration Engineer
- Sok Sreng Chan - Frontend Engineer



Introduction and Project Motivation

- **Problem:** Managing finances is fragmented and users juggle multiple bank apps, spreadsheets, and budgeting tools, leading to confusion and overspending.
- **Consequences**: Time wasted on manual updates, frequent errors, and financial stress due to lack of real time visibility.
- **Current Solution:** Separate banking and budgeting apps with no full integration or automation.
- **QuickPay Solution:** A non custodial app linking all bank accounts for real time budgeting, QR payments, and group expense splitting.
- **Benefits:** Saves time, improves transparency, enhances security, and reduces financial stress.

Customer Problem Statement



- Managing personal finances has become overwhelming for many users.
- Users struggle to:
  - Track spending across multiple bank accounts.
  - Manage and split group expenses.
  - Avoid overspending due to lack of visibility.
  - Identify where their money goes in real time.
  - Understand where their money goes
- Existing budgeting tools lack **real-time synchronization** and **flexibility**, making it easy to overspend and hard to know which account to use.
- **QuickPay** provides a **unified, real-time, non-custodial finance app** that:
  - Connects all bank accounts.
  - Features budgeting and tracking.
  - Simplifies payments and group expenses
  - Gives users full control over their money.

System Specifications: Functional

1. Multi-Account Management
▪ Link multiple bank accounts through secure APIs (Plaid).

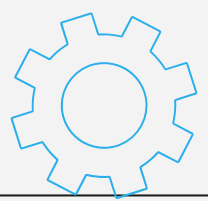
2. QR-Based Payment System
▪ Scan or upload QR codes for instant peer-to-peer or group payments.

3. **Interactive Budget Visualization**
▪ Real-time spending overview using dynamic flow charts.

4. Group Expense Splitting
▪ Split bills automatically with calculated shares and payment tracking.

5. Unified Dashboard
▪ Display all linked accounts, balances, and transaction history in one view.

6. User Management and Favorites
▪ Add and Update other users as favorites for faster transactions.



System Specifications: Non-Functional

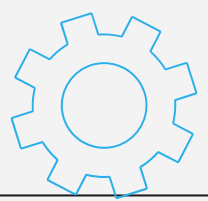
1. Security
- Data encrypted in transit and at rest.
- Supports multi-factor authentication for user protection.

2. Performance
- Payments processed within seconds.
- QR scanning and transactions optimized for quick response.

3. **Reliability**
- High uptime with regular backups and recovery measures.

4. Usability
- Simple, clean interface for easy navigation and readability.

5. Privacy
- Collects minimal user data and follows data protection standards.



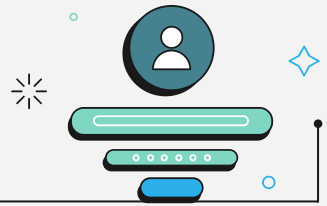
System Specifications: User Interface Requirements

A User Shall:

- View their **profile and app settings** icons.
- View all **net worth from multiple banks**, send and request money.
- Access **transaction history** with filters and details.
- Use a **navigation bar** to move between pages.
- Enter **Account Number**, **Scan or upload QR codes** for payments.

A User Should:

- Configure **budget blocks** and set spending limits.
- **Save favorite users** for quick transfers.
- **Update profile information** such as ID, contact, and security settings.



Product Roadmap

Week 01-03

Project Planning

- ❏ Defined problem statement and customer needs
- ❏ Outlined goals and system architecture

Week 04-06

System Design

- ❏ Built domain and class diagrams, data models, and sequence diagrams.
- ❏ Designed UI wireframes

Week 07-10

Implementation

- ❏ Migrated from Firebase to Supabase PostgreSQL
- ❏ Integrated Clerk Authentication

Week 11-12

Testing & Integration

- ❏ Validated database operations, authentication, and notifications
- ❏ Ensured system met reliability, performance, and security standards

Week 13-14

Finalization

- ❏ Completed final report and presentation
- ❏ Conducted team coordination and merging on GitHub

Week 15- Onwards

Future Work

- ❏ Add real-time alerts and recurring payment management
- ❏ Enhance security and analytics modules

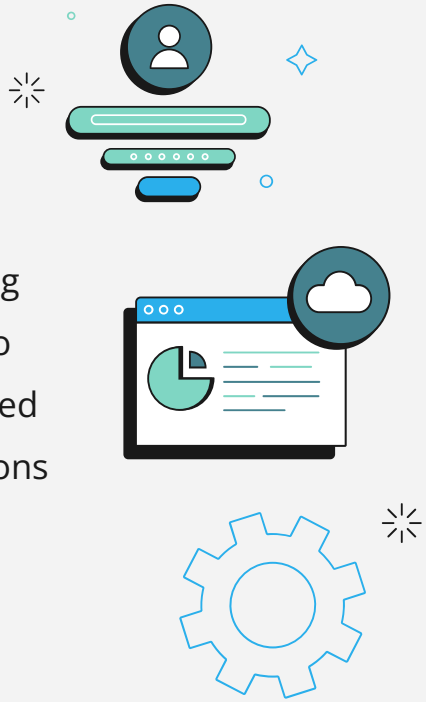
Project Challenges & Future Work

Challenges:

- Security Implementation
- Unfamiliar API Integrations
- Real-Time Data Handling
- Cross-Platform Development (Android & iOS)
- Time Constraint

Future Work:

- QR Code/Link Transfer
- Implement Multi-Bank Linking
- Enable Request/Send Bank to Bank Payments once Approved
- Support Real-Time Transactions
- Conduct More Security Tests



Conclusion

- QuickPay aims to transform how people manage their daily finances by offering an all-in-one platform that connects multiple bank accounts, automates budgeting, and simplifies group payments.
- With our advance integration like **Plaid**, **Stripe**, and **Clerk**, our team designed a secure and user-friendly system that gives users real-time visibility into their financial activities.
- The project not only reduces the stress of manual tracking but also encourages responsible spending habits.
- Moving forward, we plan to enhance QuickPay features, improve its user interface, and prepare it for real-world deployment to help people take full control of their finances effortlessly.
- **Test our project:** https://github.com/CSCI441-QuickPay/QuickPay-MobileApp