UniWallet Banking

By Joshua Purushothaman, Jorge Agueros, Aldo Carmona, Rohan Cheruku, Zunaira Firdous, Tejas Ramanujam, Varun Saravanan

Objective

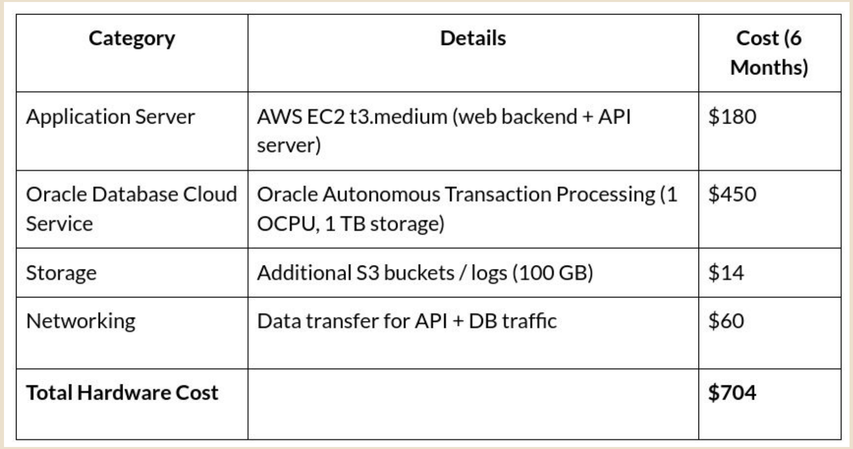
1. **Improve Financial Confidence & Literacy:** Nearly 47% of college students say they don’t feel prepared to manage their money, according to a survey of 30,000 students.

2. **Encourage Better Budgeting Habits:** Only 39% of students stick to a monthly budget, despite many rating their financial skills as “good” or “excellent.”

3. **Reduce Financial Stress & Risky Debt Behaviors:** More than half of Gen Z students (53%) report that money management is their most daunting challenge, and only about half (51%) plan to pay off credit card bills in full.

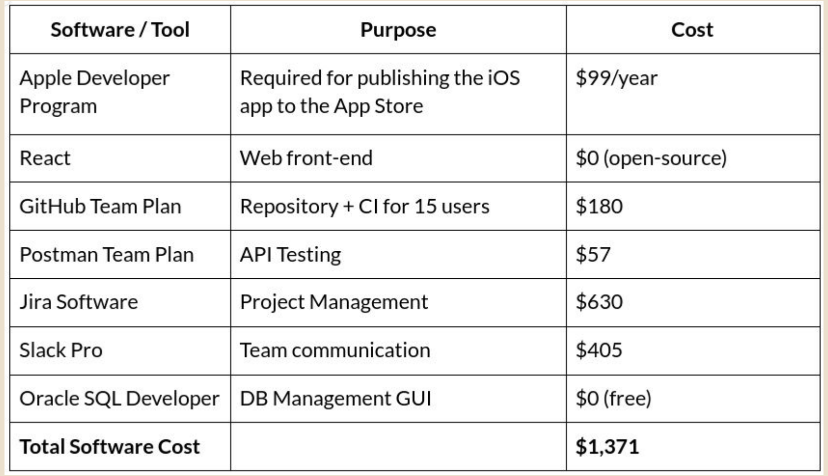
Cost Estimation

Hardware Costs:



Category	Details	Cost (6 Months)
Application Server	AWS EC2 t3.medium (web backend + API server)	$180
Oracle Database Cloud Service	Oracle Autonomous Transaction Processing (1 OCPU, 1 TB storage)	$450
Storage	Additional S3 buckets / logs (100 GB)	$14
Networking	Data transfer for API + DB traffic	$60
Total Hardware Cost		**$704**

Software Costs:



Software / Tool	Purpose	Cost
Apple Developer Program	Required for publishing the iOS app to the App Store	$99/year
React	Web front-end	$0 (open-source)
GitHub Team Plan	Repository + CI for 15 users	$180
Postman Team Plan	API Testing	$57
Jira Software	Project Management	$630
Slack Pro	Team communication	$405
Oracle SQL Developer	DB Management GUI	$0 (free)
Total Software Cost		**$1,371**

Cost Estimation

Function Point Algorithmic Estimation:

- Effort Estimation: 448 FP / 10 Productivity ≈ 44.8 person-months.
- Project Duration: 44.8 person-months / 15 developers ≈ 12 weeks.

Personnel Costs:

- Developer Cost:
    - Pay rate: $35/hour
    - Work duration: 60 days × 8 hours/day = 480 hours per developer
    - Cost per developer: 480 × $35 = $16,800
    - Team of 15 developers: 15 × $16,800 = **$252,000**
- Training Cost
    - 4 trainees × 50 hours × $20/hour = **$4,000**

Project Timeline

	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
Weeks	1-2	3-4	5-6	7-8	9-10	11-12
	Requirements & Architecture Creation of requirements documents and diagrams.	**Backend Development** Creation of user account modules, database schema, and Authentication	**Backend Development cont.** Creation of user transactions, API creation, and validation logic	**Frontend Development** Creation of user interfaces. Main screens and user flows	**Polishing** Further refinement of user interfaces. API integration, as well as increase in error Handling	**Testing & Deployment** Perform system tests. Fix bugs. Deployment

Functional Requirements

1

Withdraw Deposits

The system must allow users to withdraw and deposit funds to and from an account

2

Secure Login

The System must authenticate user logins such as two-factor authentication

3

Employee Managed

Bank employee must have access to manage customer accounts

4

Generate account statements

The system must generate account statements that provides general information of the user account

5

Add new accounts

The system must allow the addition of new accounts.

Non-Functional Requirements

1

Usability

The system shall allow users to check their account balance within 3-4 clicks

2

Performance

The system shall load transaction searches within 3 seconds

3

Dependability

The system shall maintain 99% uptime.

4

Security

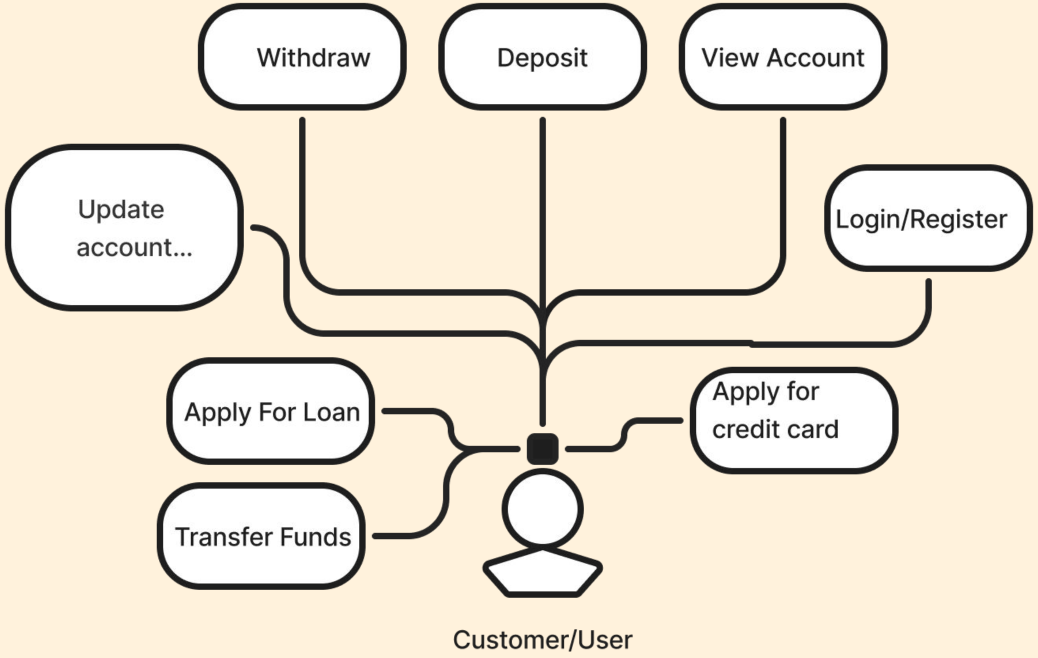
The system must encrypt all stored data and shall hash all user passwords

5

Ethical

The system must comply with current US banking regulations, and must comply with applicable data protection laws.

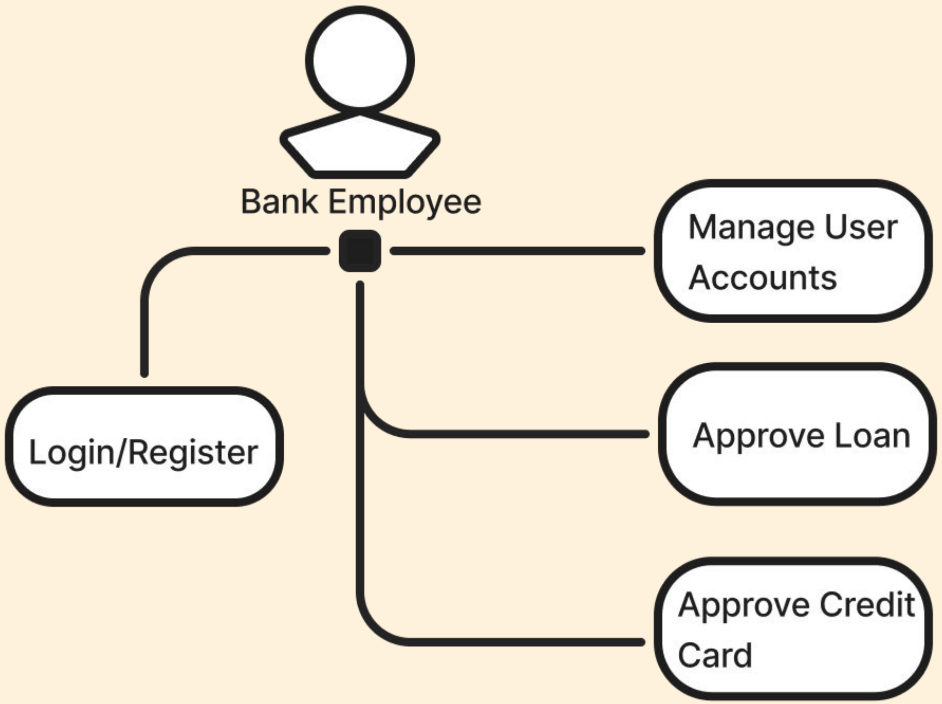
Customer/User



Customer/User

- The Customer/User will have their own view of software
- Customer can perform the key actions such as withdrawing, depositing, and viewing balances.
- Able to manage account settings and preferences.
- Include additional functionality such as applying for loans, credit cards and transfer of funds

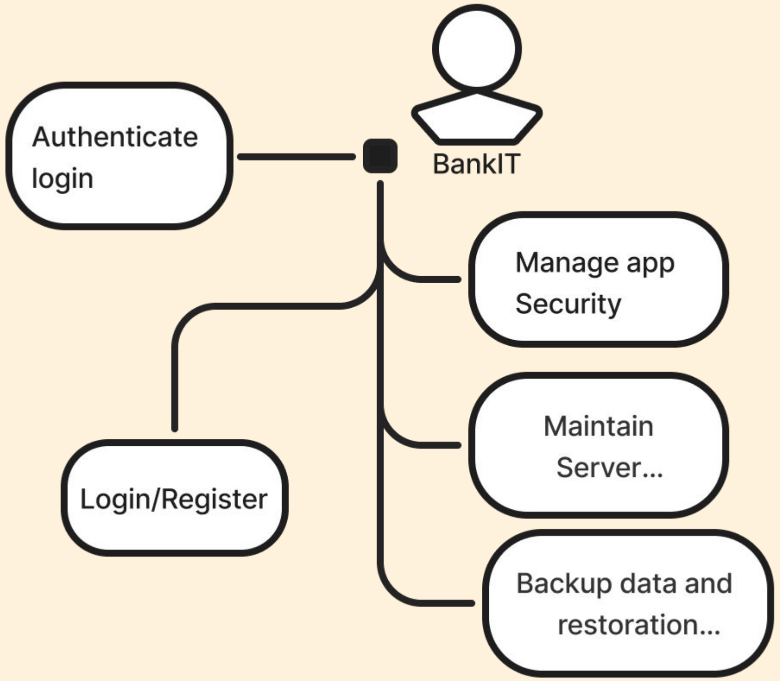
Bank Employee



Bank Employee

- Authentication
- Bank employees login to access a view separate from the customer
- Manages their customers accounts.
- Can approve loan and credit card applications

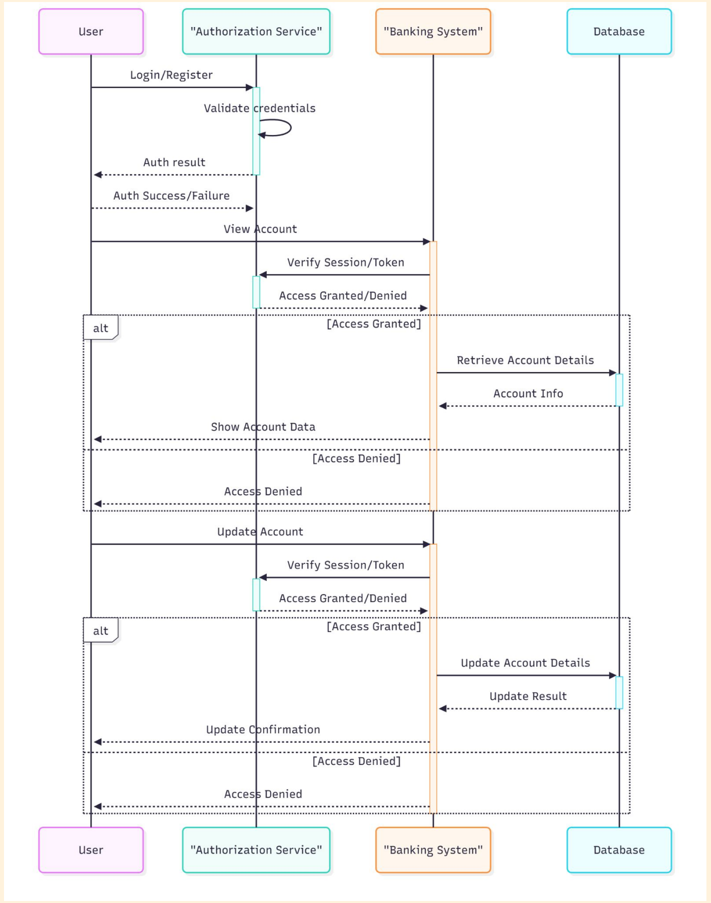
BankIT



Bank IT Employee

- The Bank IT team have a separate view where they manage various system functionalities
- Login authentication
- Software security
- Server maintenance
- Handles data process such as backups and restoration

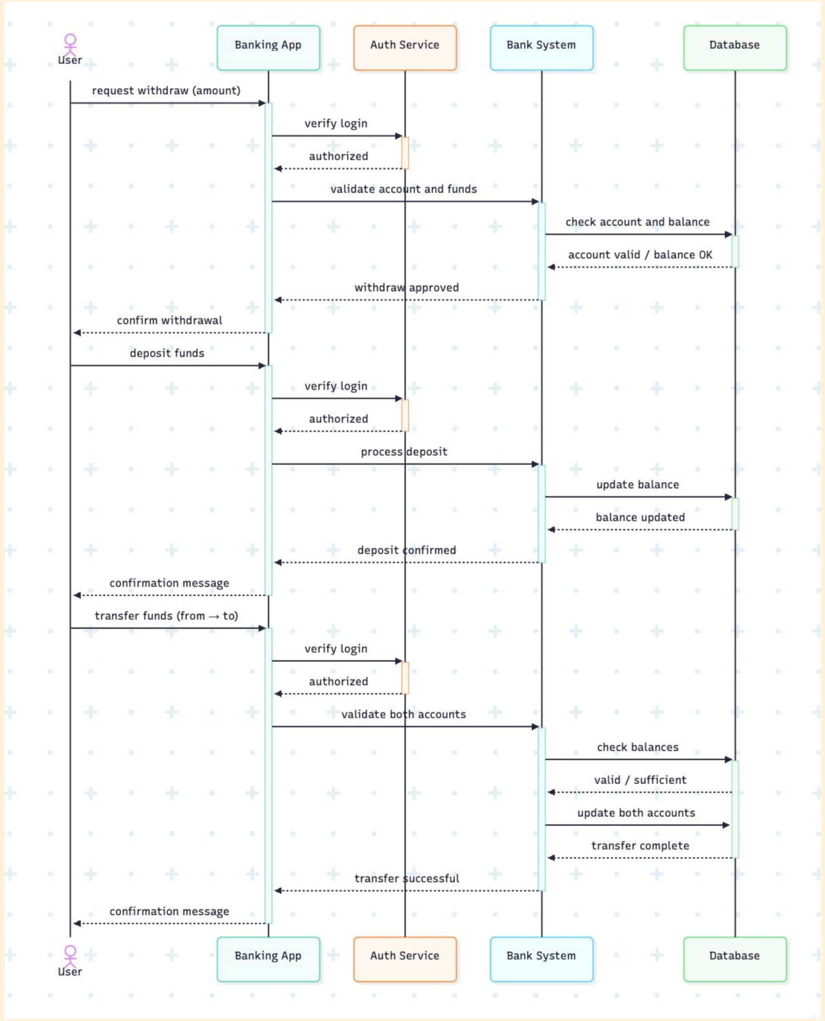
Sequence Diagram



- Shows how the user logs in and how every action goes through an authorization check.
- Displays the process for viewing the account and how the system retrieves data if access is granted.
- Illustrates the update account flow, where the system verifies the session before updating the database.
- Highlights that any failed authorization results in access being denied.

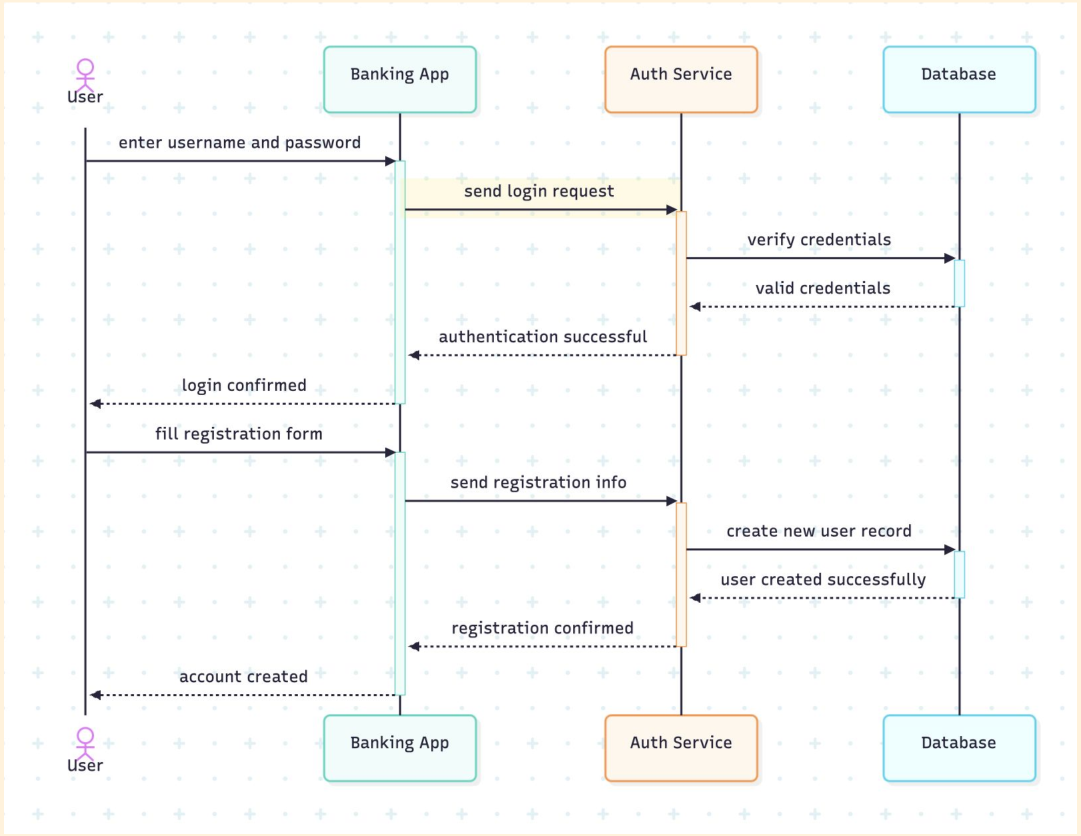
Sequence diagram

This diagram presents how the User, App, Auth Service, Bank System, and Database communicate during withdraw, deposit, login, registration, and fund transfer processes. Activation bars show when each component is actively processing a request.

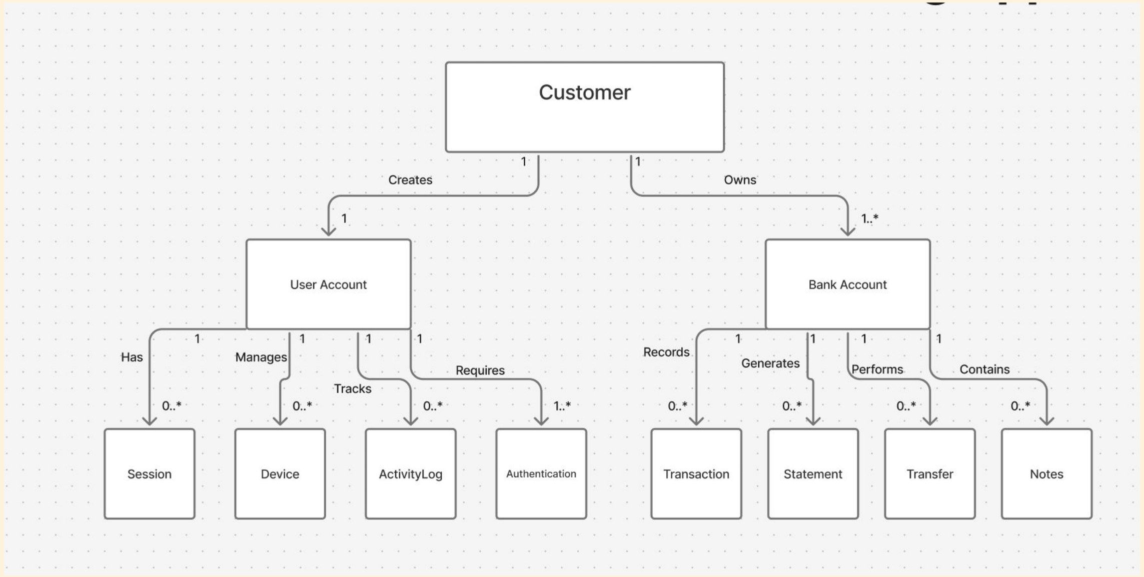


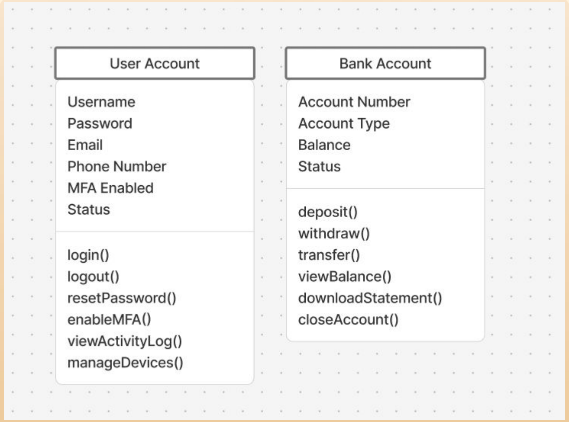
Sequence Diagram

This sequence diagram illustrates how the system handles user login and registration. It shows the message flow between the User, Banking App, Authentication Service, and Database. Activation bars indicate when each component is actively processing the request, highlighting the full interaction sequence from user input to system confirmation.



Class diagram

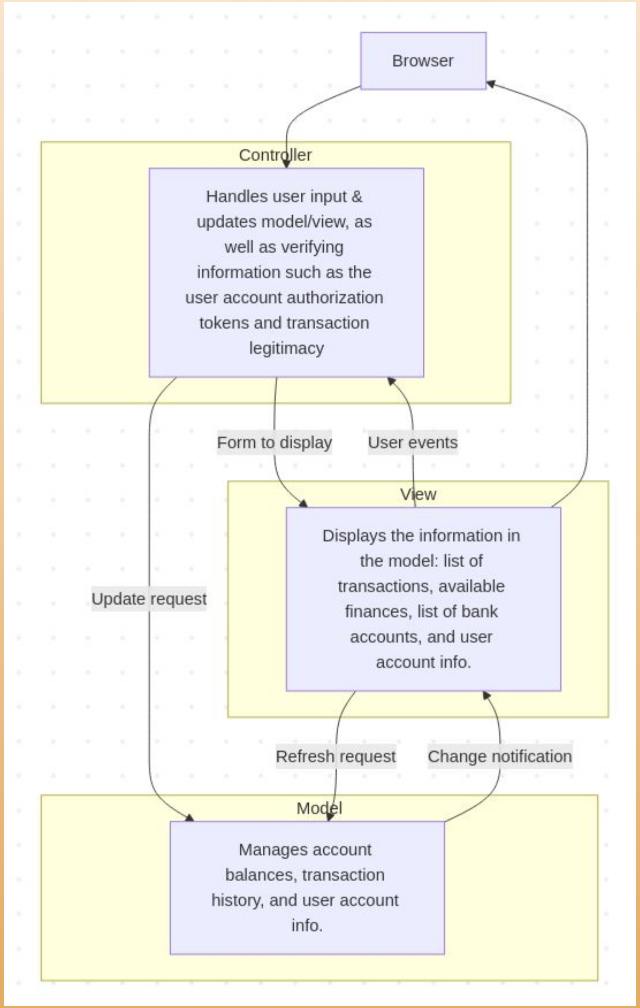




- Customer creates one User Account and owns multiple Bank Accounts
- User Account handles authentication, sessions, devices, and activity tracking
- Bank Account manages transactions, statements, transfers, and notes
- System includes security features like MFA and comprehensive activity logging

Architectural Design

- Our project utilizes the **Model-View-Controller**(MVC) pattern.
- **Model:** Manages system data and data operations such as:
    - account balances, transaction history, and user account information.
- **View:** Displays information from the Model component to the user’s browser.
    - List of transactions, current balance.
- **Controller:** Handles user input and passes it into the Model and View components. The component also verifies information and asserts security measures per interaction, such as:
    - Withdrawal and user account authorization.


Browser
Controller
Handles user input & updates model/view, as well as verifying information such as the user account authorization tokens and transaction legitimacy
Form to display
User events
View
Displays the information in the model: list of transactions, available finances, list of bank accounts, and user account info.
Update request
Refresh request
Change notification
Model
Manages account balances, transaction history, and user account info.

Sources

[1] C. L. Vien, “College students think they manage money well, but they don’t, survey finds,” *Journal of Accountancy*, Sep. 10, 2015. https://www.journalofaccountancy.com/news/2015/sep/financial-literacy-skills-201512981/

[2] D. Zapp, “Money Matters on Campus Report,” *Everfi*, Nov. 12, 2025. https://everfi.com/resources/white-paper/money-matters-on-campus-report/

[3] M. Burkhard and J. Madden, “Buckling under Debt and a Lack of Financial Knowledge, College Students Feel Toll of Financial Stress | American International Group, Inc.,” *American International Group, Inc.*, 2019. https://aig.gcs-web.com/news-releases/news-release-details/buckling-under-debt-and-lack-financial-knowledge-college (accessed Nov. 24, 2025).

Thank you!