Introduction to Product Use Case

Training Session

Today's Challenge:

Personal Banking Management System

We'll work together to identify problems, generate use cases, and design solutions step by step.

What is a Product Use Case?

Definition:

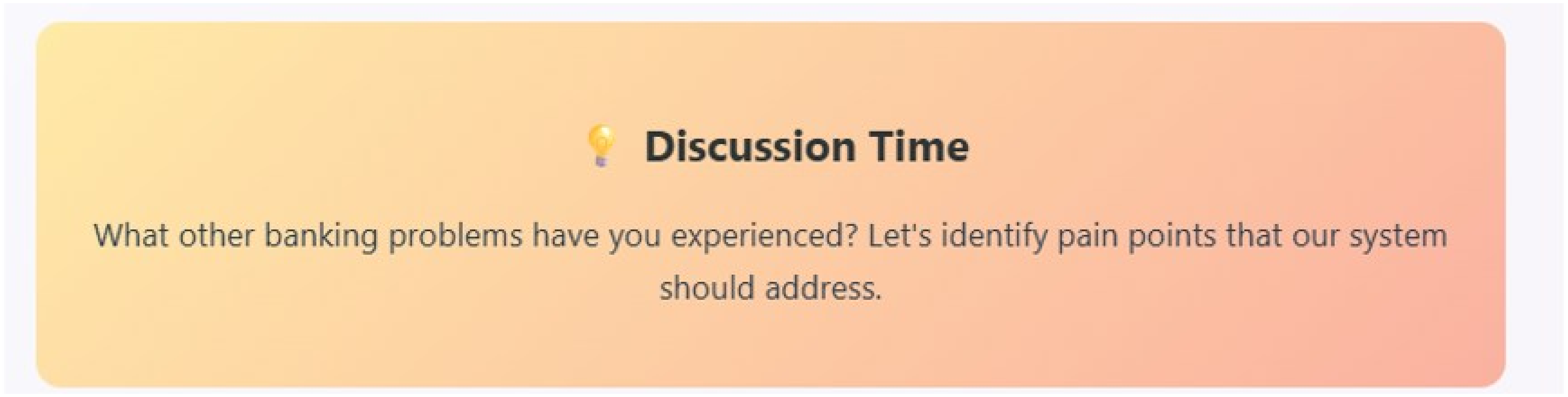
A use case describes **how a user interacts with a system** to achieve a specific goal. It captures the functional requirements from the user's perspective.

Key Components:

- **Actor:** Who is using the system? (Customer, Admin, etc.)
- **Goal:** What does the actor want to achieve?
- **Scenario:** Step-by-step interaction flow
- **Preconditions:** What must be true before starting?
- **Postconditions:** What happens after successful completion?

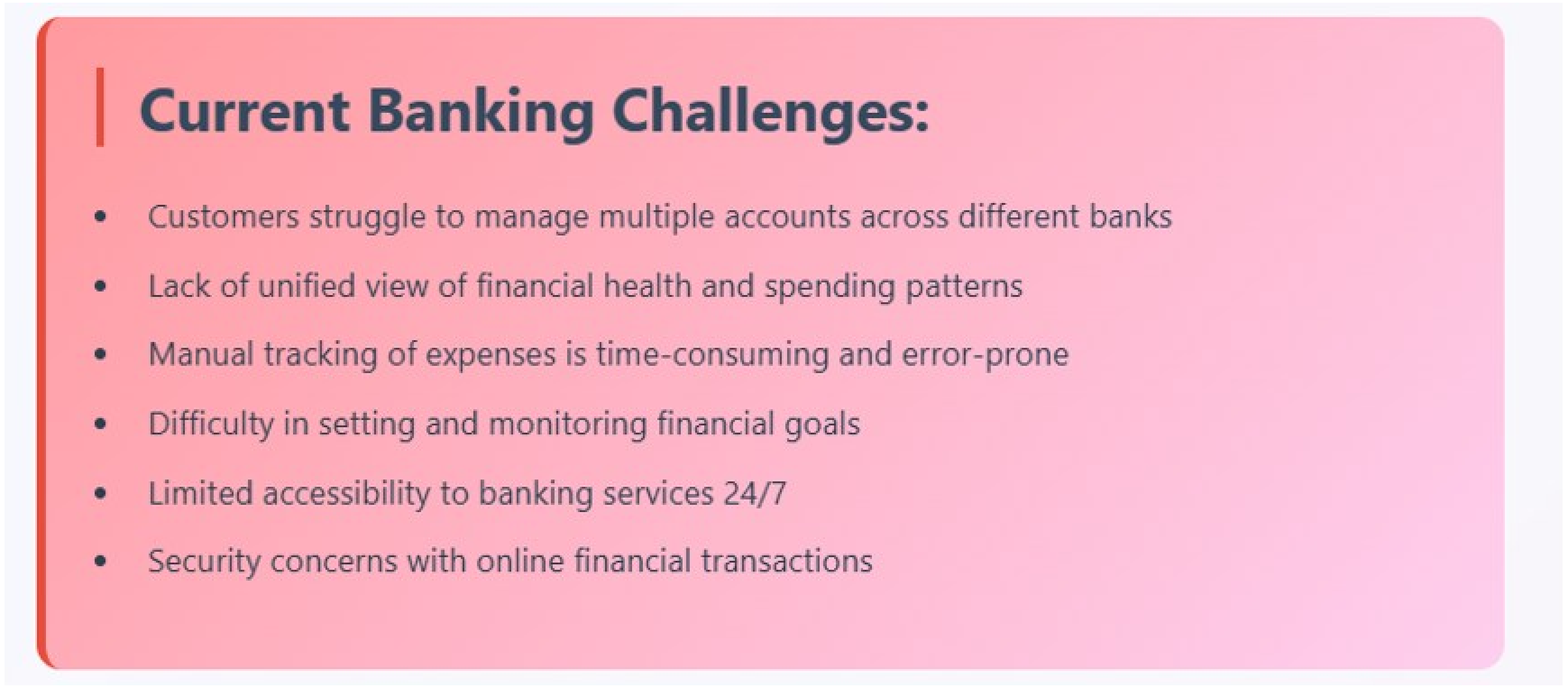
Why Use Cases Matter: They bridge the gap between business requirements and technical implementation, ensuring we build what users actually need.

Product Statement Analysis



💡 Discussion Time

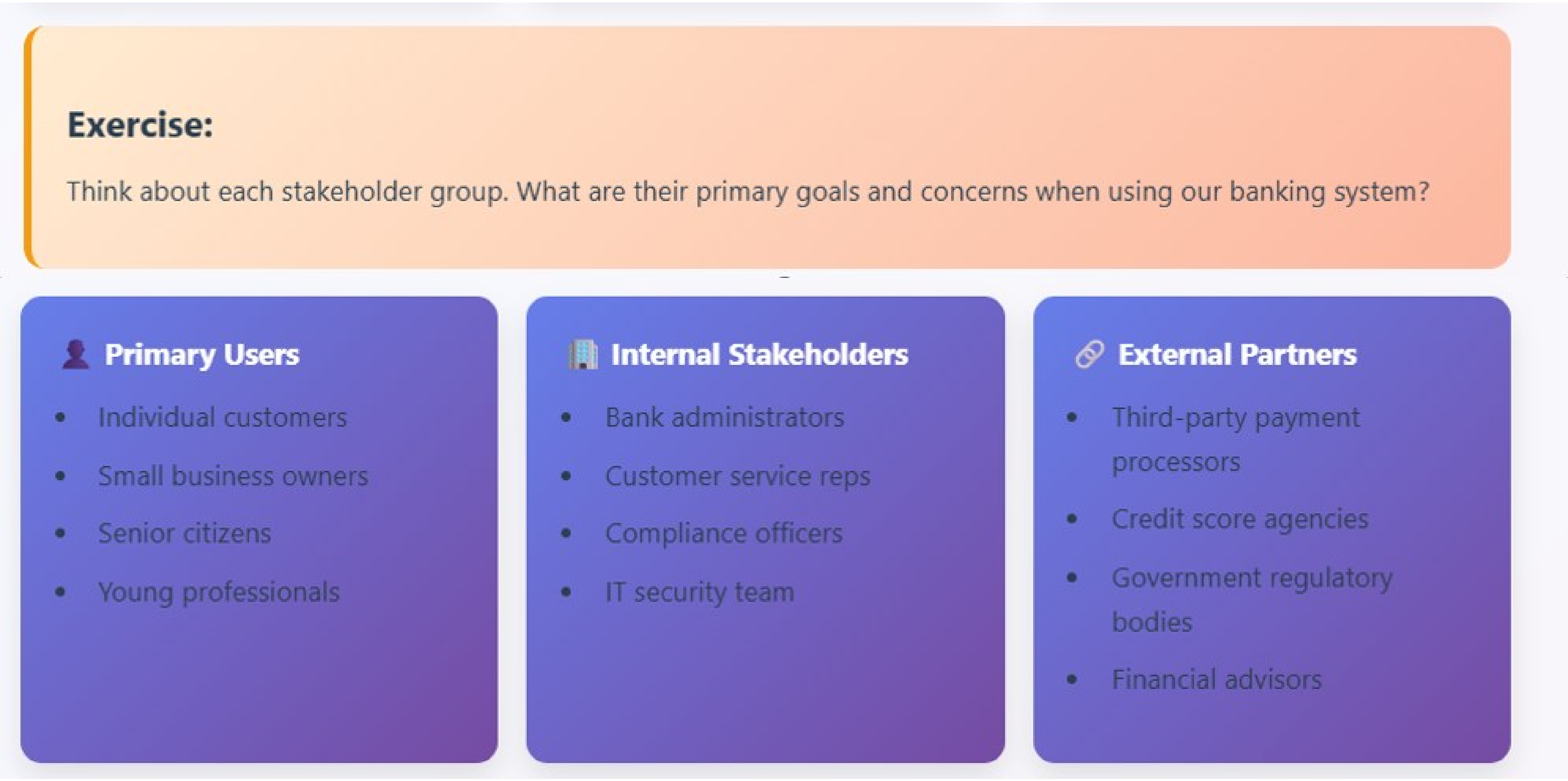
What other banking problems have you experienced? Let's identify pain points that our system should address.



Current Banking Challenges:

- Customers struggle to manage multiple accounts across different banks
- Lack of unified view of financial health and spending patterns
- Manual tracking of expenses is time-consuming and error-prone
- Difficulty in setting and monitoring financial goals
- Limited accessibility to banking services 24/7
- Security concerns with online financial transactions

Who are our stakeholders?



Exercise:

Think about each stakeholder group. What are their primary goals and concerns when using our banking system?

👤 Primary Users

- Individual customers
- Small business owners
- Senior citizens
- Young professionals

🏢 Internal Stakeholders

- Bank administrators
- Customer service reps
- Compliance officers
- IT security team

🔗 External Partners

- Third-party payment processors
- Credit score agencies
- Government regulatory bodies
- Financial advisors

Let's Generate Core Use Cases

🎯 Your Turn to Think

Based on our problem analysis, what are the main activities users need to perform?

Guided Questions:

- How do customers currently manage their money?
- What daily, weekly, or monthly tasks do they perform?
- What information do they need access to?
- What decisions do they need to make?
- What actions do they need to take?

Let's brainstorm together and list potential use cases on the board!

Sample Core Use Cases

Account Management

- View account balances
- Check transaction history
- Update personal information
- Manage account settings

Transactions

- Transfer money between accounts
- Pay bills online
- Send money to other users
- Schedule recurring payments

Financial Planning

- Set savings goals
- Track spending patterns
- Create budgets
- Generate financial reports

Security & Support

- Authenticate user login
- Report suspicious activity
- Reset forgotten passwords
- Contact customer support

Detailed Use Case: Money Transfer

Use Case: Transfer Money Between Accounts

Actor: Registered Customer

Goal: Transfer funds from one account to another securely

📋 Main Flow:

1. Customer logs into the banking system
2. System authenticates and displays dashboard
3. Customer selects "Transfer Money" option
4. System displays transfer form
5. Customer selects source account, destination account, and amount
6. System validates sufficient funds and account details
7. Customer reviews transaction details and confirms
8. System processes transfer and updates account balances
9. System sends confirmation to customer

🚨 Alternative Flows:

- Insufficient funds → Display error, suggest alternatives
- Invalid account → Show error message, request correction
- System timeout → Save draft, allow retry

From Use Cases to Solution Design

🛠️ Next Steps in Our Process:

1. **Prioritize Use Cases:** Which are most critical for MVP?
2. **Create User Stories:** Break down use cases into development tasks
3. **Design System Architecture:** How will we implement these features?
4. **Define Data Models:** What information do we need to store?
5. **Plan User Interface:** How will users interact with each feature?
6. **Consider Non-functional Requirements:** Security, performance, scalability

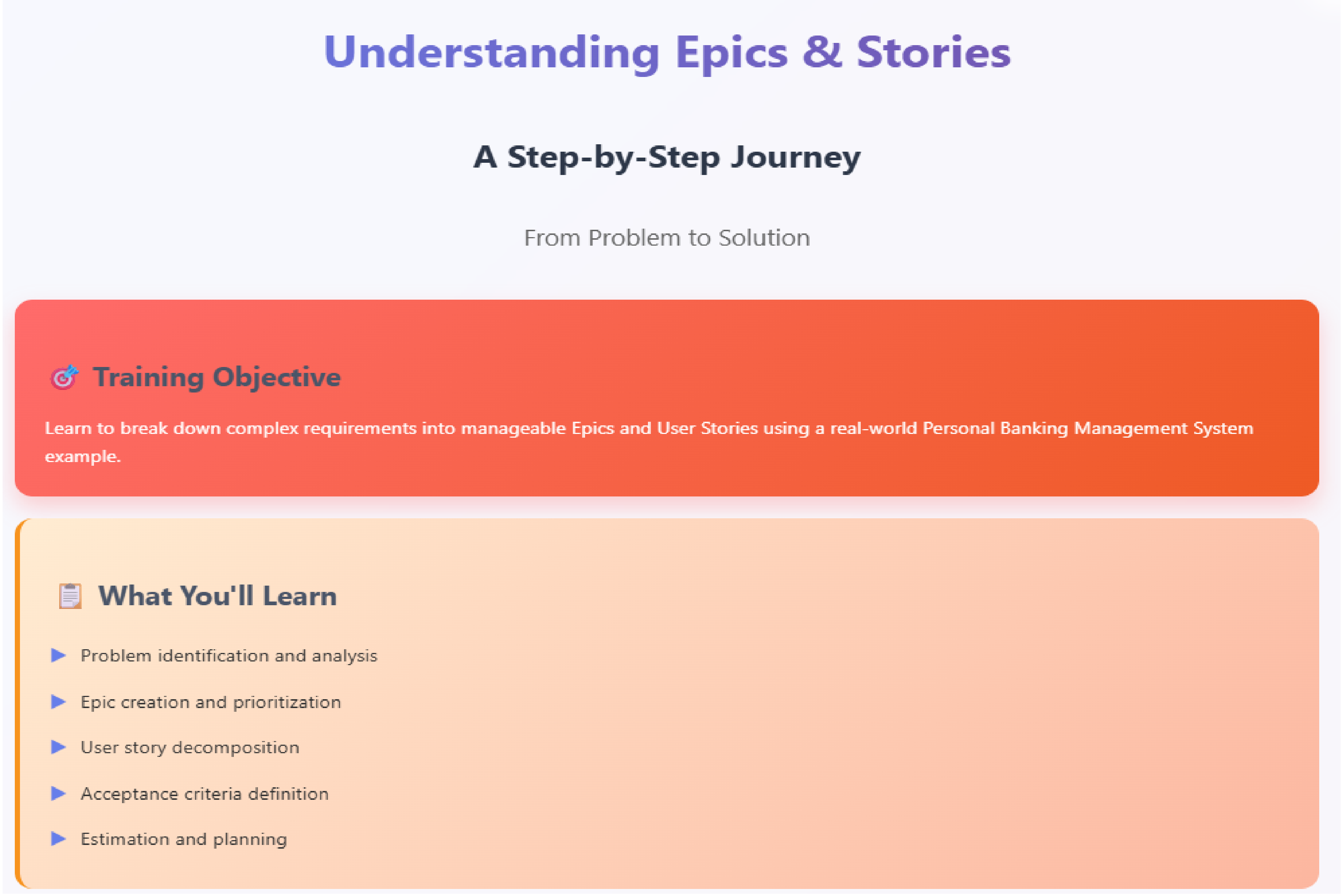
🎯 Your Assignment

Choose one use case we identified today and create a detailed specification including:

- Complete main flow
- Alternative flows
- Preconditions and postconditions
- Business rules
- UI mockup sketch

🏆 Key Takeaway

Use cases are the foundation of good software design. They help us understand user needs before we write a single line of code!



Understanding Epics & Stories

A Step-by-Step Journey

From Problem to Solution

🎯 Training Objective

Learn to break down complex requirements into manageable Epics and User Stories using a real-world Personal Banking Management System example.

📋 What You'll Learn

- ▶ Problem identification and analysis
- ▶ Epic creation and prioritization
- ▶ User story decomposition
- ▶ Acceptance criteria definition
- ▶ Estimation and planning

Explain the Waterfall model vs the Agile model

Waterfall vs. Agile

A Tale of Two Methodologies

Waterfall
Traditional & Sequential

VS

Agile
Modern & Flexible

Waterfall Model: The Traditional Approach

Linear & Sequential

Progress flows in one direction, like a waterfall. Each phase must be completed before the next begins.

Defined Phases

Requirements → Design → Implementation → Testing → Deployment

Upfront Planning

All project requirements are gathered and defined at the very beginning.

Documentation-Heavy

Relies on comprehensive documentation for each phase.

Best Suited For:

Projects with stable, well-understood requirements and a clear end goal (e.g., construction, manufacturing).

Visual Metaphor

A cascading waterfall, moving steadily downwards from one level to the next.

Agile Model: The Modern & Flexible Approach

Iterative & Incremental

The project is broken down into small, manageable cycles called "sprints."

Continuous Feedback

Constant collaboration with the customer to adapt to changing requirements.

Working Software is Key

Prioritizes delivering a functional product over extensive documentation.

Flexibility

Changes can be incorporated throughout the development process.

Best Suited For:

Projects where requirements are expected to evolve and change (e.g., software development, product design).

Visual Metaphor

A cyclical or iterative loop, indicating continuous improvement and adaptation.

🚨 The Problem Statement

Current Banking Challenges

Scenario: A traditional bank wants to modernize their services and provide customers with a comprehensive digital banking experience.

Key Pain Points:

- ▶ Customers must visit branches for most banking operations
- ▶ No real-time account monitoring capabilities
- ▶ Manual transaction processing leads to delays
- ▶ Limited visibility into spending patterns and financial health
- ▶ No integrated investment or savings management tools
- ▶ Security concerns with current legacy systems

💡 Exercise for Students

Think & Discuss: What other banking pain points have you experienced? How might technology solve these issues?

📊 Analyzing the Requirements

Personal Banking Management System

A comprehensive digital platform that empowers customers to manage their entire financial ecosystem from anywhere, anytime.

Core System Requirements

🏦 Account Management

- Multiple account types
- Real-time balance tracking
- Account statements

💸 Transaction Services

- Fund transfers
- Bill payments
- Transaction history

📈 Financial Analytics

- Spending insights
- Budget management
- Financial goals tracking

🔒 Security & Support

- Multi-factor authentication
- Customer support
- Notification system

🎯 Understanding Epics

What is an Epic?

An Epic is a large body of work that can be broken down into smaller, manageable pieces (User Stories). Think of it as a major feature or capability.

Epic Characteristics

- **Large Scope:** Too big to complete in a single sprint
- **Business Value:** Delivers significant value to users/business
- **Decomposable:** Can be broken into smaller user stories
- **Cross-functional:** May require multiple teams/skills

🏛 Example Epic: Account Management

As a bank customer, I want to manage my accounts digitally so that I can access my financial information anytime and reduce dependency on branch visits.
Business Value:
Reduces operational costs, improves customer satisfaction, increases digital engagement

🤔 Student Exercise

Can you identify what makes this an Epic rather than a User Story? Discuss the scope and complexity.

📖 Understanding User Stories

What is a User Story?

A **User Story** is a short, simple description of a feature told from the perspective of the person who desires the new capability.

User Story Format

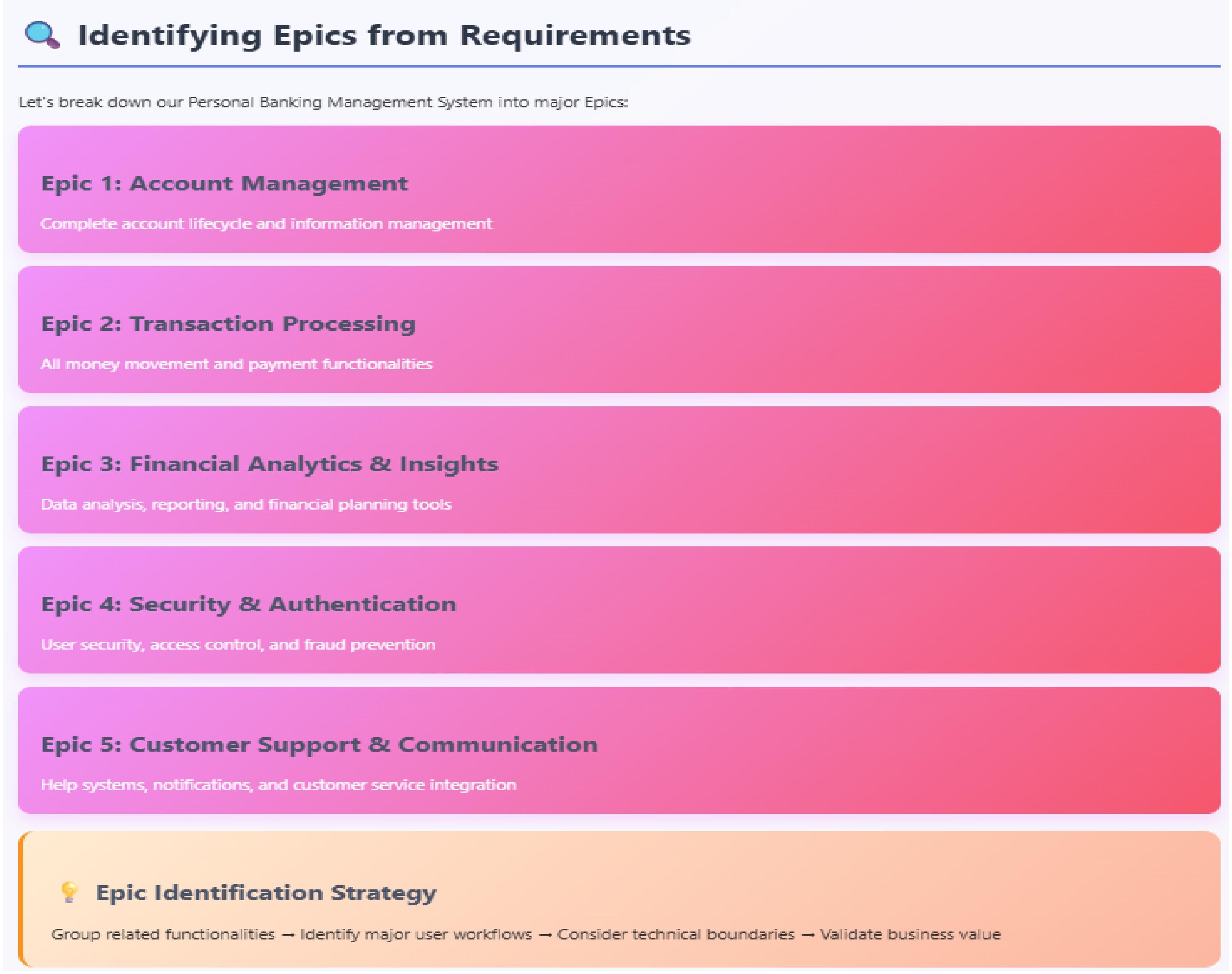
```
As a [type of user],
I want [some goal/functionality]
So that [some reason/value]
```

INVEST Criteria for Good Stories

Independent
Negotiable

Valuable
Estimable

Small
Testable



🔍 Identifying Epics from Requirements

Let's break down our Personal Banking Management System into major Epics:

Epic 1: Account Management

Complete account lifecycle and information management

Epic 2: Transaction Processing

All money movement and payment functionalities

Epic 3: Financial Analytics & Insights

Data analysis, reporting, and financial planning tools

Epic 4: Security & Authentication

User security, access control, and fraud prevention

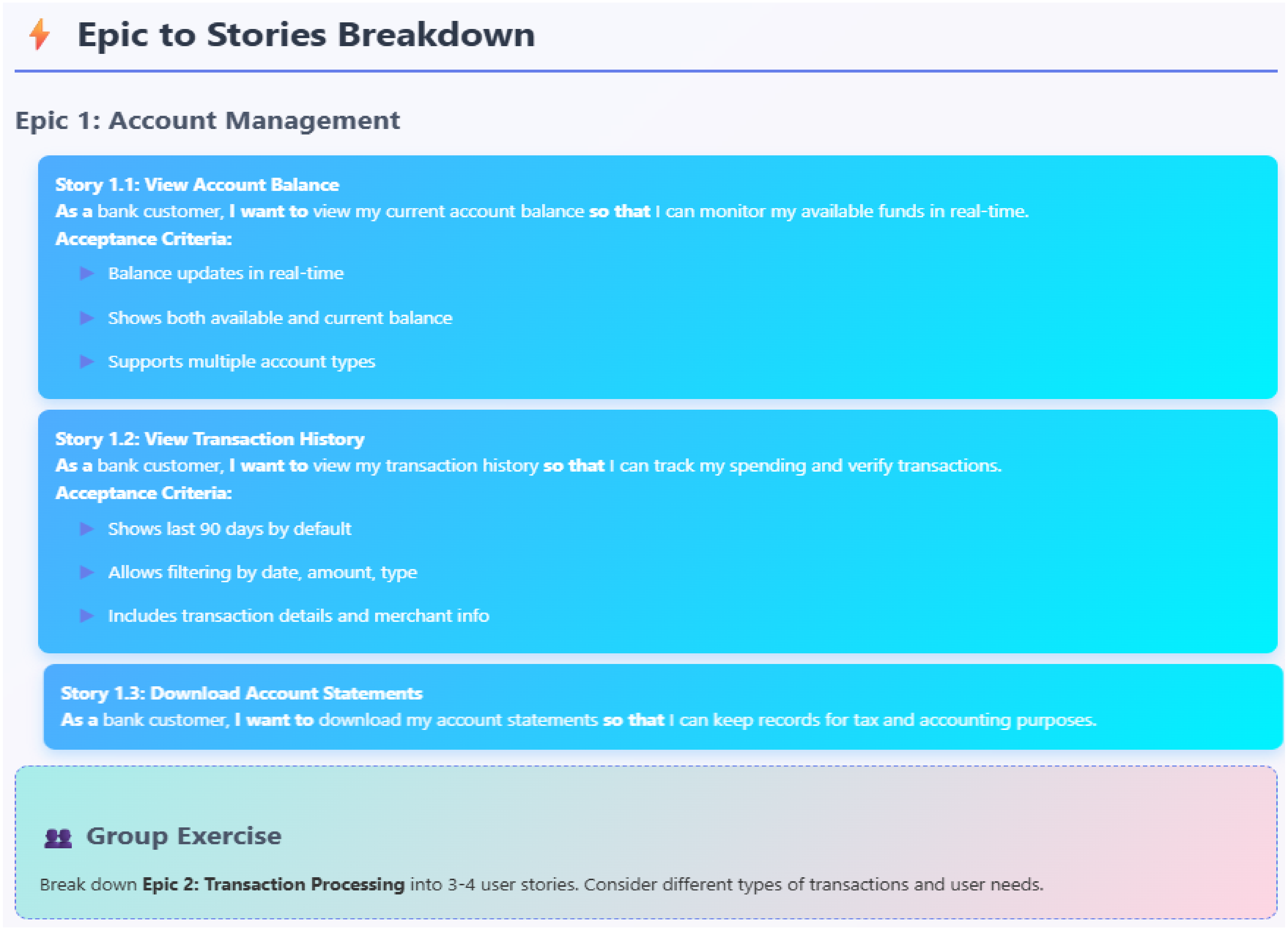
Epic 5: Customer Support & Communication

Help systems, notifications, and customer service integration

💡 Epic Identification Strategy

Group related functionalities → Identify major user workflows → Consider technical boundaries → Validate business value

Create a markdown file for your Epic and Stories



Epic to Stories Breakdown

Epic 1: Account Management

Story 1.1: View Account Balance
As a bank customer, **I want to** view my current account balance **so that** I can monitor my available funds in real-time.
Acceptance Criteria:

- ▶ Balance updates in real-time
- ▶ Shows both available and current balance
- ▶ Supports multiple account types

Story 1.2: View Transaction History
As a bank customer, **I want to** view my transaction history **so that** I can track my spending and verify transactions.
Acceptance Criteria:

- ▶ Shows last 90 days by default
- ▶ Allows filtering by date, amount, type
- ▶ Includes transaction details and merchant info

Story 1.3: Download Account Statements
As a bank customer, **I want to** download my account statements **so that** I can keep records for tax and accounting purposes.

Group Exercise

Break down **Epic 2: Transaction Processing** into 3-4 user stories. Consider different types of transactions and user needs.

Edit your markdown file for stories

🎉 Putting It All Together

Complete Epic-Story Hierarchy

We've transformed a complex banking system requirement into a structured, manageable backlog ready for development teams.

What We've Accomplished

📈 **Problem → Solution**

- ▶ Identified core banking challenges
- ▶ Defined system requirements
- ▶ Created solution roadmap

🏗️ **Structure → Execution**

- ▶ 5 Major Epics identified
- ▶ Stories with acceptance criteria
- ▶ Ready for sprint planning

🚀 Next Steps in Real Projects

1. ▶ **Story Estimation:** Size stories using story points
2. ▶ **Prioritization:** Order by business value and dependencies
3. ▶ **Sprint Planning:** Assign stories to development sprints
4. ▶ **Definition of Done:** Establish completion criteria
5. ▶ **Continuous Refinement:** Regular backlog grooming sessions

🎯 Final Challenge

Apply Your Learning: Choose a different domain (e.g., E-commerce, Healthcare, Education) and create 2-3 Epics with corresponding User Stories using the same approach we practiced today.

Defining MVP & Iterations

Building Products the Smart Way

From Problem to Progressive Solution

🎯 Training Objective

Learn to define and build a Minimum Viable Product (MVP) and plan iterative improvements using our Personal Banking Management System as a practical example.

📋 What You'll Learn

- Understanding MVP principles and benefits
- Problem identification and customer validation
- Feature prioritization and selection
- Iteration planning and execution strategy
- Success metrics and feedback loops

🚨 The Core Problem

Banking Industry Modernization Challenge

Scenario: Traditional banks face pressure to digitize services while competing with fintech startups that move faster and focus on customer experience.

Key Challenges:

- ▶ Long development cycles (18-24 months for new products)
- ▶ High upfront investment with uncertain market reception
- ▶ Complex regulatory requirements delaying launches
- ▶ Customer needs evolving faster than product delivery
- ▶ Risk of building features customers don't actually want
- ▶ Competition from agile fintech companies

💭 Student Discussion

Question: What happens when companies spend years building a "perfect" product? Share examples of products that failed because they took too long to launch.

📊 Industry Reality

Statistics: 70% of banking IT projects exceed their budget by 27% on average, and 17% go so badly that they threaten the existence of the company.

🚀 Understanding MVP

Minimum Viable Product (MVP)

The version of a product with just enough features to satisfy early customers and provide feedback for future development.

❌ Common Misconception

"Minimum Viable Product"
A barely functional, low-quality version of your product

- Missing core functionality
- Poor user experience
- Incomplete features

VS

✅ Actual Definition

"Minimum Viable Product"
The simplest version that delivers core value and enables learning

- Solves the main problem
- Delights early adopters
- Enables rapid feedback

🎯 MVP Principles

🔬 Learn
Validate assumptions quickly

📊 Measure
Gather real user data

🔄 Build
Iterate based on insights

🤔 Think About It

If you were building a car as MVP, would you start with a wheel, chassis, or skateboard? Why?

🔍 Customer Problem Deep Dive

Banking Customer Pain Points Analysis

Before defining our MVP, we need to understand and prioritize customer problems.

High Impact

High Frequency

🚀 MVP CORE
High Impact + High Frequency
- Account balance checking
- Fund transfers
- Transaction history

📈 Version 2
High Impact + Low Frequency
- Loan applications
- Investment tools
- Credit score monitoring

🍩 Nice to Have
Low Impact + High Frequency
- App themes
- Transaction categories
- Spending notifications

❌ Avoid
Low Impact + Low Frequency
- Advanced analytics
- Social features
- Gamification

👥 Group Exercise

Task: Interview 2-3 people about their banking frustrations. Plot these problems on the priority matrix above. Which quadrant do most problems fall into?

🏛️ Personal Banking System MVP

MVP Goal

Enable customers to perform essential banking tasks digitally, reducing branch dependency by 60% for routine transactions.

Core MVP Features (Must-Have)

🏧 Account Overview
Why: #1 customer need
View real-time balance across all accounts

💸 Fund Transfer
Why: Eliminates most branch visits
Transfer between own accounts and to other banks

📋 Transaction History
Why: Essential for financial tracking
Last 90 days with search and filter

🔒 Secure Login
Why: Non-negotiable for banking
Multi-factor authentication

💰 Bill Payment
Why: High-frequency customer need
Pay utilities and credit card bills

📱 Mobile Responsive
Why: 80% of banking is mobile
Works perfectly on all devices

🎯 Student Challenge

Question: Which feature would you remove if you had to cut the MVP scope by 30%? Defend your choice with customer impact reasoning.

⛔ What's NOT in Our MVP

The Art of Saying "No" (For Now)

Defining what NOT to build is as important as what to build. These features will come in later iterations.

📊 Advanced Analytics
Why Later: Complex to build, fewer users need it initially
Spending insights, budgeting tools, financial forecasting

🏠 Loan Management
Why Later: Regulatory complexity, lower frequency
Mortgage applications, personal loans, credit monitoring

💼 Investment Platform
Why Later: Different user segment, high complexity
Stock trading, mutual funds, portfolio management

🤖 AI Chat Support
Why Later: Resource-intensive, can use human support initially
Intelligent chatbot, automated problem resolution

🌍 Multi-currency
Why Later: Niche requirement, complex implementation
Foreign exchange, international transfers

🎮 Gamification
Why Later: Nice-to-have, focus on core value first
Savings challenges, achievement badges, social features

⚠️ Common MVP Mistakes

- ▶ **Feature Creep:** "Just one more small feature..."
- ▶ **Perfectionism:** "It's not ready until it's perfect"
- ▶ **Internal Bias:** "Our CEO really wants this feature"
- ▶ **Competitor Copying:** "But XYZ bank has this..."

📊 MVP Success Metrics

How Will We Know Our MVP Succeeds?

Clear metrics help us measure MVP success and guide iteration decisions.

🎯 Customer Success Metrics

- **Adoption Rate:** 40% of existing customers try the app within 3 months
- **Task Completion:** 85% successfully complete first transaction
- **Customer Satisfaction:** NPS score above 50
- **Usage Frequency:** 60% of users return within 7 days

📈 Business Impact Metrics

- **Branch Traffic:** 30% reduction in routine transaction visits
- **Call Center Load:** 25% decrease in balance/history inquiries
- **Cost Savings:** $2M annual operational cost reduction
- **Competitive Position:** Match top 3 fintech features

📏 Leading vs Lagging Indicators

🚀 **Leading (Early Signals)**	📊 **Lagging (Results)**
App downloads per day	Monthly active users
Registration completion rate	Branch visit reduction
First transaction attempt rate	Customer satisfaction scores
Daily active users	Revenue impact

🎯 Exercise

Define Success: If you were the product manager, what would be your #1 success metric for the MVP? Why did you choose this over others?

🔄 Planning Future Iterations 1/2

Iteration Strategy

Each iteration should build on MVP success while addressing the next most important customer problems.

🚀 MVP Launch (Months 1-3)

Focus: Core banking transactions

- ▶ Account overview and balance checking
- ▶ Fund transfers (own accounts + external)
- ▶ Transaction history and search
- ▶ Bill payment (top 5 utility companies)

Success Target: 10,000 active users, 70% task completion rate

📈 Version 2 (Months 4-6)

Focus: Enhanced user experience + financial insights

- ▶ Spending categorization and basic analytics
- ▶ Enhanced bill payment (all major providers)
- ▶ Push notifications for important transactions
- ▶ Customer support chat integration

Success Target: 25,000 active users, 80% task completion rate

🔄 Planning Future Iterations 2/2

🌟 Version 3 (Months 7-9)
Focus: Advanced financial management

- ▶ Budget creation and tracking tools
- ▶ Savings goals with progress tracking
- ▶ Credit score monitoring
- ▶ Loan pre-qualification tools

Success Target: 50,000 active users, NPS > 60

💼 Version 4 (Months 10-12)
Focus: Investment and wealth management

- ▶ Basic investment platform integration
- ▶ Financial advisor booking system
- ▶ Advanced analytics and forecasting
- ▶ Multi-currency support

Success Target: 100,000 active users, 90% customer retention

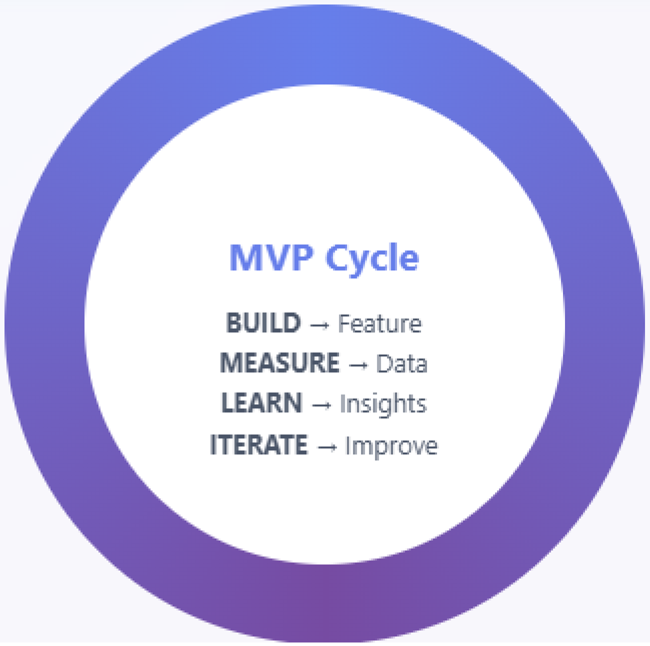
🛠️ Planning Exercise

Your Turn: Based on MVP feedback showing users struggle with budget management, would you prioritize budgeting tools in V2 or V3? What factors would influence your decision?

Build-Measure-Learn Cycle 1/2

The Continuous Improvement Engine

Each iteration follows the Build-Measure-Learn cycle to ensure we're building the right product.



🔄 Build-Measure-Learn Cycle 2/2

🏦 Banking MVP Example Cycle

🔨 BUILD
Fund transfer feature with 3-step process
Development: 2 weeks

📊 MEASURE
60% users abandon at step 2
Average completion: 3.2 minutes
Data collection: 2 weeks

🧠 LEARN
Too many verification fields
Users confused by step 2 UI
Analysis: 1 week

🔄 ITERATE
Reduce to 2 steps, improve UI clarity, add progress indicator

🎯 Scenario Exercise

Situation: Your MVP shows users love the balance checking feature (95% usage) but hate the transaction history (20% usage). What would you measure to understand why? What might you learn and build next?

🎯 Iteration Decision Framework

How to Prioritize Next Features

Not all feedback is equal. Use a systematic approach to decide what to build next.

Evaluation Criteria	Weight	Feature A: Budget Tools	Feature B: Investment Platform	Feature C: AI Chatbot
Customer Impact	40%	High (8/10)	Low (4/10)	Medium (6/10)
Development Effort	30%	Low (3/10)	High (9/10)	Medium (6/10)
Market Differentiation	20%	Medium (5/10)	High (8/10)	High (7/10)
Revenue Impact	10%	Medium (5/10)	High (8/10)	Low (3/10)
Weighted Score	-	**6.4/10**	**5.7/10**	**5.9/10**

🏆 Winner: Budget Tools for Version 2

Reasoning: Highest customer impact with lowest development effort creates the best value proposition for next iteration.

🤔 Critical Thinking

Discussion: The investment platform scored lower but might attract high-value customers. How would you factor in strategic long-term value vs immediate customer impact?

⚠️ Common Pitfalls & Best Practices

❌ Common Pitfalls

Feature Creep
Problem: Adding "just one more small feature" before launch
Result: Delayed launch, complex product, unclear value

Perfection Paralysis
Problem: Waiting for the "perfect" product before launching
Result: Missed market opportunities, wasted resources

Building for Everyone
Problem: Trying to satisfy all possible user segments
Result: Mediocre experience for everyone

Ignoring Metrics
Problem: Building next features based on gut feeling
Result: Wasted effort on low-impact features

✅ Best Practices

Start Smaller Than You Think
Approach: Cut your initial scope by 50%
Benefit: Faster time-to-market, clearer learning

Define Success Upfront
Approach: Set clear metrics before building
Benefit: Objective decision-making, focused effort

Talk to Customers Weekly
Approach: Regular user interviews during development
Benefit: Catch problems early, validate assumptions

Plan for Iterations
Approach: Design MVP architecture to support future features
Benefit: Smoother iteration development

💡 Pro Tips

- ▶ **The One Feature Rule:** If you can only build one feature, what would it be?
- ▶ **The Mom Test:** Can your mom understand the core value in 30 seconds?
- ▶ **The Concierge Approach:** Start by manually doing what software will eventually do
- ▶ **The 80/20 Rule:** Focus on features that solve 80% of user problems

🎉 Complete MVP & Iteration Strategy 1/2

From Problem to Progressive Solution

We've created a comprehensive MVP strategy that balances customer value, business objectives, and technical feasibility for our Personal Banking Management System.

📋 Complete Implementation Roadmap

MVP

Months 1-3: Foundation
Core Value: Essential digital banking transactions
Key Features: Balance checking, transfers, transaction history, bill pay
Success Metric: 60% branch visit reduction for routine transactions

V2

Months 4-6: Enhanced Experience
Core Value: Better UX + financial insights
Key Features: Spending analytics, notifications, enhanced bill pay
Success Metric: 80% task completion rate, NPS > 50

🎉 Complete MVP & Iteration Strategy 2/2

V3

Months 7-9: Financial Management
Core Value: Proactive financial planning
Key Features: Budget tools, savings goals, credit monitoring
Success Metric: 50% of users actively use budgeting features

🎯 Key Success Factors

🚀 Speed to Market
3-month MVP launch vs 18-month traditional approach

📊 Data-Driven Decisions
Every iteration based on customer feedback and usage data

🔄 Continuous Learning
Build-Measure-Learn cycle every 3 months

🎯 Final Challenge

Apply Your Learning: Choose a different domain (e.g., Healthcare, E-commerce, Education) and create your own MVP + iteration plan:

- ▶ Identify the core customer problem
- ▶ Define MVP scope (3-5 core features)
- ▶ Plan 3 future iterations
- ▶ Set success metrics for each phase
- ▶ Identify potential pitfalls and mitigation strategies

Remember: Start Small, Learn Fast, Iterate Smart!

Appendix