Campus

EXPENSE MANAGER

Group 4

Thành Phong

ViếtTiến

Quang Minh

Khánh Huy

Mạnh Đức

Meeting Agenda

01

CampusExpense Manager

02

What the Parties Want from the System

03

Estimated budget

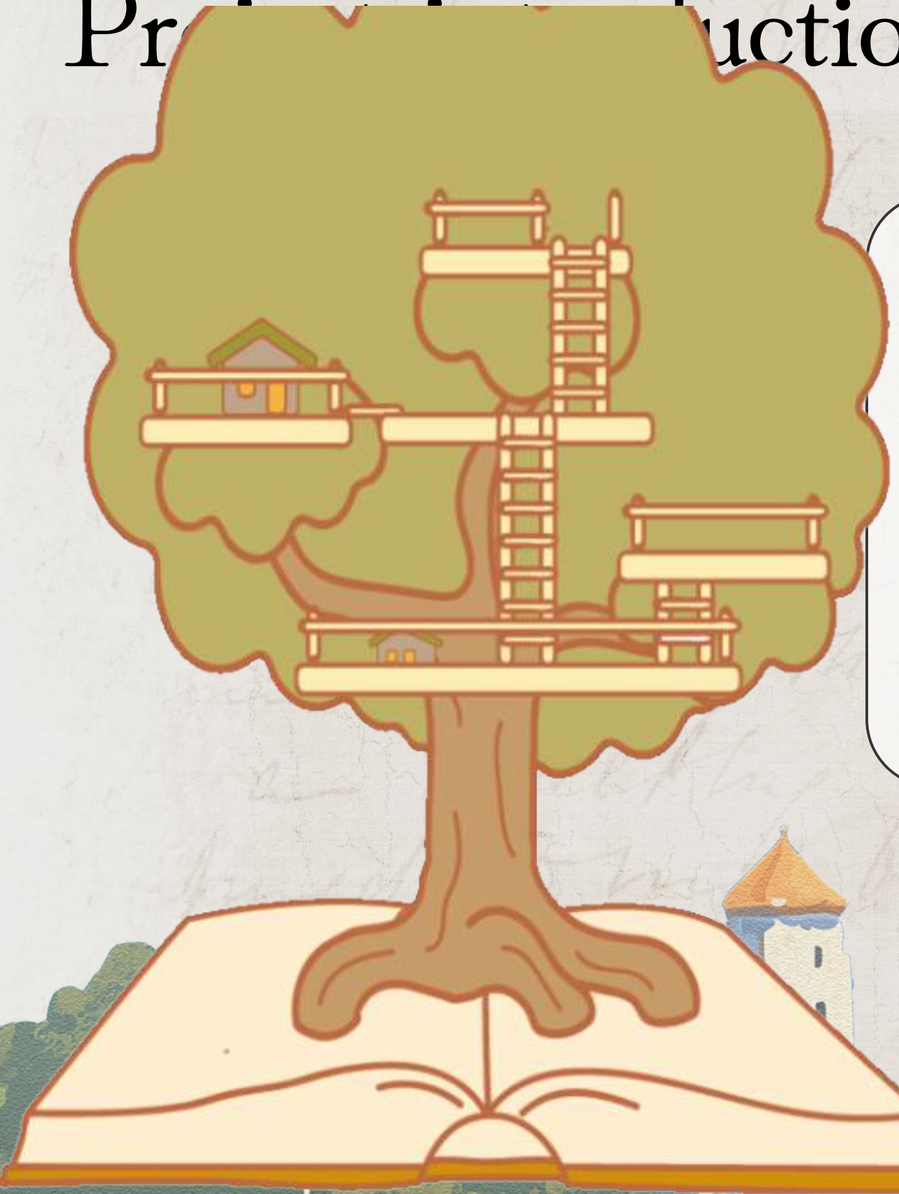
04

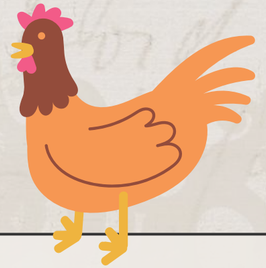
Difficulties of CampusExpense Manager Project

05

Functional Requirements or Non-functional requirements

Pr uction and goals





The CampusExpense Manager project is a mobile application specifically designed for university students to help them track and manage their expenses effectively. The application will assist students in controlling their personal budget through features such as entering expenses, categorizing expenses, setting budget alerts, and providing detailed reports on their financial situation. The project aims to provide a simple, easy-to-use tool for students to manage their personal finances in a smarter and more economical way.

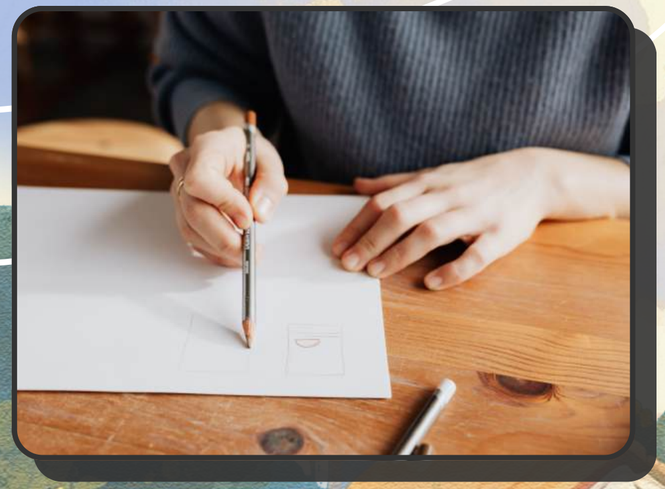
IDENTIFY STAKEHOLDERS

Investor

Primary users (Students)

Project Manager

Development Team



Investor

PHAM QUANG MINH

Role: Provides 10000$ to develop the project

Requirements:

- Clearly understand the project’s progress and its potential for profitability.
- Transparent reports on the utilization of funds.
- A clear roadmap for ROI (Return on Investment) and post-launch development plans.

Desires:

- Ensure the application meets the needs of students and enhances the school’s image.
- The application can scale up or create long-term value, such as strengthening the connection between students and the school.

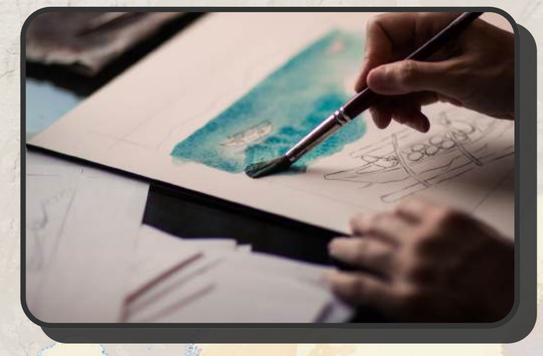
Primary User

STUDENT

Role: Students, staff, or residents on campus who need expense management.

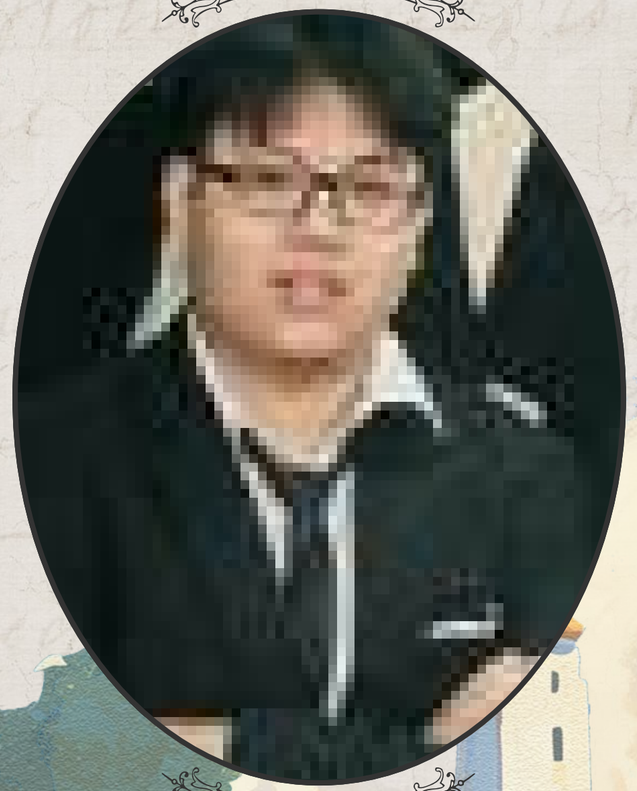
Requirements:

- User-friendly interface.
- Expense management features (tracking, categorizing, reporting).
- Support for payments and account management within the campus.
- High security and personal data protection.
- Essential features: Expense tracking, statistical reporting, budget alerts, and expense categorization.



Project manager

NGUYEN VIET TIEN



Role: Manages the entire development process and ensures the project meets its goals.

Requirements:

- Ensure the project stays on schedule and maintains high quality.
- Organize and coordinate tasks among team members.
- Regularly report to the investor and handle any issues that arise.
- Meet budget constraints and fulfill all requirements.
- Bridge teams and resolve problems during the development process.

Development team

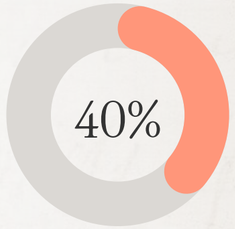
- NGUYỄN PHÚ THÀNH PHONG
- TRƯƠNG MẠNH ĐỨC
- NGUYỄN ĐỨC KHÁNH HUY

Role: Responsible for developing and maintaining the application.

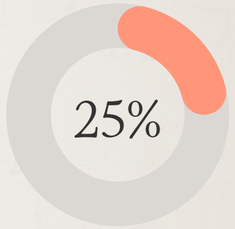
Requirements:

- Build an application with a user-friendly interface and features that meet user needs.
- Ensure the system is secure and performs efficiently.
- Test and debug the application to ensure smooth functionality.
- Support updates and expand features in the future.

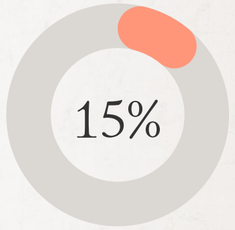
Estimated budget



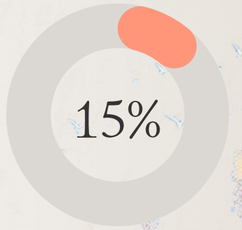
Personal Cost



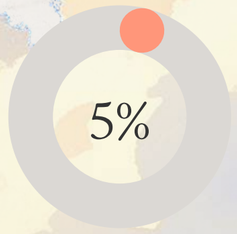
Equipment and sofware



Test and Security



Promotion and Deploy



Contigency Reserve

Given the investor's budget of $10,000, the allocation is proposed as follows:

Personnel Costs (40% - $4,000):

- Salaries and stipends for developers, project manager, and contributors.
- Covers the development phase, including coding, UI/UX design, and feature integration.

Equipment and Software Costs (25% - $2,500):

- Purchasing or subscribing to development tools, cloud services, and licenses.
- Hardware or peripherals required for testing or development.

Testing and Security Costs (15% - $1,500):

- Quality assurance (QA) processes, including usability testing and bug fixes.
- Implementation of security measures to protect user data and transactions.

Promotion and Deployment Costs (15% - $1,500):

- Marketing campaigns to raise awareness about the application.
- App store deployment fees (if applicable).

Contingency Reserve (5% - $500):

- Reserved for unforeseen expenses, ensuring the project remains on track despite unexpected costs.

Functional Requirement

USER ACCOUNTS AND AUTHENTICATION

- Users can create accounts with secure usernames and passwords.
- Implement a secure authentication system, allowing users to log in and access their spending data safely.

EXPENSE MANAGEMENT

- Users can add, edit, and categorize expenses (e.g., rent, food, transportation).
- Each expense entry includes a description, date, amount, and category.
- Users can attach receipts or images to expenses for better record-keeping (optional enhancement).

BUDGET MANAGEMENT

- Personal budgets: Users can set budgets for individual expense categories to maintain financial control.
- Budget alerts: Notifications are sent when users approach or exceed their budget limits, helping them adjust their spending in time.

RECURRING EXPENSES

- Users can add recurring expenses (e.g., monthly rent) with specified start and end dates.
- These expenses are automatically added to the user's monthly budget.

Functional Requirement

Expense Summaries and Reports

- Provide a monthly summary of expenses, including:
  - Total spending.
  - Remaining budget.
  - Breakdown by category.
- Generate detailed reports for specific time periods (e.g., monthly, yearly).
  - Include a breakdown of spending by category and visual aids like charts.

Expense Summaries and Reports

- Allow users to view expense trends over time, helping them understand and optimize their financial habits.
- Compare spending trends across different periods (e.g., this month vs. last month).

Spending Notifications

- Notifications for:
  - Approaching or exceeding budget limits in specific categories.
  - Upcoming recurring payments.
  - Weekly or monthly spending summaries.

Non-Function

Maintainability

- Ease of Maintenance: Use a well-structured and clearly documented codebase to facilitate future maintenance, upgrades, and feature expansion.

Design and User Experience

- Friendly Design: The application should have an intuitive and easy-to-navigate user interface (UI), allowing users to quickly become familiar with its features.
- User Experience (UX): Ensure a seamless and enjoyable experience across all functionalities, from registration and login to expense recording and report viewing.

Customer Support

- Online Support: Provide an efficient online support channel to address user inquiries and resolve issues related to the application quickly.

Non-Function

Security

- Protection of Personal Data: Encrypt and securely store all user data, including financial and personal information, to safeguard against cyber threats.
- Secure Authentication: Implement multi-factor authentication (2FA) to strengthen account security and reduce the risk of unauthorized access.

Platform Compatibility

- Multi-platform Support: Ensure smooth functionality across multiple devices and operating systems, including Android and iOS, to cater to a broad user base.
- Compatibility with Latest OS Versions: Maintain compatibility with the newest versions of operating systems and a variety of device configurations.

Difficulties of Projectao

1. Security and Data Privacy

Challenge: Protecting sensitive user data (e.g., personal details, financial records) from cyberattacks and unauthorized access.

Solution:

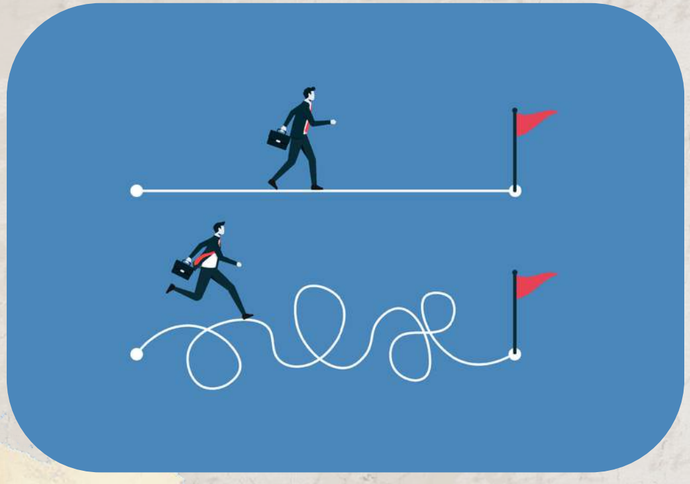
- Implement robust encryption protocols for data storage and transmission.
- Use secure authentication mechanisms such as multi-factor authentication (2FA).
- Regularly conduct security audits and penetration testing to identify vulnerabilities.

2. Performance Optimization

Challenge: Ensuring the application performs efficiently, especially during peak usage when multiple users access it simultaneously.

Solution:

- Optimize the backend architecture for scalability using cloud-based services.
- Implement efficient database queries and caching mechanisms.
- Conduct stress testing to simulate heavy traffic and refine performance bottlenecks.



Difficul of project

4. Budget Constraints

Challenge: Managing a limited budget while delivering a high-quality product with essential features.

Solution:

- Prioritize core features for the initial release and plan for additional functionalities in future updates.
- Use open-source tools and libraries to reduce costs.
- Allocate a contingency reserve to cover unforeseen expenses.

5. User Engagement and Adoption

Challenge: Encouraging users to adopt the app and use it consistently.

Solution:

- Implement an intuitive and user-friendly interface to reduce the learning curve.
- Include interactive onboarding tutorials to guide new users.
- Use push notifications and reminders to keep users engaged.

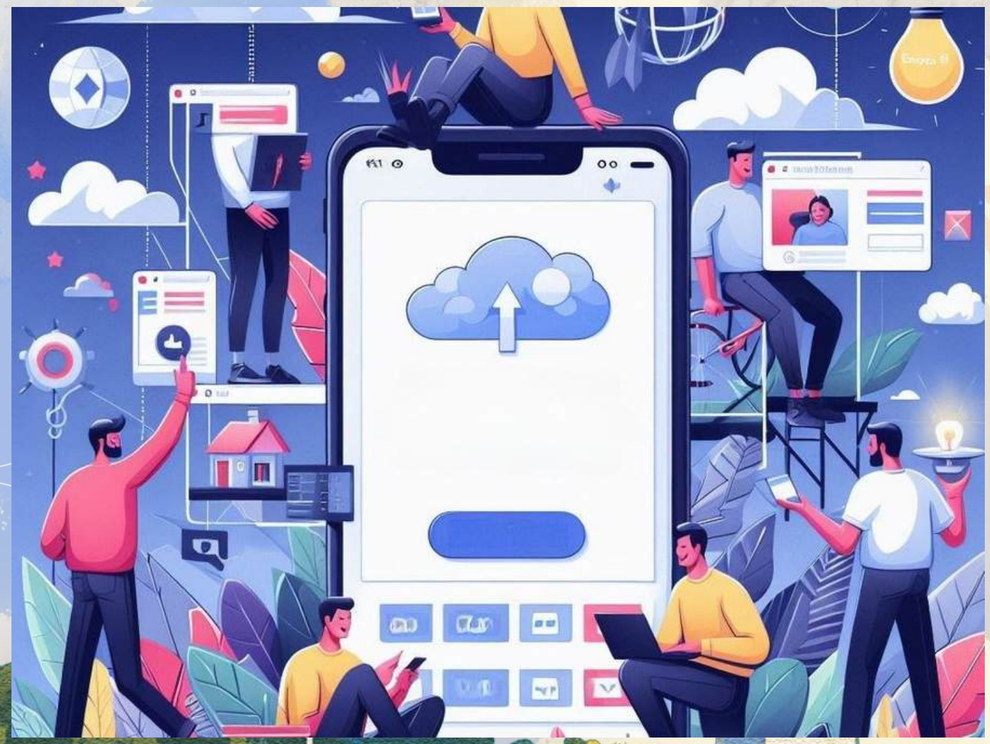
6. Maintenance and Scalability

Challenge: Ensuring the app remains maintainable and scalable as the user base grows or new features are introduced.

Solution:

- Adopt modular and well-documented coding practices to ease future upgrades.
- Use version control systems to manage updates efficiently.
- Plan for regular app updates to fix bugs and add new features based on user feedback.

Comparison Between Mobile App Programming And Desktop App Programming



1. Challenges

Aspect	Mobile Apps	Desktop Apps
Compatibility	Must support multiple operating systems (Android, iOS) and various screen sizes and devices.	Fewer OS versions (Windows, macOS, Linux) but must adapt to diverse hardware configurations.
Performance	Needs to optimize performance for devices with limited hardware capabilities and battery life.	Typically less constrained by hardware but may require optimization for resource-heavy tasks.
Security	High risks like data leaks, API attacks, or fake apps in app stores.	Vulnerable to malware or unauthorized distribution of software.
Updates and Maintenance	Updates depend on app store approvals, which can delay rollouts.	Updates are managed directly or via built-in systems, with fewer third-party dependencies.
Network Dependency	Often reliant on internet connectivity, especially for cloud features.	Can work offline more easily but may require data synchronization periodically.
User Experience	Must optimize for small screens, touch gestures, and simple, intuitive design.	Larger screens allow for more complex interfaces and advanced features.

Comparison Between Mobile App Programming And Desktop App Programming

2. Functional Requirements

Mobile Apps	Desktop Apps
Account Management: User login, registration, authentication.	Similar but may include multi-user management on the same device.
Notifications: Push notifications for user engagement.	Notifications are typically system-based or sent via email.
Data Synchronization: Often requires cloud-based syncing.	May provide options for local storage or manual synchronization.
Resource Management: Optimized for battery and RAM usage.	Focuses on leveraging more robust system resources efficiently.
Hardware Integration: Access to GPS, camera, sensors.	Typically supports more advanced peripherals like GPUs or specialized input devices.

3. Non-Functional Requirements

Aspect	Mobile Apps	Desktop Apps
Security	Requires encrypted data storage and secure transmission, often with 2FA for added security.	Needs robust antivirus and firewall protection against unauthorized access.
Performance	Optimized for smooth operation on low-end devices.	Designed for handling resource-intensive tasks like rendering or large data processing.
Scalability	Must handle a growing user base and support easy addition of new features.	Needs to scale across different hardware configurations without compromising performance.
Maintainability	Codebase should support cross-platform development (e.g., Android/iOS) and future updates.	Less cross-platform complexity but still requires a modular codebase for updates and bug fixes.
Compatibility	Must ensure compatibility with multiple OS versions and device types.	Usually limited to fewer platforms but must address backward compatibility with older OS versions.
User Experience (UX)	Simplicity and speed are crucial, optimized for touch input and small screens.	Allows for a more feature-rich and complex interface with advanced interactions.

Thank You

Group 4