Functional Requirements (User Stories)

ID	I WANT	SO THAT I AM ABLE TO	Acceptance Criteria
UH01	View My Balance	Know how much I have in my digital wallet	● It should display the Wallet code, name, and current balance.
UH02	Log in to my personal session	Maintain privacy when conducting my financial activities	● The 'username' and 'password' must be unique for each client.
UH03	Register	Manage my finances	● The system must provide a registration form with the following customer information (Name, Last Name, ID, Phone Number, Email, Address). ● The system must validate the information entered by the user to ensure that the format and length requirements for each field are met. ● Clear and descriptive error messages must be displayed if incorrect or missing data is entered. ● The system must generate a unique identifier (a customer number) for each registered customer.
UH04	Add a record to my transactions	Have control over my financial flow	● The system must provide an interface that allows the user to add a new transaction to their transaction record. ● There should be a dedicated form or screen to enter the transaction details (amount, date, category, optional description). ● The system must allow the user to select the type of transaction they wish to add (category) (e.g., income, expense, transfer, etc.). ● There should be predefined options for different types of transactions, and the user should

			also have the option to add a custom type if necessary.
UH05	Generate a report of my transactions	Carry out an analysis of my income, expenses, among others.	● The report format should include clear columns displaying transaction details such as date, transaction type, amount, description, etc. ● The user should be able to choose how the data is grouped and sorted in the report, for example, by date, by transaction type, by category, etc. ● The user should be able to choose how the data is grouped and sorted in the report, for example, by date, by transaction type, by category, etc.
UH06	Characterize my transaction report by category	Carry out a specific and more detailed analysis of each category	● The system must allow the user to select the transaction categories they wish to include in the report. ● Predefined options for the most common categories must be available, as well as the option for custom categories if necessary. ● The system must group transactions in the report according to the categories selected by the user. ● Each group should display the total number of transactions and the total amount associated with that category.

1. **User Registration:** Allow users to register in the application by providing basic information such as name, email address, and password.

2. **Login:** Allow users to log in to the application with their previously registered credentials.

3. **Transaction Recording:** Allow users to record financial transactions, such as income, expenses, transfers between accounts, bill payments, etc.

4. **Transaction Categorization:** Allow users to categorize recorded transactions for better tracking and analysis of expenses and income.

5. **Balance Display:** Show users the current balance of their financial accounts and their available balance after accounting for recorded transactions.

6. **Report Generation:** Allow users to generate detailed reports on their financial transactions, including summaries of income and expenses by time period, trend charts, etc.

Requerimientos No Funcionales

1. **Usability:** The application should be intuitive and easy to use, with a clear and user-friendly interface.
    a. Metric: Percentage of users who successfully complete the registration/login process on the first attempt.
    b. Goal: More than 90% of users should complete the registration/login process correctly on the first attempt.

2. **Performance:** The application should be fast and efficient, with minimal loading times and the ability to handle multiple users simultaneously.
    a. Metric: Average load time of the application's main page.
    b. Goal: Less than 3 seconds of average load time for the main page.

3. **Availability:** The application should be available 24/7, with minimal scheduled downtime for maintenance.
    a. Metric: Planned and unplanned downtime over a specified period.
    b. Goal: Less than 1 hour of planned and unplanned downtime per month.

4. **Scalability:** The application should be able to scale horizontally to handle an increase in the number of users and transactions without performance degradation.
    a. Metric: Server response time under different user loads.
    b. Goal: Server response time should stay below 500 ms for all simulated user loads.

5. **Security:** The application should protect users' financial data and ensure its confidentiality, integrity, and availability.
    a. Metric: Number of unauthorized access attempts blocked by the security system.
    b. Goal: All unauthorized access attempts should be blocked and logged, with a 100% detection rate.

6. **Compatibility:** The application should be compatible with a variety of devices and web browsers to ensure a consistent experience for all users.
    a. Metric: Percentage of users accessing the application from different devices and browsers.
    b. Goal: The application should be compatible with at least 95% of the most used devices and browsers according to usage statistics.

7. **Regulatory Compliance:** The application must comply with relevant financial and privacy regulations, such as GDPR, PCI-DSS, etc.
    a. Metric: Results of regulatory compliance audits (e.g., security audits, GDPR compliance, PCI-DSS compliance, etc.).
    b. Goal: The application must pass all regulatory compliance audits without significant findings.

8. **Error Handling:** The application must handle errors appropriately, providing clear error messages and securely processing transactions even in the event of system failure.
    a. Metric: Average system response time to critical errors (e.g., server errors, database errors).
    b. Goal: The average system response time to critical errors should be less than 1 second.

ARCHITECTURE

The decision taken by the team was to use microservices as the architecture to implement, for the following reasons:

- Scalability: Microservices allow specific parts of the application to scale independently, which can be beneficial in a financial application where certain functionalities may experience spikes in load while others do not.

- Agile Deployment: By dividing the application into smaller, manageable microservices, teams can work more autonomously and quickly in developing, testing, and deploying new features.

- Resilience: Microservices can isolate failures, meaning that a failure in one microservice will not affect others, thereby improving the overall system's resilience.

- Technological Flexibility: Each microservice can be developed and deployed using the most suitable technology for its specific purpose, enabling greater technological flexibility and adaptability.

The architecture in the C4 model is found in the attached document.

TEST PLAN

GENERAL

FinanceMe serves as a solution for personal finance management that is not included in financial products such as savings accounts in banks or other financial institutions. It serves to have a better understanding of the economic flow (including income and expenses) and empower each user to analyze how their capital moves and better manage their resources.

LIMITS (It's dynamic, never fixed. Continuous learning)

This section describes the activities that will be carried out in the project.

SCOPE

Covered	Description	Not Covered	Description
Functional Testing	• **Modules:** User registration, login, transaction registration, transaction categorization, balance visualization, and report generation. • **Techniques:** Unit, integration, and system testing. • **Priority:** High.		
Non-Functional Testing	• **Modules:** All modules of the application. • **Techniques:** Load and performance testing. • **Priority:** High.		
Usability Testing	• **Modules:** All modules of the application. • **Techniques:** User feedback collection and satisfaction evaluation.		
Entry and Exit Criteria	Established when all functional and non-functional tests pass satisfactorily.		
Data Quality	Ensure that test data reflects real-world scenarios.		
Unit Testing	Unit tests will be conducted for each microservice.		
		Installation	Installation

		and Migration	and migration to another development environment will not be evaluated.
		White Box Testing	Detailed evaluations of the internal structure will not be conducted. Tests will focus on overall functionality, usability, performance, and other external aspects of the application.

If any of the covered tests are not conducted, they will automatically be considered as not covered, and it will be understood that they were initially contemplated but it was decided not to conduct such tests.

WORK TEAM ORGANIZATION

- Workflow:
    1. Product Owner provides documentation such as requirements, test data, etc.
    2. The Devs team provides software builds and deliverables.
    3. The DevOps team supports the testing environment by executing test cases, and if a failure is found, it is reported to the tester for analysis.
    4. The DevOps team provides support to the testing environment by executing test cases, and if a failure is found, it is reported to the tester for analysis.
    5. Finally, the testing team's activities include creating the test plan, creating test cases, and updating test cases if there are changes in requirements.

After completing the workflow, the testing team provides the Product Owner with a report of test cases, reports defects and software failures to the Devs team, and provides detailed test results to the DevOps team for adjustments in the testing environment.

QUALITY RISKS

Risks reported by the PO or the Devs team that have already been identified, so the testing team should take this into account when developing their work to mitigate these risks. These risks are rated from 1 to 5, with 5 being very high risk and 1 being very low risk.

In this case, it would be:

Risk	**Mitigation**
Failures in microservices integration. **Risk Priority: 5**	Conduct comprehensive integration testing.
Inconsistency in transaction categorization due to different user interpretations. **Risk Priority: 3**	Provide a clear interface and examples to help users understand and follow a consistent approach in categorization.
Deviation in main page loading times exceeding established limits during user peaks or due to simultaneous transaction loading. **Risk Priority: 5**	Conduct comprehensive load and performance testing to identify and address potential bottlenecks.

MILESTONE CALENDAR

Moments are proposed to discuss activities within the testing team. The Product Owner or Devs team leader plans this calendar, and meetings are documented from the moment the product is conceptualized until it is ready for end users. Meetings have a start date and an end date for any reinforcement and/or continuous evaluation of a process

TRANSITIONS

Criteria to be considered when transitioning from one testing level to another.

ENTRY CRITERIA

- All functional and non-functional tests must have been passed.
- The DevOps team must have confirmed that the deliverable is functional in the testing environment.
- A comprehensive assessment of test coverage must be performed, ensuring that sufficient tests have been conducted in all aspects of the application.
- Before advancing to the next phase, formal approval is required from the Devs team indicating that they have satisfactorily addressed all issues and defects identified during testing.

EXIT CRITERIA

-
- Report of test cases, defect report, and detailed results delivered to the respective responsible parties by the testing team.

- There are no failures, crashes, or processes unexpectedly terminated (exceeding 5% of the tests).
- The Product Owner approves that the product defined by the latest system tests reasonably meets user expectations.
- The testing team has executed the planned tests for each environment.

TEST ENVIRONMENT CONFIGURATION

The system will operate on various internet browsers and operating system combinations. It is noted that the responsible parties for configuring these environments will be the DevOps team, but the test environment configuration remains under the control of the testing team leader.

Internet Browser	**OS**
Chrome	Windows 10
Brave	Windows 10
Opera	Windows 11
Chrome	Windows 11
Brave	Windows 11