
VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Software Architecture

# Personal finances storing and analyzing website

Done by:
Oleksii Morozov

Supervisor:
Teach. Asst. Andrius Vytautas Misiukas Misiūnas

Vilnius
2025

# Contents

**Introduction** **3**

**1 Planned functionality** **3**

**2 Technologies** **3**

**3 Non-functional requirements** **4**

**4 UML diagrams** **4**

4.1 Class diagram . . . . . 4

4.2 Use-Case diagram . . . . . 4

**5 Database** **5**

# Introduction

The essence of the project is to effectively control personal financial flows by entering data into the system. Each user will enter data on received and spent money to view the overall picture. The analysis will be available with total expenses, expenses for a certain period (day, week, month, year), saved money, deferred money, borrowed money, creating goals for the month, and budget distribution.

# 1 Planned functionality

- **I iteration:**
    1. User registration and authentication;
    2. Adding financial transactions.
- **II iteration:**
    1. Transaction Categorization;
    2. Filtering transactions;
    3. CRUD of categories.
- **III iteration:**
    1. Financial summary dashboard;
    2. Setting budget limits, and targets.
- **IV iteration:**
    1. Recurring transactions;
    2. Expense Predictions.

# 2 Technologies

The Personal Finances Storing and Analyzing Website will use Python with the Django framework for both backend and frontend development. Django will handle user authentication, transaction management, and business logic while also rendering dynamic HTML templates using Django's built-in Template Engine for the frontend. MySQL will serve as the database management system, ensuring efficient storage and retrieval of financial data. The application will be hosted on an Apache web server, managing incoming requests and serving the application securely. For deployment, Virtual Machines (VMs) will be used, providing scalability, system isolation, and efficient resource allocation. This technology stack ensures a reliable, secure, and high-performance platform for managing personal finances.

# 3 Non-functional requirements

The Personal Finances Storing and Analyzing Website must meet the following non-functional requirements to ensure performance, security, and usability:

- Performance & Scalability – Support 1,000+ concurrent users, response time 2s, optimized queries and caching.
- Security – Encrypt data, use hashed passwords, protect against SQL injection, XSS, CSRF, and support MFA.
- Availability & Reliability – Ensure 99.9% uptime, daily automated backups, and automatic recovery.
- Usability & Accessibility – Provide a responsive UI, follow WCAG 2.1, and offer an intuitive dashboard.
- Maintainability & Extensibility – Use Django best practices, modular design, and maintain clear documentation.

# 4 UML diagrams

## 4.1 Class diagram

The class diagram represents a financial tracking system where users can manage their income and expenses.

Each User has an account and can track multiple financial Records, which represent transactions such as income or spending. Users can sign up, log in, and access their financial records to monitor their budget and spending habits. Each record belongs to a specific user and includes details like the amount, category (e.g., food, rent), and transaction type (income or expense). The system allows users to add, update, and delete records, as well as view summaries of their total expenses and income over a given period. This structure ensures that users can effectively track and analyze their financial activities in a simple and organized way.

## 4.2 Use-Case diagram

The use case diagram illustrates how users interact with the Finances Storing and Analyzing Website to manage their financial transactions.

Users can sign up to create an account and sign in to access their financial records. They can add, edit, categorize, and delete records to track income and expenses, with the option to set transactions as recurring. The system allows users to filter records and view financial analysis, helping them monitor spending habits and set budget limits. Additionally, users can access expense predictions based on past data, providing insights for better financial planning. The diagram ensures a smooth and intuitive experience for managing personal finances effectively.

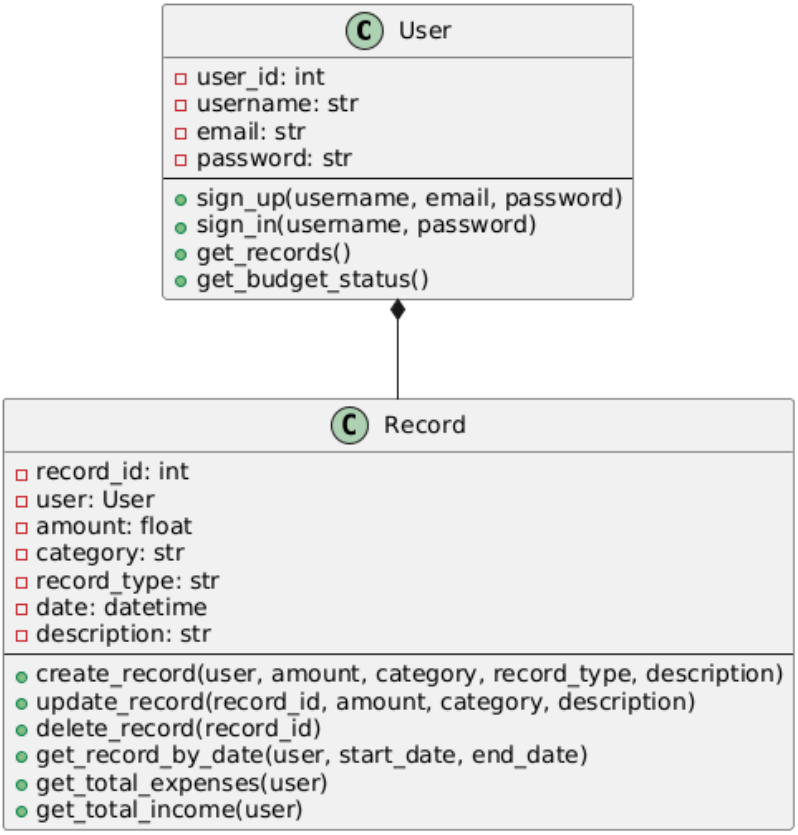


Figure 1. Class diagram

# 5 Database

The Entity-Relationship (ER) Diagram represents the database structure for a financial management system, illustrating the relationships between entities. The User entity contains attributes such as ID, username, password, and email, which uniquely identify and authenticate each user in the system. A User has a one-to-many (1:M) relationship with the *Financial_Record* entity, indicating that a single user can manage multiple financial records. The *Financial_Record* entity, uniquely identified by RID (Record ID), stores essential transactional attributes, including *R_type* (record type: income or expense), description, amount, category, and date. The Controls relationship ensures that users can manage their financial data effectively. This ER diagram provides a structured representation of how financial transactions are stored, categorized, and linked to users within the database, ensuring data integrity and efficient retrieval.

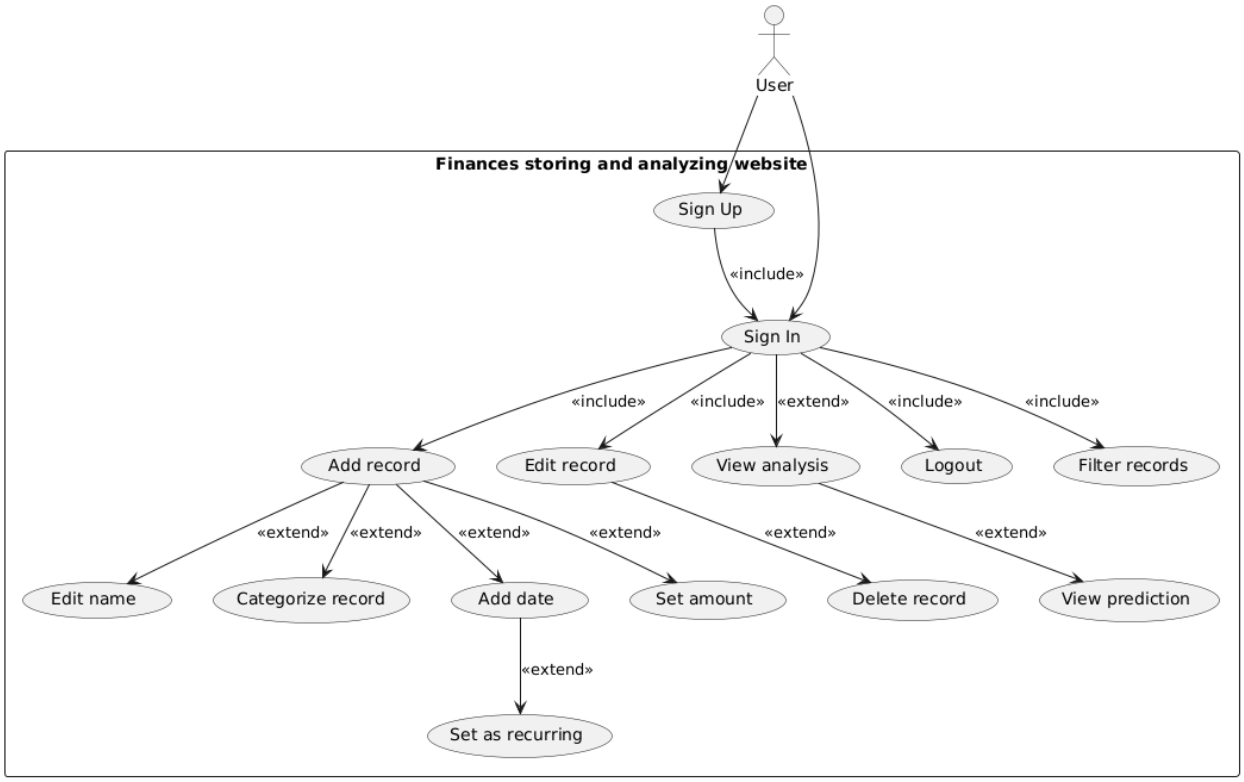


Figure 2. Use case diagram

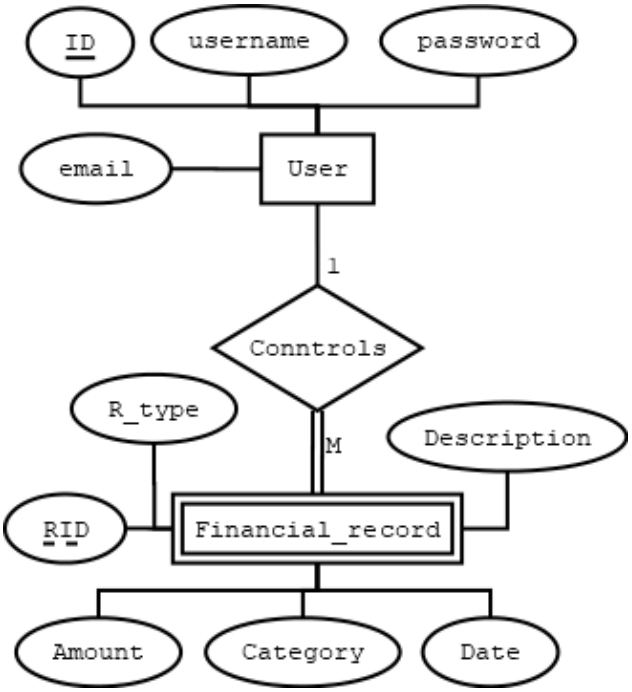


Figure 3. ER diagram