Project Kit

Title of the Project

Interest Calculation Automation System

Abstract of the Project

The Interest Calculation Automation System automates daily interest calculations for retail bank accounts with positive balances. The system dynamically adjusts for deposits and withdrawals, stores cumulative interest daily, and credits the cumulative amount to customers' accounts at the end of each month. With increasing customer demand for accurate and efficient banking operations, this system aims to streamline interest calculation processes. Automating the daily interest calculation ensures accuracy, reduces human errors, and enhances operational efficiency in retail banking.

Generic keyword:

Interest Calculation, Banking Automation, Daily Interest Calculation, Retail Banking System, Financial Software, Account Management, Deposit and Withdrawal, Interest Crediting, Cumulative Interest, Backend API, Real-time Balance Update, Transaction Management, Customer Portal, Admin Dashboard, Secure Financial Transactions

Specific Technology keywords:

HTML, CSS, React.Js, MongoDb, Node.Js, Web Application, Database Management.

Functional Components:

1. **User Registration and Authentication**
   - Allows users (customers and admins) to securely create accounts and log in to manage bank accounts and access financial services**.**
2. **Account Management**
   - Users can view, update, and manage account details such as balances, transaction history, and personal information.
3. **Daily Interest Calculation**
   - Automatically calculates daily interest for accounts with positive balances, adjusted for deposits and withdrawals**.**
4. **Cumulative Interest Storage**
   - Stores cumulative daily interest for each account, which updates daily based on the current balance.
5. **Monthly Interest Crediting**
   - Credits the cumulative interest to the customer's account at the end of each month.
6. **Transaction Management**
   - Manages user deposits and withdrawals, reflecting changes in account balances in real-time**.**
7. **Admin Dashboard**

- Provides admins with tools to manage customer accounts, set interest rates, and review system performance.

8. **Expense and Interest Reporting**
    - Allows customers to view detailed reports of their account balances, interest earned, and transaction history over specific periods.
9. **Interest Rate Configuration**
    - Admins can set and update interest rates for different types of accounts.
10. **Security and Data Protection**
    - Ensures the secure storage of user data, transactions, and account balances with encryption and authentication protocols.

Functionality

Users of the system:

Customers:

Role: Access their accounts, manage expenses, set budgets, and view financial reports.

Administrators:

Role: Manage user accounts, monitor expense activities, generate reports, and ensure system security.

Core Functionalities:

• User registration and authentication for both customers and administrators to access the system securely.
• Management of user profiles, including personal account details and transaction history.
• Automated daily interest calculation based on the account's positive balance, adjusted for deposits and withdrawals.
• Real-time transaction management for deposits and withdrawals, updating account balances dynamically.
• Cumulative interest storage that updates daily and keeps track of total interest earned for each account.
• Monthly interest crediting, transferring the accumulated interest to customer accounts at the end of the month.
• Administrative tools for managing customer accounts, setting interest rates, and monitoring the system's performance.
• Generation of detailed financial reports, including interest earned, account balances, and transaction history.
• Robust security measures, including encryption, to ensure the protection of user data and secure communication.

Steps to start-off the project:

The following steps will be helpful to start off the project –

1. Gain a solid understanding of the necessary technologies.
2. Research personal finance management and user behaviour.
3. Define the different types of users (customers and administrators) and their roles.

4. Ensure the interface is user-friendly and intuitive.
5. Maintain a consistent UI/UX design with professional visuals and coherent navigation.

Requirements

Hardware Requirements:

Number	Description	Alternative (if Available)
1.	Minimum requirements- Processor, x86-64 bit CPU	
2.	Ram - 4Gb, Disk Space - 5Gb.	

Software Requirements:

Numbers	Descriptions	Alternatives (if Available)
1.	Client on Intranet - User Interface, Windows OS	
2.	Development Environment - IntelliJ/VS Code, Spring Boot, Angular, MySQL, Windows OS	

Manpower requirements:

2 to 3 students can complete this in 4 – 6 months if they work fulltime on it.

Milestones and Timelines

Number	Milestonename	Milestone Description	Timeline Week no. From the start ofthe project	Remarks
1.	Requirements Specification	Complete specification of the system with assumptions. Write a document and present it.	2-3	Attempt to add more relevant functionalities beyond the basic requirements.
2.	Technology familiarization	Understand the technologies needed to implement the project.	4	Present the understanding with a focus on practical application rather than theory.
3.	Database creation	Create a database with users and categories for expenses.	5-6	Finalize the database to ensure smooth development and testing.
4.	High-Level and Detailed Design	Map each requirement to a scenario and create flowcharts or pseudocode to handle them.	7-8	Design should be comprehensive and cover all specified requirements.

5.	Front-End Implementation	Implement the login system and initial expense tracker interface.	9-10	Begin working on a test plan for the system during this milestone.
6.	Front-End and Database Integration	Connect the front-end with the database and ensure expense data is stored correctly.	11-12	Ensure the system is ready for integration testing at this point.
7.	Integration Testing	Thoroughly test the system using the test plan created earlier.	13-14	Another 2 weeks should be there to handle any issues found duringtesting of the system. After that, the final demo canbe arranged.
8.	Final Review	Ensure all requirements are met and prepare for the final demo	15-16	Confirm that all system requirements have been fulfilled or document reasons for any gaps

Guidelines and Reference:

- Object Oriented Modelling and Design with UML- Michael Blaha, Jams Rumbaugh. Software Engineering, Seventh edition Ian Sommerville.
- ReactJs Document https://react.dev/reference/react
- NodeJs https://nodejs.org/docs/latest/api/
- Wikipedia - www.wikipedia.com
- Database Management Systems-Navathe. Complete Reference - J2EE - Keogh.