Bankist Project Technical Documentation

1. Overview

This project is a demonstration of a mock banking application used to show web design and technical skills through JavaScript, HTML, and CSS. Its scope is the UX/UI capabilities through front-end techniques while using a basic mock database as its backend for bank information. This application navigates through what a banking site would have/suggest for the customer if they were to use our services. It covers a landing page, a dashboard with mock data, a sign-up page, and a calculating savings page.

This section provides a summary of the software project, including its purpose, scope, and key functionalities. It should offer readers a high-level understanding of what the software does and why it exists.

1.1 Project Purpose

This software project demonstrates a linear example of what a banking site could have and use as its UI/UX design. Its key purpose is to demonstrate a simplistic design through minimalism to avoid complexity over simple banking tasks. The application showcases modern web development practices including responsive design, form validation, user authentication, and dynamic content generation.

Describe the main goal of the software and the problem it solves.

1.2 Scope

What is included in the scope of this project are front-end techniques like a navigation bar, dashboard for mock information, basic savings calculation and a basic sign-up page when creating an account. What is excluded is a back-end to house any database information (accounts, transactions, etc.).

Included in scope**:**

- Front-end techniques including responsive navigation bar
- Interactive dashboard displaying mock banking data
- Compound interest savings calculator with year-by-year breakdown
- User registration with password validation
- Client-side authentication system
- Responsive design for mobile, tablet, and desktop devices
- Smooth scroll animations and transitions
- Modal-based login interface

Excluded from scope**:**

- Backend server infrastructure
- Persistent database storage
- Real financial transactions
- Payment processing integration
- User session management with cookies/tokens
- Email verification

- Password recovery functionality
- Multi-factor authentication

1.3 System Summary

Provide a concise overview of the system architecture and core modules.

Follows a client-side architecture with four main pages connected through a JavaScript-based navigation system. Data is stored in JS objects and arrays, this simulates a database. This project uses no frameworks for DOM manipulation, event handling, and data processing. CSS is what handles all the styling and operates entirely in the browser.

2. System Architecture

Detail the system's architecture, including diagrams if available. Explain major components, data flow, and integration points.

2.1 Components

Landing Page (landing.html):

- Marketing and informational homepage
- Features section showcasing banking benefits
- Operations tabs displaying service offerings
- Testimonial slider with customer reviews
- Modal-based login interface
- Smooth scroll navigation to page sections

Dashboard/Home Page (home.html):

- Protected main banking interface
- Real-time balance display
- Transaction history with scrollable list
- Money transfer functionality
- Loan request system
- Account closure capability
- Automatic logout timer for security
- Summary display (income, expenses, interest)

Sign Up Page (signup.html):

- Account registration form
- Password strength validation (7+ characters, uppercase, lowercase, number, special character)
- Username validation (letters only, 2+ characters)
- Success notification banner
- Automatic redirect to landing page after registration

Savings Calculator Page (savings.html):

- Compound interest calculator
- Input fields for initial deposit, monthly contributions, interest rate, and time period

- Real-time calculation display
- Year-by-year breakdown table
- Results showing final balance, total contributions, and interest earned

2.2 Data Flow

Describe how data moves through the system. Include diagrams where possible.

Data Persistence**:**

- All data exists in browser memory only
- Page refresh clears all data
- Returns to initial state with demo accounts (Gustavo, Jessica)

2.3 Technologies Used

Core Technologies**:**

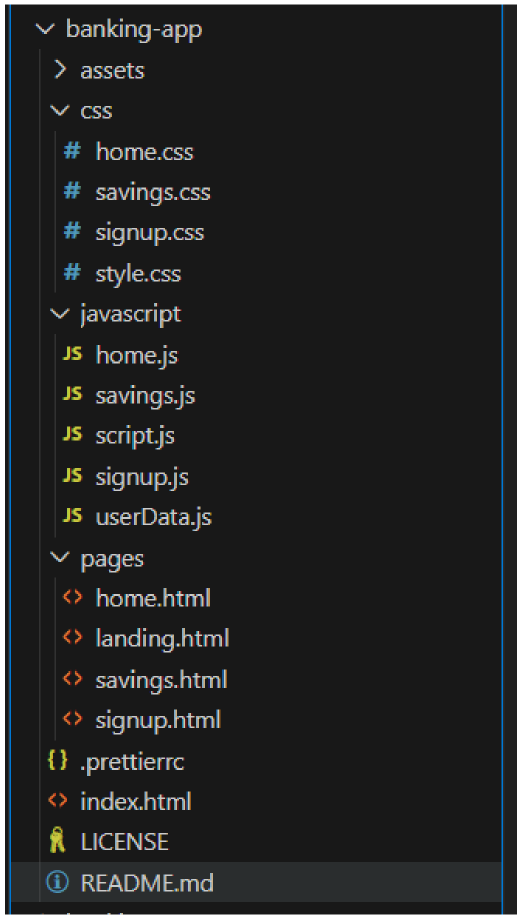
- HTML: Semantic markup, forms, modals, structure
- CSS: Flexbox, Grid, animations, transitions, media queries, custom properties (variables)
- JavaScript: Classes, arrow functions, array methods, DOM manipulation, event listeners, Intersection Observer API, template literals

3. Implementation Details

Explain how the software is implemented, including folder structure, key modules, and critical algorithms.

3.1 Folder Structure

- 3 folders hold the structures of this project. CSS folder for all the styling, HTML for all the website structure, and a JS folder to hold all the JS files.

● 

```
banking-app
  assets
  css
    home.css
    savings.css
    signup.css
    style.css
  javascript
    home.js
    savings.js
    script.js
    signup.js
    userData.js
  pages
    home.html
    landing.html
    savings.html
    signup.html
  .prettierrc
  index.html
  LICENSE
  README.md
```

3.2 Key Modules

Describe the purpose and function of major modules.

userData.js (Data Layer):

- Defines CreateAccount class for user objects
- Stores demo accounts (account1, account2)
- Maintains accounts array as in-memory database
- Properties include: owner, pin/password, movements, movementsDates, interestRate, currency, locale

script.js (Landing Page Controller):

- Modal open/close functionality
- Login authentication logic
- Smooth scroll navigation
- Section reveal animations (Intersection Observer)
- Testimonial slider with keyboard/mouse controls
- Operations tabbed interface
- Navigation fade effects on hover
- Sticky navigation on scroll

home.js (Dashboard Controller):

- Login form handler
- Display functions for movements, balance, summary
- Transfer money between accounts
- Loan request processing
- Account closure functionality
- Sort transactions by amount
- Logout timer with countdown
- Currency and date formatting
- Username generation from full names

signup.js (Registration Controller):

- Form validation (username and password)
- Password strength checking (5+ chars, upper/lower/number/special)
- Username validation (letters only, 2+ characters)
- Account creation using CreateAccount class
- Success banner animation
- Error message display
- Form clearing and redirect logic

savings.js (Calculator Controller - embedded in HTML):

- Compound interest calculation algorithm
- Monthly contribution processing
- Year-by-year breakdown generation
- Currency formatting for results
- Dynamic table population
- Form validation for numeric inputs
- Real-time result updates

4. Setup and Installation

Instructions for setting up the development environment and running the software locally or in production.

4.1 Prerequisites

List dependencies, such as software versions and required tools.

- Text editor or IDE (VS Code recommended)
- Understanding of HTML/CSS/JavaScript

4.2 Configuration

Explain configuration files, environment variables, and startup parameters.

- Application runs without configuration
- All settings hardcoded in source files

5. Maintenance and Future Work

Explain maintenance practices and list planned improvements or future features.

5.1 Known Issues

Document current bugs, limitations, or technical debt.

Data Persistence:

- All data lost on page refresh
- No localStorage or sessionStorage implementation
  - Accounts reset on refresh/reload.

Authentication:

- Passwords stored in JavaScript. (Could implement a real authenticator down the line).
- UserName is set to full names, not actually usernames.

5.2 Future Enhancements

List potential features or optimizations for later versions.

Data Persistence:

- LocalStorage for temporary data storage
- Backend database for permanent storage

Security Improvements:

- HTTPS enforcement
- Two-factor authentication (2FA)
- CAPTCHA to prevent Botting.

Design Improvements:

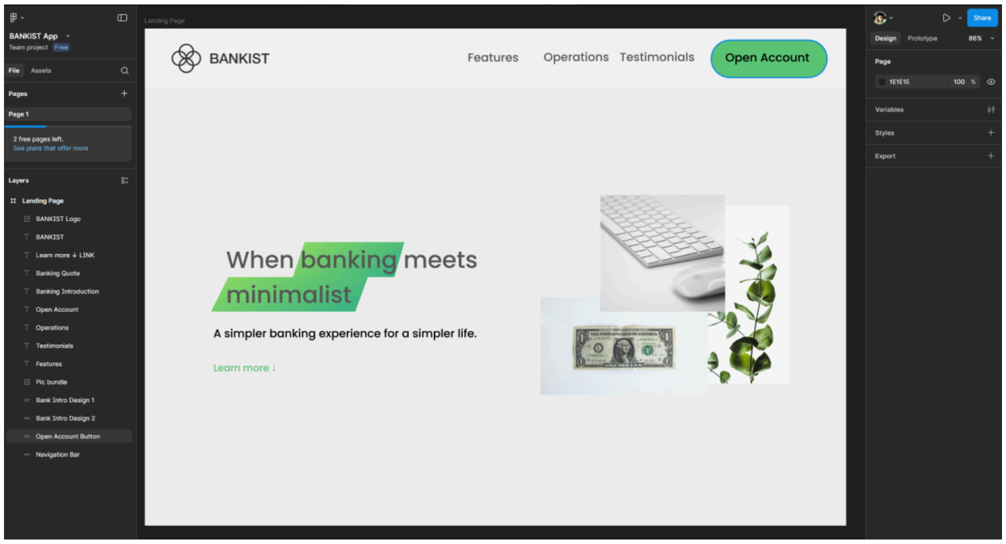
- Micro-interactions and animations

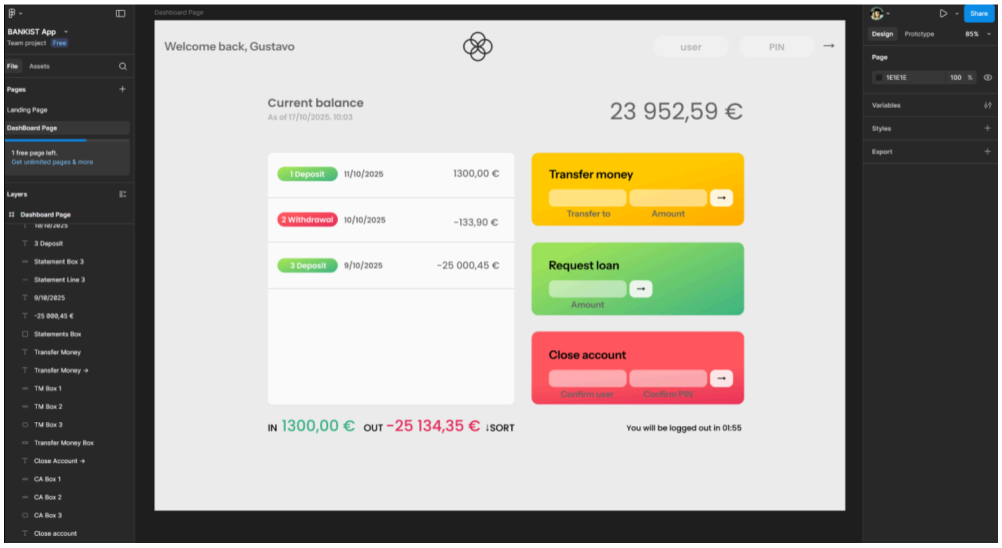
Code Enhancements:

- Refactor to use modern framework (React, Vue)
- Implement state management (Redux, Vuex)
- Add TypeScript for type safety

6. Appendices

Figma diagrams used for landing page and dashboard for testing purposes below:


BANKIST App
Team project Free
File
Assets
Pages
Page 1
2 free pages left.
See plans that offer more
Layers
Landing Page
BANKIST Logo
BANKIST
Learn more ↓ LINK
Banking Quote
Banking Introduction
Open Account
Operations
Testimonials
Features
Pic bundle
Bank Intro Design 1
Bank Intro Design 2
Open Account Button
Navigation Bar
Landing Page
BANKIST
Features
Operations
Testimonials
Open Account
When banking meets minimalist
A simpler banking experience for a simpler life.
Learn more ↓
Share
Design
Prototype
86%
Page
1E1E1E
100 %
Variables
Styles
Export


BANKIST App
Team project Free
File
Assets
Pages
Landing Page
DashBoard Page
1 free page left.
Get unlimited pages & more
Layers
Dashboard Page
10/10/2025
3 Deposit
Statement Box 3
Statement Line 3
9/10/2025
-25 000,45 €
Statements Box
Transfer Money
Transfer Money →
TM Box 1
TM Box 2
TM Box 3
Transfer Money Box
Close Account →
CA Box 1
CA Box 2
CA Box 3
Close account
Dashboard Page
Welcome back, Gustavo
user
PIN
→
Current balance
As of 17/10/2025, 10:03
23 952,59 €
1 Deposit
11/10/2025
1300,00 €
2 Withdrawal
10/10/2025
-133,90 €
3 Deposit
9/10/2025
-25 000,45 €
Transfer money
Transfer to
Amount
Request loan
Amount
Close account
Confirm user
Confirm PIN
IN 1300,00 €
OUT -25 134,35 €
↓SORT
You will be logged out in 01:55
Share
Design
Prototype
85%
Page
1E1E1E
100 %
Variables
Styles
Export