IBAN Validation System Design Document

Table of Contents

1. Introduction

Purpose of the Document

This document serves as a detailed guide to the design and architecture of the IBAN validation system. It outlines the system's components, functionality, and data flow.

System Overview

The IBAN validation system is designed to allow users to register, log in, and validate IBAN numbers. Admins have access to view a list of all entered IBAN numbers. The system follows a client-server architecture.

2. System Architecture

High-Level Architecture

The system follows a client-server architecture with a frontend built using React and a backend built using Laravel. The frontend communicates with the backend API to perform user actions and IBAN validation.

Components and Modules

The system comprises the following main components:
- Frontend (React)
- Backend (Laravel)
- Database (MySQL)

Data Flow

- User registration and login requests are handled by the backend API.
- IBAN validation requests are also processed by the backend API.
- Admins access the list of IBAN numbers through the backend.

3. Database Schema

User Table

The User table stores user registration information, including username, hashed password and role.

IBANs Table

The IBANs table stores validated IBAN numbers, associated user IDs, and timestamps.

Relationships

There is a one-to-many relationship between users and IBANs, where each user can have multiple IBANs associated with their account.

4. User Registration and Authentication

User Registration Flow

Users register by providing a username and password. User passwords are securely hashed before storage.

Authentication Flow

Authentication is done by comparing the hashed password during login with the stored hashed password.

5. IBAN Validation

Validation Logic

This custom validation rule for International Bank Account Numbers (IBANs) ensures that user provided IBANs meet the correct format and mathematical requirements through the following steps:

Step 1: IBAN Input Preparation

- The first step involves preparing the user-provided IBAN. This includes removing any spaces and converting it to uppercase, ensuring a standardized format for validation.

Step 2: Format Validation

- Following input preparation, the IBAN undergoes format validation. It is checked against a regular expression to verify if it matches the expected pattern. A valid IBAN should consist of:
- Two uppercase letters (country code).
- Two digits (check digits).
- Up to 30 alphanumeric characters (the account number).

Step 3: IBAN Rearrangement

- After format validation, the IBAN is rearranged. This involves moving the first 4 characters to the end of the IBAN, aligning it with the typical IBAN structure.

Step 4: Letter-to-Digit Conversion

- To facilitate mathematical calculations, letters within the IBAN are replaced with their corresponding numeric values. This conversion is essential for consistent mathematical operations.

Step 5: IBAN Modulo Operation

- The validation process includes a modulo operation (remainder) on the rearranged and converted IBAN. This operation calculates the remainder when dividing the IBAN by '97'. A valid IBAN should yield a remainder of '1'.

Step 6: Validation Result

- Based on the result of the modulo operation, the validation rule returns `true` if the remainder is '1', indicating that the IBAN is valid. If the remainder is not '1', it returns `false`, indicating an invalid IBAN.

Step 7: Error Message

- In the case of validation failure, an error message is generated: "The :attribute is not a valid IBAN." This message is displayed to inform users that the provided IBAN is invalid.

By following these steps, the validation rule ensures the accuracy and validity of IBANs entered by users in the application.

6. Admin Functionality

Admin Authentication

Admins are authenticated using separate admin credentials. Only admins have access to admin-specific features.

Admin Access to IBAN Data

Admins can view a paginated list of all entered IBAN numbers.

7. Error Handling

Detailed error handling is implemented to provide meaningful responses for various scenarios, including validation errors, authentication failures, and server errors.

8. API Endpoints

The system provides various API endpoints for user registration, login, IBAN validation, and admin functionality.

9. Security

User data is protected by securely hashing passwords and using HTTPS for data transmission. Authentication and authorization mechanisms are in place to ensure that only authorized users can access certain features.

10. Deployment

Deployment instructions, including environment setup and deployment steps, are provided.

11. Testing

Testing Strategy

Manual Testing Approach

1. Validation Logic Testing:
    - Scenario: Manually entered various IBANs, including valid and invalid ones.
    - Procedure: Entered IBANs into the application's IBAN validation input field.
    - Observations: Observed the application's response to each IBAN input, checking whether it correctly identified valid and invalid IBANs.
    - Results: Documented the results, noting any discrepancies or issues in the validation logic.

2. User Interface Testing:
    - Scenario: Tested the IBAN input form in the application's user interface.
    - Procedure: Interacted with the IBAN input form, entering IBANs and reviewing validation feedback.
    - Observations: Assessed the user interface's behavior, including the display of error messages for invalid IBANs.
    - Results: Documented any anomalies or user interface-related issues encountered during testing.

3. End-to-End Testing:
    - Scenario: Simulated end-to-end user workflows, including registration, login, IBAN entry, and validation.
    - Procedure: Manually executed the complete user journey, from registration to IBAN validation.
    - Observations: Verified that the IBAN validation feature seamlessly integrated into the user experience and correctly stored valid IBANs in the database.
    - Results: Documented any deviations or unexpected behavior observed during the end-to-end testing.

5. Exploratory Testing:
    - Scenario: Conducted exploratory testing to uncover any unanticipated issues.
    - Procedure: Explored the IBAN validation feature with a curious and open mindset, trying different inputs and interactions.
    - Observations: Noted any unexpected behavior or edge cases that surfaced during exploratory testing.
    - Results: Documented findings from exploratory testing, especially if they revealed critical issues.

6. Regression Testing:
    - Scenario: Performed regression testing after code changes or updates.
    - Procedure: Revisited previously tested scenarios to ensure that recent changes did not introduce new issues.

- Observations: Checked if the IBAN validation feature still functioned correctly alongside other application features.
- Results: Documented the outcomes of regression testing.

Manual testing was essential to validate the IBAN validation feature and ensure its correctness and usability.

12. Conclusion

This document summarizes the design and architecture of the IBAN validation system, providing a comprehensive overview of its functionality and technical aspects.