**Software Requirements Specification (SRS) for Modern Banking Application**

**1. Introduction**

**1.1 Purpose**

This document outlines the requirements for a modern web-based banking application that allows customers to manage their accounts and transactions, while providing managers with oversight capabilities.

**1.2 Scope**

The banking application will provide:

* Customer account management (registration, login, transactions)
* Transaction processing (deposits, withdrawals, transfers)
* Manager oversight of all customers and transactions
* Responsive web interface accessible on multiple devices
* Comprehensive data validation and error handling

**1.3 Definitions, Acronyms, and Abbreviations**

* **SRS**: Software Requirements Specification
* **UI**: User Interface
* **UX**: User Experience
* **localStorage**: HTML5 web storage for persisting data in the browser

**1.4 References**

* HTML5, CSS3, and JavaScript ES6+ standards
* Responsive web design principles

**2. Overall Description**

**2.1 Product Perspective**

The Modern Banking Application is a standalone web application that operates entirely within a web browser, using localStorage for data persistence.

**2.2 Product Functions**

* User registration and authentication with validation
* Customer banking operations (deposit, withdraw, transfer) with transaction validation
* Transaction history and statements
* Manager dashboard with system oversight
* Comprehensive input validation and error handling
* Responsive design for mobile and desktop devices

**2.3 User Characteristics**

* **Customers**: Bank account holders who need to perform transactions and view their account history
* **Managers**: Bank staff who need to monitor all customer activities and system transactions

**2.4 Constraints**

* Application data persists only in browser's localStorage
* No server-side component or database
* Limited to single-browser usage (no cross-device synchronization)

**2.5 Assumptions and Dependencies**

* Users have modern browsers with JavaScript enabled
* localStorage is available and not disabled

**3. System Features and Requirements**

**3.1 Authentication System**

**3.1.1 User Registration**

**Functional Requirements:**

* The system shall allow new users to register as either customers or managers
* The system shall validate all input fields during registration with specific rules:
  + Name: Required, minimum 2 characters, maximum 50 characters
  + Email: Required, valid email format, unique across system
  + Password: Required, minimum 6 characters
  + Initial Deposit: Required for customers, numeric, minimum $1, maximum $10,000
  + Manager Code: Required for managers, must equal "MANAGER2023"

**Validation Details:**

* Email format validation using regex pattern: /.+@.+\..+/
* Email uniqueness check against existing users
* Numeric validation for initial deposit with range checking
* Manager code exact match validation

**3.1.2 User Login**

**Functional Requirements:**

* The system shall allow registered users to log in
* The system shall validate credentials against stored user data
* The system shall validate:
  + Email format
  + Password presence
  + User type selection

**Validation Details:**

* All fields required validation
* Credential matching against stored users
* Error message for invalid credentials: "Invalid email or password"

**3.1.3 Session Management**

* The system shall maintain user session using localStorage
* The system shall provide a logout function
* The system shall update UI elements based on login status

**3.2 Customer Features**

**3.2.1 Account Dashboard**

* The system shall display:
  + Customer name
  + Current account balance
  + Account activity date
  + Transaction history with filtering capabilities

**3.2.2 Deposit Functionality**

**Functional Requirements:**

* The system shall allow customers to deposit funds into their account
* Each deposit shall require:
  + Amount (must be positive numeric value)
  + Optional description (max 100 characters)

**Validation Details:**

* Amount validation: Required, numeric, greater than 0, maximum $10,000 per transaction
* Balance update validation: Ensure arithmetic correctness
* Transaction recording: Date, type, amount, description, new balance

**3.2.3 Withdrawal Functionality**

**Functional Requirements:**

* The system shall allow customers to withdraw funds from their account
* Each withdrawal shall require:
  + Amount (must be positive numeric value and not exceed available balance)
  + Optional description (max 100 characters)

**Validation Details:**

* Amount validation: Required, numeric, greater than 0
* Sufficient funds validation: Amount must be ≤ current balance
* Error message for insufficient funds: "Insufficient funds for this withdrawal"
* Transaction recording: Date, type, amount, description, new balance

**3.2.4 Transfer Functionality**

**Functional Requirements:**

* The system shall allow customers to transfer funds to other customers
* Each transfer shall require:
  + Recipient's email address (must belong to existing customer)
  + Amount (must be positive numeric value and not exceed available balance)
  + Optional description (max 100 characters)

**Validation Details:**

* Recipient validation: Required, must be valid email, must belong to existing customer
* Self-transfer validation: Prevent transfers to own account
* Amount validation: Required, numeric, greater than 0, ≤ current balance
* Error messages:
  + "Recipient not found" for invalid email
  + "You cannot transfer money to yourself" for self-transfer
  + "Insufficient funds for this transfer" for amount exceeding balance

**3.2.5 Transaction History**

**Functional Requirements:**

* The system shall display all transactions for the customer
* The system shall provide filtering options by:
  + Transaction type (all, deposit, withdrawal, transfer)
  + Date
  + Search keywords
* Each transaction shall display:
  + Date and time (formatted consistently)
  + Description
  + Transaction type (with visual indicators)
  + Amount (with color coding for deposits/withdrawals)
  + Resulting balance

**Validation Details:**

* Date formatting validation: Consistent across all displays
* Transaction type categorization: Accurate classification of all transactions
* Balance calculation validation: Ensure arithmetic accuracy across all transactions

**3.3 Manager Features**

**3.3.1 Manager Dashboard**

* The system shall display:
  + Total number of customers
  + Total funds held in all accounts
  + Total number of transactions
  + Number of transactions today

**Validation Details:**

* Statistical calculations: Accurate counts and sums across all data
* Date filtering: Correct identification of "today's" transactions

**3.3.2 Customer Management**

* The system shall display a list of all customers
* The list shall include:
  + Customer ID
  + Customer name
  + Email address
  + Current balance
* The system shall provide search functionality for customers
* The system shall allow managers to view transaction history for any customer

**Validation Details:**

* Data integrity: Accurate customer information display
* Search functionality: Case-insensitive search by name or email

**3.3.3 Transaction Oversight**

* The system shall allow managers to view all system transactions
* The system shall display transactions with:
  + Date and time
  + Description
  + Transaction type
  + Amount
  + Associated customer

**Validation Details:**

* Data aggregation: Correct compilation of all transactions
* Customer association: Accurate linking of transactions to customers

**3.4 System-Wide Features**

**3.4.1 Notification System**

* The system shall display temporary notifications for:
  + Successful operations
  + Error conditions
  + Validation messages

**Validation Details:**

* Notification timing: 3-second display duration
* Message clarity: Clear, user-friendly messages
* Type indication: Visual differentiation between success and error messages

**3.4.2 Responsive Design**

* The system shall adapt to different screen sizes
* The system shall provide a mobile-friendly navigation menu

**3.4.3 Data Persistence**

* The system shall persist data using browser's localStorage
* The system shall initialize with default data structures if none exist

**Validation Details:**

* Data structure integrity: Proper JSON formatting
* Storage limits: Handling of localStorage size limitations

**4. External Interface Requirements**

**4.1 User Interfaces**

* **Home Page**: Marketing page with application information and call-to-action
* **Login Page**: Form for user authentication with validation
* **Registration Page**: Form for new user registration with validation
* **Customer Dashboard**: Overview of account and transaction history
* **Manager Dashboard**: System overview and customer management

**4.2 Hardware Interfaces**

* The application shall be accessible on devices with web browsers

**4.3 Software Interfaces**

* The application shall use modern web standards (HTML5, CSS3, JavaScript)
* The application shall use the browser's localStorage API for data persistence

**4.4 Communications Interfaces**

* The application does not require network communication beyond initial page load

**5. Non-Functional Requirements**

**5.1 Performance Requirements**

* The application shall load within 3 seconds on average broadband connection
* UI interactions shall respond within 100ms
* Data filtering operations shall complete within 500ms

**5.2 Security Requirements**

* Passwords shall be stored in plain text (note: this is a simplification for the demo)
* Users shall only access their own accounts
* Managers shall have view-only access to customer data
* Input sanitization to prevent script injection attacks

**5.3 Software Quality Attributes**

* **Usability**: Intuitive interface with clear navigation and feedback
* **Maintainability**: Well-structured code with clear separation of concerns
* **Reliability**: Proper error handling for all operations
* **Data Integrity**: Validation ensuring data consistency and accuracy

**6. Data Validation Specifications**

**6.1 Input Validation Rules**

**User Registration:**

| Field | Type | Required | Validation Rules | Error Message |
| --- | --- | --- | --- | --- |
| Name | Text | Yes | 2-50 characters | "Please enter a valid name (2-50 characters)" |
| Email | Email | Yes | Valid format, unique | "Email already registered" or "Invalid email format" |
| Password | Text | Yes | Minimum 6 characters | "Password must be at least 6 characters" |
| Initial Deposit | Number | For customers | 1-10000 | "Initial deposit must be between $1 and $10,000" |
| Manager Code | Text | For managers | Exact match: "MANAGER2023" | "Invalid manager authorization code" |

**Financial Transactions:**

| Field | Type | Required | Validation Rules | Error Message |
| --- | --- | --- | --- | --- |
| Amount | Number | Yes | > 0, ≤ current balance (for withdrawals/transfers) | "Please enter a valid amount" or "Insufficient funds" |
| Recipient Email | Email | For transfers | Valid format, exists in system, not self | "Recipient not found" or "Cannot transfer to yourself" |
| Description | Text | No | Max 100 characters | "Description too long (max 100 characters)" |

**6.2 Business Rule Validation**

**Transaction Validation:**

1. **Withdrawal Validation**:
   * Check available balance ≥ withdrawal amount
   * Prevent negative balance
   * Record transaction with timestamp
2. **Transfer Validation**:
   * Validate recipient exists and is a customer
   * Prevent self-transfer
   * Validate sufficient funds
   * Update both accounts atomically
3. **Balance Calculation**:
   * Ensure arithmetic accuracy across all transactions
   * Maintain transaction history integrity

**6.3 Error Handling Specifications**

**Validation Error Types:**

1. **Field Validation Errors**: Highlight specific fields with issues
2. **Business Rule Errors**: Display contextual error messages
3. **System Errors**: Generic error message for unexpected issues

**Error Message Guidelines:**

* Clear, user-friendly language
* Specific to the validation failure
* Guidance on how to correct the issue
* Consistent styling and placement

**7. Appendix**

**7.1 Use Case Scenarios**

**UC-1: Customer Registration with Validation**

1. User navigates to registration page
2. User selects "Customer" account type
3. User fills in:
   * Name: "John Doe" (valid)
   * Email: "[john@example.com](https://mailto:john@example.com/)" (valid, unique)
   * Password: "password123" (valid)
   * Initial Deposit: "1000" (valid)
4. System validates all inputs successfully
5. System creates new account with initial transaction
6. System logs user in and redirects to customer dashboard
7. System shows success notification

**UC-2: Failed Registration - Existing Email**

1. User navigates to registration page
2. User selects "Customer" account type
3. User fills in:
   * Name: "Jane Smith" (valid)
   * Email: "[john@example.com](https://mailto:john@example.com/)" (already exists)
   * Password: "pass123" (valid)
   * Initial Deposit: "500" (valid)
4. System validates inputs, finds email already registered
5. System displays error message: "Email already registered"
6. System highlights email field as invalid
7. User corrects email to "[jane@example.com](https://mailto:jane@example.com/)" (unique)
8. System completes registration successfully

**UC-3: Money Transfer with Validation**

1. Customer selects "Transfer" action
2. Customer enters:
   * Recipient: "[mary@example.com](https://mailto:mary@example.com/)" (valid, exists, not self)
   * Amount: "250" (valid, ≤ current balance)
   * Description: "Birthday gift" (valid)
3. System validates all inputs successfully
4. System updates both accounts' balances
5. System records transaction in both accounts' history
6. System shows success notification: "Transfer successful!"

**UC-4: Failed Transfer - Insufficient Funds**

1. Customer selects "Transfer" action
2. Customer enters:
   * Recipient: "[mary@example.com](https://mailto:mary@example.com/)" (valid)
   * Amount: "5000" (exceeds current balance of $1000)
   * Description: "Loan repayment"
3. System validates amount, finds insufficient funds
4. System displays error message: "Insufficient funds for this transfer"
5. System highlights amount field as invalid
6. User corrects amount to "500" (valid)
7. System completes transfer successfully

This comprehensive SRS document provides detailed requirements for the Modern Banking Application, with particular emphasis on functional details and data validation requirements based on the provided JavaScript implementation.