**Currency Converter**

Process, Design, and Data Flow Guide

Business Overview

The Currency Converter is a web-based application that enables users to convert between different currencies in real-time. It provides a user-friendly interface for currency conversion and maintains a history of previous conversions.

Why?

- Need for quick and accurate currency conversion

- Real-time exchange rate updates

- Historical tracking of conversions

- User-friendly interface for financial calculations

Who?

- Primary Users: Individuals needing currency conversion

- Secondary Users: Financial analysts, travelers, business professionals

- Stakeholders: Development team, product owners, end-users

Technology

- Frontend: HTML5, CSS3, JavaScript (ES6+)

- API: Real-time exchange rate API

- Storage: Local Storage for history

- Testing: Manual testing, cross-browser testing

Process Flow

Module 1: Currency Conversion Module

Brief:

Handles the core currency conversion functionality, including input validation, API integration, and result display.

Features

Feature 1: Currency Selection

Task 1: Dropdown Implementation

- Acceptance Criteria:

  - Dropdown should display all available currencies

  - Country flags should be visible

  - Currency codes should be accurate

  - Selection should update immediately

- Test Cases:

  1. Verify dropdown opens on click

  2. Check all currency options are present

  3. Verify flag images load correctly

  4. Test currency selection updates

- Technical Details:

  - User Story: As a user, I want to select currencies from dropdown menus

  - Implementation: Custom dropdown component with flag icons

  - Data Flow: Currency selection → Update conversion parameters

Feature 2: Amount Input

\*\*Task 1: Input Validation\*\*

- Acceptance Criteria:

  - Only numeric input allowed

  - Decimal point handling

  - Maximum value limits

  - Real-time validation

- Test Cases:

  1. Test numeric input

  2. Test decimal input

  3. Test large numbers

  4. Test invalid characters

- Technical Details:

  - User Story: As a user, I want to enter amounts safely

  - Implementation: Input validation with regex

  - Data Flow: Input → Validation → Update conversion

Module 2: History Module

Brief:

Manages the storage and display of conversion history.

Features

Feature 1: History Storage

\*\*Task 1: Local Storage Implementation\*\*

- Acceptance Criteria:

  - History should persist between sessions

  - Maximum history limit

  - Clear history functionality

  - Timestamp recording

- Technical Details:

  - User Story: As a user, I want to see my conversion history

  - Implementation: Local Storage API

  - Data Flow: Conversion → Store → Display

Third Party Integration

Exchange Rate API

- Purpose: Fetch real-time exchange rates

- Integration Points:

  - Currency conversion endpoint

  - Rate update mechanism

  - Error handling

- Data Flow:

  1. Request rates

  2. Process response

  3. Update conversion

Security Matrix

- Input sanitization

- API key protection

- XSS prevention

- Rate limiting

Notifications Matrix

- Error notifications

- Rate update notifications

- Input validation messages

- API status messages

Data Design

Database Structure

- Local Storage Schema:

  - Conversion History

  - User Preferences

  - API Cache

Table Descriptions

1. Conversion History

   - Timestamp

   - Source Currency

   - Target Currency

   - Amount

   - Result

2. User Preferences

   - Default Currencies

   - Theme Settings

   - History Limit

Data Dictionary

| Element Name | Description | Data Type | Format | Remarks |

|-------------|-------------|-----------|--------|---------|

| timestamp | Conversion time | String | ISO 8601 | UTC time |

| sourceCurrency | From currency | String | ISO 4217 | 3-letter code |

| targetCurrency | To currency | String | ISO 4217 | 3-letter code |

| amount | Input amount | Number | Decimal | Positive only |

| result | Conversion result | Number | Decimal | Rounded to 2 places |

Project Estimation

- Planning: 1 week

- Development: 2 weeks

- Testing: 1 week

- Deployment: 1 day

Milestones

1. UI Design Completion

2. Core Functionality

3. API Integration

4. Testing Phase

5. Deployment

Glossary

- ISO 4217: Standard for currency codes

- API: Application Programming Interface

- XSS: Cross-Site Scripting

- Local Storage: Web storage API

- Exchange Rate: Value of one currency in terms of another