Finance Flow - Requirements Specification

1. Introduction

1.1 Purpose

This document defines the functional and non-functional requirements for Finance Flow, a comprehensive personal finance management web application designed for local deployment. The application will migrate from Base44 cloud storage to PostgreSQL while maintaining all existing functionality.

1.2 Scope

Finance Flow provides centralized tracking and management of personal finances including income, expenses, bills, debts, assets, insurance policies, and financial accounts. The application is designed for individual users with local data storage and no cloud dependencies.

1.3 Definitions and Acronyms

- **UI**: User Interface
- **API**: Application Programming Interface
- **CRUD**: Create, Read, Update, Delete operations
- **JWT**: JSON Web Token
- **PostgreSQL**: Open-source relational database
- **Base44**: Current cloud-based backend platform (to be replaced)

1.4 Document Overview

This specification covers functional requirements, non-functional requirements, system constraints, and acceptance criteria for the Finance Flow application.

2. Overall Description

2.1 Product Perspective

Finance Flow is a standalone web application that operates entirely on local infrastructure. It consists of:

- **Frontend**: React-based single-page application
- **Backend**: Node.js/Express API server
- **Database**: PostgreSQL database for data persistence
- **Local Deployment**: Self-hosted with optional Docker containerization

2.2 Product Features Summary

Based on the comprehensive analysis of existing implementation and specification documents:

11 Main Application Pages:

1. Dashboard - Financial overview with AI insights
2. Income - Multi-category income tracking

3. Expenses - Comprehensive expense management
4. Bills - Payment tracking with due dates
5. Debts - Debt portfolio with reduction strategies
6. Assets - Asset tracking with appreciation analysis
7. Budget - Flexible budgeting system
8. Debt Reduction - AI-powered payoff strategies
9. Credit & Loans - Credit card and loan management
10. Insurance - Policy management system
11. Accounts - Secure credential storage

10 Core Data Entities: Income, Expense, Bill, Debt, Asset, CreditCard, InsurancePolicy, BudgetItem, Budget, Account

2.3 User Characteristics

- **Primary Users**: Individuals managing personal finances
- **Technical Skill Level**: Basic to intermediate computer users
- **Usage Pattern**: Daily to weekly data entry and review
- **Data Volume**: Hundreds to thousands of financial records per user

2.4 Operating Environment

- **Client Side**: Modern web browsers (Chrome, Firefox, Safari, Edge)
- **Server Side**: Linux/Windows/macOS with Node.js 18+
- **Database**: PostgreSQL 14+ with local installation
- **Network**: Local area network (no internet dependency required)

3. Functional Requirements

3.1 User Authentication & Management

3.1.1 User Registration

REQ-AUTH-001: Users shall be able to create accounts with email and password

- **Input**: Email address, password, name (optional)
- **Validation**: Valid email format, password complexity requirements
- **Output**: User account creation confirmation
- **Error Handling**: Duplicate email, weak password notifications

3.1.2 User Login

REQ-AUTH-002: Users shall be able to authenticate with email and password

- **Input**: Email address and password
- **Security**: JWT token generation for session management
- **Output**: Authentication token and redirect to dashboard
- **Error Handling**: Invalid credentials, account lockout after failed attempts

3.1.3 Password Management

REQ-AUTH-003: Users shall be able to change passwords

- **Input**: Current password, new password, confirmation
- **Validation**: Current password verification, new password complexity
- **Security**: Secure password hashing with bcrypt
- **Output**: Password change confirmation

3.2 Dashboard Functionality

3.2.1 Financial Overview Cards

REQ-DASH-001: Dashboard shall display key financial metrics

- **Metrics**: Monthly income, expenses, net worth, savings rate
- **Calculation**: Real-time computation from user data
- **Display**: Card-based layout with visual indicators
- **Update Frequency**: Real-time on data changes

3.2.2 Expense Breakdown Visualization

REQ-DASH-002: Dashboard shall show expense distribution

- **Chart Type**: Interactive pie chart
- **Data Source**: Current month expense categories
- **Interactivity**: Click to drill down into category details
- **Colors**: Consistent color scheme across categories

3.2.3 Upcoming Bills Widget

REQ-DASH-003: Dashboard shall display upcoming bill payments

- **Time Range**: Next 7 days
- **Information**: Bill name, amount, due date, status
- **Alerts**: Visual indicators for overdue bills
- **Actions**: Quick payment marking functionality

3.2.4 Financial Health Score

REQ-DASH-004: Dashboard shall provide AI-powered financial insights

- **Analysis**: Income vs expenses, debt-to-income ratio, savings rate
- **Recommendations**: Actionable financial advice
- **Scoring**: Numerical health score with explanation
- **Updates**: Monthly recalculation of metrics

3.3 Income Management

3.3.1 Income Entry

REQ-INC-001: Users shall be able to record income transactions

- **Fields**: Source, amount, category, date received, description
- **Categories**: Salary, freelance, investment, rental, business, other
- **Validation**: Positive amounts, valid dates, required fields
- **Output**: Confirmation of income record creation

3.3.2 Recurring Income Setup

REQ-INC-002: Users shall be able to set up recurring income

- **Frequencies**: Weekly, bi-weekly, monthly, quarterly, yearly
- **Automation**: Automatic generation of future income records
- **Management**: Ability to modify or stop recurring entries
- **Notifications**: Alerts for generated recurring entries

3.3.3 Income Analytics

REQ-INC-003: Users shall view income analysis and trends

- **Chart Types**: Category breakdown pie chart, monthly trend line chart
- **Filtering**: By category, date range, income source
- **Calculations**: Total income, average income, growth trends
- **Export**: Ability to export data for external analysis

3.3.4 Income History

REQ-INC-004: Users shall access complete income transaction history

- **Display**: Paginated table with sorting capabilities
- **Search**: Full-text search across descriptions and sources
- **Filtering**: By date range, category, amount range
- **Actions**: Edit, delete, duplicate existing entries

3.4 Expense Management

3.4.1 Expense Entry

REQ-EXP-001: Users shall be able to record expense transactions

- **Fields**: Description, amount, category, date, payment method
- **Categories**: Housing, transportation, food, utilities, healthcare, entertainment, shopping, insurance, debt payments, other
- **Payment Methods**: Cash, credit card, debit card, bank transfer, check
- **Validation**: Positive amounts, valid dates, required fields

3.4.2 Recurring Expense Management

REQ-EXP-002: Users shall be able to set up recurring expenses

- **Frequencies**: Weekly, bi-weekly, monthly, quarterly, yearly
- **Automation**: Automatic generation with user confirmation

- **Management**: Modify, pause, or cancel recurring expenses
- **Tracking**: History of all generated recurring transactions

3.4.3 Expense Analytics

REQ-EXP-003: Users shall view comprehensive expense analysis

- **Visualizations**: Pie chart by category, bar chart trends, spending heatmap
- **Comparisons**: Month-over-month, year-over-year analysis
- **Budgeting**: Actual vs budgeted expense comparison
- **Insights**: Spending pattern analysis and recommendations

3.4.4 Expense Filtering and Search

REQ-EXP-004: Users shall be able to filter and search expenses

- **Filters**: Category, date range, payment method, amount range
- **Search**: Description and merchant name search
- **Sorting**: By date, amount, category, payment method
- **Bulk Actions**: Select multiple expenses for batch operations

3.5 Bills Management

3.5.1 Bill Creation and Management

REQ-BILL-001: Users shall be able to manage recurring bills

- **Fields**: Name, amount, due date, category, notes
- **Categories**: Utilities, rent, mortgage, insurance, phone, internet, streaming services
- **Frequencies**: Weekly, bi-weekly, monthly, quarterly, yearly
- **Status**: Pending, paid, overdue tracking

3.5.2 Payment Status Tracking

REQ-BILL-002: Users shall be able to track bill payment status

- **Status Updates**: Mark bills as paid with payment date
- **History**: Complete payment history for each bill
- **Overdue Alerts**: Visual indicators for overdue payments
- **Auto-pay Indicators**: Track which bills have automatic payment

3.5.3 Bill Notifications

REQ-BILL-003: Users shall receive bill payment reminders

- **Upcoming Bills**: Dashboard widget showing next 7 days
- **Overdue Tracking**: Highlight and track overdue payments
- **Payment Confirmation**: Confirm payment marking actions
- **Bulk Actions**: Mark multiple bills as paid simultaneously

3.6 Debt Management

3.6.1 Debt Portfolio Tracking

REQ-DEBT-001: Users shall be able to track multiple debts

- **Fields**: Name, current balance, original amount, interest rate, minimum payment
- **Types**: Credit card, personal loan, student loan, mortgage, auto loan
- **Due Dates**: Track payment due dates and schedules
- **Priority Levels**: High, medium, low priority assignment

3.6.2 Debt Analytics

REQ-DEBT-002: Users shall view debt analysis and projections

- **Calculations**: Payoff time at current payment rate
- **Interest Tracking**: Total interest paid and remaining
- **Debt-to-Income Ratio**: Calculate and track ratio over time
- **Visualizations**: Charts showing debt by type and priority

3.6.3 Payment Tracking

REQ-DEBT-003: Users shall track debt payments and progress

- **Payment History**: Record of all payments made
- **Balance Updates**: Automatic balance calculation after payments
- **Progress Tracking**: Visual progress bars for each debt
- **Impact Analysis**: Show payment impact on total debt

3.7 Debt Reduction Strategies

3.7.1 AI Strategy Generation

REQ-DEBT-RED-001: System shall provide AI-powered debt reduction strategies

- **Methods**: Debt snowball vs avalanche comparison
- **Calculations**: Interest savings and payoff time comparisons
- **Recommendations**: Optimal payment allocation suggestions
- **Scenarios**: Multiple strategy scenario analysis

3.7.2 Payment Recommendations

REQ-DEBT-RED-002: System shall suggest optimal payment strategies

- **Analysis**: Current financial capacity assessment
- **Suggestions**: Monthly payment increase recommendations
- **Projections**: Time and interest savings calculations
- **Action Steps**: Specific steps to implement strategies

3.7.3 Progress Tracking

REQ-DEBT-RED-003: Users shall track debt reduction progress

- **Milestones**: Set and track debt reduction goals
- **Achievements**: Celebrate debt payoff milestones
- **Projections**: Updated timelines based on actual payments
- **Motivation**: Progress visualization and encouragement

3.8 Asset Management

3.8.1 Asset Portfolio Tracking

REQ-ASSET-001: Users shall be able to track asset portfolio

- **Fields**: Name, current value, purchase price, purchase date, category
- **Categories**: Real estate, vehicle, investment, savings, retirement
- **Locations**: Asset location or account tracking
- **Documentation**: Notes and description fields

3.8.2 Asset Valuation

REQ-ASSET-002: Users shall track asset value changes

- **Current Value**: Regular value updates and tracking
- **Appreciation**: Calculate appreciation rates and gains/losses
- **Performance**: Track asset performance over time
- **Comparisons**: Portfolio allocation and performance analysis

3.8.3 Asset Analytics

REQ-ASSET-003: Users shall view asset performance analytics

- **Visualizations**: Portfolio allocation pie chart, performance trends
- **Calculations**: Total portfolio value, appreciation rates
- **Comparisons**: Asset performance comparison
- **Projections**: Future value projections based on historical data

3.9 Budget Management

3.9.1 Budget Creation

REQ-BUDGET-001: Users shall be able to create flexible budgets

- **Budget Items**: Income and expense line items
- **Frequencies**: Weekly, bi-weekly, semi-monthly, monthly, yearly
- **Categories**: All income and expense categories
- **Time Periods**: Set budget start and end dates

3.9.2 Budget Tracking

REQ-BUDGET-002: Users shall track actual vs budgeted amounts

- **Comparisons**: Real-time budget vs actual comparison
- **Variance Analysis**: Calculate and display budget variances
- **Progress Indicators**: Visual progress bars and indicators
- **Alerts**: Notifications when approaching budget limits

3.9.3 Budget Analytics

REQ-BUDGET-003: Users shall view budget performance analytics

- **Visualizations**: Budget performance charts and graphs
- **Historical Analysis**: Budget performance over time
- **Category Performance**: Performance by budget category
- **Recommendations**: Suggestions for budget improvements

3.9.4 Calendar View

REQ-BUDGET-004: Users shall view budget items in calendar format

- **Monthly Calendar**: Display budget items by due date
- **Visual Indicators**: Color coding for different item types
- **Interactions**: Click calendar items for quick editing
- **Navigation**: Easy month-to-month navigation

3.10 Credit and Loans Management

3.10.1 Credit Card Management

REQ-CREDIT-001: Users shall manage credit card accounts

- **Fields**: Card name, credit limit, current balance, APR, annual fee
- **Utilization**: Calculate and track credit utilization rates
- **Payments**: Track payments and due dates
- **Rewards**: Track rewards programs and benefits

3.10.2 Credit Utilization Tracking

REQ-CREDIT-002: Users shall monitor credit utilization across all cards

- **Calculations**: Individual card and total utilization rates
- **Alerts**: Warnings when utilization exceeds recommended levels
- **Recommendations**: Suggestions for optimal utilization
- **History**: Track utilization changes over time

3.10.3 Loan Management

REQ-CREDIT-003: Users shall manage various loan types

- **Loan Types**: Personal, auto, student, mortgage loans
- **Tracking**: Principal, interest, payment schedules
- **Comparisons**: Loan comparison tools for refinancing

- **Analysis**: Loan performance and payment impact analysis

3.11 Insurance Management

3.11.1 Policy Management

REQ-INS-001: Users shall manage insurance policies

- **Fields**: Policy name, provider, policy number, type, premium
- **Types**: Health, auto, home, life, disability, travel, pet insurance
- **Coverage**: Coverage amounts and deductible tracking
- **Dates**: Policy start and end date management

3.11.2 Premium Tracking

REQ-INS-002: Users shall track insurance premiums and payments

- **Frequencies**: Monthly, quarterly, semi-annual, annual premiums
- **Payment Tracking**: Premium payment history and due dates
- **Cost Analysis**: Insurance cost analysis and comparisons
- **Renewal Alerts**: Notifications for policy renewals

3.11.3 Coverage Analysis

REQ-INS-003: Users shall analyze insurance coverage

- **Gap Analysis**: Identify potential coverage gaps
- **Recommendations**: Suggestions for coverage improvements
- **Cost Optimization**: Premium cost optimization recommendations
- **Policy Comparisons**: Compare different policy options

3.12 Account Credential Management

3.12.1 Secure Credential Storage

REQ-ACC-001: Users shall securely store financial account credentials

- **Fields**: Account name, type, website URL, username, password, account number
- **Encryption**: All sensitive data encrypted at rest
- **Categories**: Bank, credit card, investment, insurance, utility accounts
- **Security**: Password strength indicators and recommendations

3.12.2 Credential Access and Management

REQ-ACC-002: Users shall manage stored credentials securely

- **Visibility Controls**: Toggle password visibility with authentication
- **Copy Functionality**: Secure copy-to-clipboard for credentials
- **Access Logging**: Track credential access for security
- **Regular Updates**: Prompts for regular password updates

3.12.3 Account Integration

REQ-ACC-003: Users shall link accounts to other financial data

- **Website Links**: Direct links to account websites
- **Contact Information**: Store email addresses and phone numbers
- **Account Status**: Track active/inactive account status
- **Last Login**: Track last login dates for security monitoring

4. Data Management Requirements

4.1 Data Storage

4.1.1 PostgreSQL Database

REQ-DATA-001: System shall use PostgreSQL for data persistence

- **Version**: PostgreSQL 14 or higher
- **Schema**: Normalized database schema with proper relationships
- **Indexes**: Optimized indexes for query performance
- **Constraints**: Data integrity constraints and validations

4.1.2 Data Security

REQ-DATA-002: System shall implement comprehensive data security

- **Encryption**: Sensitive data encrypted at rest
- **Access Control**: User-based data isolation
- **Authentication**: Secure user authentication with JWT tokens
- **Audit Trail**: Logging of data access and modifications

4.1.3 Data Backup

REQ-DATA-003: System shall provide data backup capabilities

- **Automated Backups**: Daily automated database backups
- **Retention**: Configurable backup retention policies
- **Recovery**: Point-in-time recovery capabilities
- **Export**: Data export functionality for user portability

4.2 Data Migration

4.2.1 Base44 Migration

REQ-MIG-001: System shall migrate data from Base44 platform

- **Data Extraction**: Export all user data from Base44
- **Data Transformation**: Convert Base44 format to PostgreSQL schema
- **Data Validation**: Verify data integrity during migration
- **User Notification**: Inform users of migration progress and completion

4.2.2 Data Integrity

REQ-MIG-002: Migration shall maintain data integrity

- **Validation**: Verify all data relationships and constraints
- **Error Handling**: Handle migration errors gracefully
- **Rollback**: Ability to rollback failed migrations
- **Testing**: Comprehensive testing of migrated data

4.3 Data Privacy

4.3.1 Local Storage

REQ-PRIV-001: All user data shall be stored locally

- **No Cloud Dependencies**: Complete local data storage
- **User Control**: Users have full control over their data
- **Data Portability**: Ability to export and import data
- **Compliance**: Adherence to data privacy best practices

4.3.2 Data Sharing

REQ-PRIV-002: System shall not share user data externally

- **No Analytics**: No external analytics or tracking
- **No Cloud Sync**: No automatic cloud synchronization
- **User Consent**: Any data sharing requires explicit user consent
- **Transparency**: Clear privacy policy and data handling practices

5. User Interface Requirements

5.1 Design System

5.1.1 Visual Design

REQ-UI-001: Interface shall follow consistent design system

- **Color Palette**: Professional financial application color scheme
- **Typography**: Clear, readable font hierarchy
- **Icons**: Consistent icon library (Lucide React)
- **Spacing**: Consistent spacing and layout grid

5.1.2 Component Library

REQ-UI-002: Interface shall use standardized component library

- **Shadcn/ui**: Consistent UI components based on Radix UI
- **Form Controls**: Standardized form inputs and validation
- **Data Display**: Consistent tables, cards, and list components
- **Navigation**: Standardized navigation and menu components

5.1.3 Accessibility

REQ-UI-003: Interface shall meet accessibility standards

- **WCAG 2.1**: Compliance with Web Content Accessibility Guidelines
- **Keyboard Navigation**: Full keyboard navigation support
- **Screen Readers**: Screen reader compatibility
- **Color Contrast**: Adequate color contrast ratios

5.2 Responsive Design

5.2.1 Multi-Device Support

REQ-RESP-001: Interface shall work across different screen sizes

- **Desktop**: Optimized for desktop screens (1024px+)
- **Tablet**: Responsive design for tablet devices (768px-1024px)
- **Mobile**: Mobile-friendly design (320px-768px)
- **Navigation**: Adaptive navigation for different screen sizes

5.2.2 Touch Interface

REQ-RESP-002: Interface shall support touch interactions

- **Touch Targets**: Appropriately sized touch targets (44px minimum)
- **Gestures**: Support for common touch gestures
- **Feedback**: Visual feedback for touch interactions
- **Performance**: Smooth scrolling and animations

5.3 User Experience

5.3.1 Navigation

REQ-UX-001: Application shall provide intuitive navigation

- **Sidebar Navigation**: Fixed sidebar with main application sections
- **Breadcrumbs**: Clear indication of current location
- **Search**: Global search functionality across all data
- **Quick Actions**: Frequently used actions easily accessible

5.3.2 Data Entry

REQ-UX-002: Data entry shall be efficient and user-friendly

- **Form Validation**: Real-time form validation with clear error messages
- **Auto-complete**: Smart auto-completion for frequently used values
- **Bulk Operations**: Ability to perform bulk operations on data
- **Keyboard Shortcuts**: Keyboard shortcuts for power users

5.3.3 Data Visualization

REQ-UX-003: Financial data shall be presented visually

- **Charts**: Interactive charts and graphs using Recharts
- **Tables**: Sortable and filterable data tables
- **Dashboards**: Comprehensive dashboard with key metrics
- **Trends**: Visual trend analysis and comparisons

6. Performance Requirements

6.1 Response Time

6.1.1 Page Load Performance

REQ-PERF-001: Application pages shall load within acceptable time limits

- **Initial Load**: First page load within 3 seconds
- **Subsequent Navigation**: Page transitions within 1 second
- **Data Fetching**: API responses within 2 seconds
- **Search Results**: Search results within 1 second

6.1.2 Database Performance

REQ-PERF-002: Database operations shall perform efficiently

- **Query Response**: Database queries complete within 500ms
- **Complex Reports**: Financial calculations complete within 2 seconds
- **Concurrent Users**: Support for multiple concurrent local users
- **Data Volume**: Efficient handling of large datasets (10,000+ records)

6.2 Resource Usage

6.2.1 Memory Usage

REQ-PERF-003: Application shall use system resources efficiently

- **Client Memory**: Browser memory usage optimized
- **Server Memory**: Backend memory usage within reasonable limits
- **Database Memory**: PostgreSQL memory configuration optimized
- **Caching**: Appropriate caching strategies implemented

6.2.2 Storage Requirements

REQ-PERF-004: Application shall manage storage efficiently

- **Database Size**: Efficient database schema and indexing
- **File Storage**: Minimal file storage requirements
- **Growth Management**: Efficient handling of data growth over time
- **Cleanup**: Automated cleanup of unnecessary data

7. Security Requirements

7.1 Authentication and Authorization

7.1.1 User Authentication

REQ-SEC-001: System shall implement secure user authentication

- **Password Security**: Strong password requirements and hashing
- **Session Management**: Secure session handling with JWT tokens
- **Account Lockout**: Protection against brute force attacks
- **Password Recovery**: Secure password reset mechanism

7.1.2 Authorization

REQ-SEC-002: System shall enforce proper authorization

- **User Isolation**: Users can only access their own data
- **Role-Based Access**: Appropriate access controls implemented
- **API Security**: All API endpoints properly secured
- **Input Validation**: Comprehensive input validation and sanitization

7.2 Data Protection

7.2.1 Encryption

REQ-SEC-003: Sensitive data shall be properly encrypted

- **Data at Rest**: Database encryption for sensitive fields
- **Data in Transit**: HTTPS/TLS for all communications
- **Key Management**: Secure encryption key management
- **Algorithm Standards**: Use of industry-standard encryption algorithms

7.2.2 Privacy Protection

REQ-SEC-004: User privacy shall be protected

- **Data Minimization**: Only collect necessary data
- **Local Processing**: All data processing performed locally
- **No Tracking**: No external tracking or analytics
- **User Control**: Users control their data completely

7.3 System Security

7.3.1 Application Security

REQ-SEC-005: Application shall implement security best practices

- **OWASP Compliance**: Follow OWASP security guidelines
- **SQL Injection Prevention**: Parameterized queries and input validation
- **XSS Protection**: Cross-site scripting prevention measures
- **CSRF Protection**: Cross-site request forgery protection

7.3.2 Infrastructure Security

REQ-SEC-006: Infrastructure shall be properly secured

- **Network Security**: Appropriate firewall and network configuration
- **Server Hardening**: Operating system and service hardening
- **Update Management**: Regular security updates and patches
- **Monitoring**: Security monitoring and logging

8. Reliability and Availability

8.1 System Reliability

8.1.1 Error Handling

REQ-REL-001: System shall handle errors gracefully

- **User-Friendly Errors**: Clear, actionable error messages
- **Error Recovery**: Automatic recovery from transient errors
- **Data Integrity**: Maintain data integrity during error conditions
- **Logging**: Comprehensive error logging for troubleshooting

8.1.2 Data Backup and Recovery

REQ-REL-002: System shall provide reliable data backup and recovery

- **Automated Backups**: Regular automated backups
- **Backup Verification**: Verify backup integrity regularly
- **Recovery Testing**: Regular testing of recovery procedures
- **User-Initiated Backups**: Allow users to create manual backups

8.2 System Availability

8.2.1 Uptime Requirements

REQ-AVAIL-001: System shall maintain high availability

- **Local Availability**: 99.9% availability for local deployment
- **Maintenance Windows**: Planned maintenance with minimal downtime
- **Graceful Degradation**: System continues to function during partial failures
- **Health Monitoring**: System health monitoring and alerting

8.2.2 Disaster Recovery

REQ-AVAIL-002: System shall provide disaster recovery capabilities

- **Backup Strategy**: Comprehensive backup and recovery plan
- **Recovery Time**: Minimal recovery time objectives
- **Data Recovery**: Complete data recovery capabilities
- **Documentation**: Clear disaster recovery procedures

9. Compatibility Requirements

9.1 Browser Compatibility

9.1.1 Web Browser Support

REQ-COMPAT-001: Application shall support modern web browsers

- **Chrome**: Latest version and one version back
- **Firefox**: Latest version and one version back
- **Safari**: Latest version and one version back
- **Edge**: Latest version and one version back

9.1.2 JavaScript Requirements

REQ-COMPAT-002: Application shall require modern JavaScript support

- **ES2020**: Support for modern JavaScript features
- **Module Support**: ES6 module support required
- **LocalStorage**: Browser local storage support
- **WebSQL**: IndexedDB support for client-side storage

9.2 Operating System Compatibility

9.2.1 Server Operating Systems

REQ-COMPAT-003: Backend shall support multiple operating systems

- **Linux**: Ubuntu 20.04+, CentOS 8+, RHEL 8+
- **Windows**: Windows Server 2019+, Windows 10+
- **macOS**: macOS 11+ (Big Sur and later)
- **Docker**: Docker container support for all platforms

9.2.2 Database Compatibility

REQ-COMPAT-004: System shall support PostgreSQL versions

- **PostgreSQL**: Version 14+ with backwards compatibility to 12
- **Extensions**: Support for common PostgreSQL extensions
- **Configuration**: Flexible database configuration options
- **Migration**: Database migration tools and scripts

10. Maintenance and Support

10.1 System Maintenance

10.1.1 Update Management

REQ-MAINT-001: System shall support easy updates and maintenance

- **Version Control**: Clear versioning and release management

- **Update Process**: Simple update process for users
- **Backward Compatibility**: Maintain compatibility across versions
- **Migration Scripts**: Automated migration for database schema changes

10.1.2 Monitoring and Logging

REQ-MAINT-002: System shall provide comprehensive monitoring

- **Application Monitoring**: Monitor application performance and health
- **Database Monitoring**: Track database performance and usage
- **Error Logging**: Comprehensive error logging and reporting
- **Performance Metrics**: Track key performance indicators

10.2 Documentation and Support

10.2.1 User Documentation

REQ-DOC-001: System shall provide comprehensive user documentation

- **User Guide**: Complete user manual with screenshots
- **Quick Start**: Quick start guide for new users
- **FAQ**: Frequently asked questions and troubleshooting
- **Video Tutorials**: Video guides for complex features

10.2.2 Technical Documentation

REQ-DOC-002: System shall provide technical documentation

- **Installation Guide**: Detailed installation and setup instructions
- **Configuration**: Configuration options and best practices
- **API Documentation**: Complete API documentation
- **Developer Guide**: Documentation for customization and development

11. Testing Requirements

11.1 Testing Strategy

11.1.1 Unit Testing

REQ-TEST-001: System shall have comprehensive unit test coverage

- **Coverage**: Minimum 80% code coverage for critical components
- **Automated Testing**: Automated test execution in CI/CD pipeline
- **Test Framework**: Jest and React Testing Library for frontend testing
- **Backend Testing**: Node.js testing framework for API testing

11.1.2 Integration Testing

REQ-TEST-002: System shall include integration testing

- **API Testing**: Complete API endpoint testing
- **Database Testing**: Database integration and data integrity testing
- **Frontend Integration**: Component integration testing
- **End-to-End Testing**: Full user workflow testing

11.2 Quality Assurance

11.2.1 Code Quality

REQ-QA-001: System shall maintain high code quality standards

- **Code Review**: Peer code review process
- **Linting**: ESLint and Prettier for code consistency
- **Static Analysis**: Static code analysis tools
- **Security Scanning**: Automated security vulnerability scanning

11.2.2 User Acceptance Testing

REQ-QA-002: System shall undergo user acceptance testing

- **Test Scenarios**: Complete test scenarios for all features
- **User Feedback**: Incorporate user feedback into testing
- **Accessibility Testing**: Verify accessibility compliance
- **Performance Testing**: Load and performance testing

12. Acceptance Criteria

12.1 Functional Acceptance

12.1.1 Feature Completeness

REQ-ACCEPT-001: All specified features shall be fully implemented and functional

- **Dashboard**: Complete dashboard with all widgets and analytics
- **Financial Tracking**: All income, expense, bill, and debt tracking features
- **Asset Management**: Complete asset portfolio tracking and analytics
- **Budget System**: Flexible budgeting with actual vs budget comparison
- **Security**: Secure credential storage with encryption

12.1.2 Data Migration

REQ-ACCEPT-002: Data migration from Base44 shall be complete and accurate

- **Data Integrity**: All data migrated without loss or corruption
- **Relationship Preservation**: All data relationships maintained
- **User Verification**: Users can verify their migrated data
- **Migration Tools**: Provide tools for users to verify migration

12.2 Performance Acceptance

12.2.1 Performance Benchmarks

REQ-ACCEPT-003: System shall meet all performance requirements

- **Response Time**: All response time requirements met
- **Concurrent Users**: Support specified number of concurrent users
- **Data Volume**: Handle specified data volumes efficiently
- **Resource Usage**: Stay within specified resource limits

12.2.2 Reliability Testing

REQ-ACCEPT-004: System shall demonstrate required reliability

- **Error Handling**: Graceful handling of all error conditions
- **Data Recovery**: Successful data backup and recovery testing
- **Stress Testing**: System stability under stress conditions
- **Security Testing**: Pass all security vulnerability tests

12.3 User Acceptance

12.3.1 Usability Testing

REQ-ACCEPT-005: System shall pass usability testing with target users

- **Task Completion**: Users can complete all major tasks
- **Learning Curve**: New users can learn the system quickly
- **Efficiency**: Experienced users can work efficiently
- **Satisfaction**: Users express satisfaction with the system

12.3.2 Documentation Acceptance

REQ-ACCEPT-006: All documentation shall be complete and accurate

- **Completeness**: All features documented thoroughly
- **Accuracy**: Documentation matches actual system behavior
- **Clarity**: Documentation is clear and easy to understand
- **Updates**: Documentation updated with system changes

This requirements specification provides a comprehensive foundation for implementing the Finance Flow application with PostgreSQL migration while maintaining all existing functionality and ensuring a robust, secure, and user-friendly personal finance management system.