# **Test Plan**

Personal Finance Management App

Group Name: B7

Chu Chun Yin 1155194319 [1155194319@link.cuhk.edu.hk](mailto:1155194319@link.cuhk.edu.hk)

Ko Kam San 1155194979 [1155194979@link.cuhk.edu.hk](mailto:1155194979@link.cuhk.edu.hk)

Lai Tsz Hin 1155194965 [1155194965@link.cuhk.edu.hk](mailto:1155194965@link.cuhk.edu.hk)

Chan Yiu Cheung 1155193060 [1155193060@link.cuhk.edu.hk](mailto:1155193060@link.cuhk.edu.hk)

Instructor: Dr. Lam Tak Kei

Course: CSCI3100

Department: Computer Science and Engineering

Printing Date: 9/5/2025

## **1. Introduction**

This is the test plan for our project: Personal Finance Management App. This test plan ensures the app, which is developed with React (frontend), Go with GORM (backend), and SQL (database), meet functional and non-functional requirements for an secure, efficient, and user-friendly experience. The app enable users to manage finances through authentication, transaction tracking, budgeting, and reporting.

**2. Scope and Objectives**

### **2.1 Scope**

The test plan encompasses:

* **Functional Testing**: User authentication, transaction management, budget setting, reporting, and potential new features (e.g., profile management).
* **Non-Functional Testing**: Performance, security, usability, compatibility.
* **Testing Environments**: Development, testing, and staging with SQL databases.

**In Scope**:

* User authentication via server/handlers/auth.go and web/src/pages/Login.js, web/src/pages/Register.js.
* Transaction CRUD in server/handlers/transactions.go and web/src/pages/Transactions.js.
* Budget management in server/handlers/budgets.go and web/src/pages/Budgets.js
* Reporting in web/src/pages/Reports.js.
* Dashboard summaries in web/src/pages/Dashboard.js.
* New components (e.g., profile page)
* Compatibility with Chrome, Firefox, Safari, Edge on desktop
* Performance under 1000 transactions.
* Security (JWT, SQL injection prevention).
* Usability per WCAG 2.1.

**Out of Scope**:

* Third-party integrations
* Unsupported browsers (e.g., Internet Explorer).
* Extensive penetration testing beyond OWASP Top 10.
* Multi-language support unless in web/src/locales/

### **2.2 Objectives**

* Verify functionality aligns with docs/requirements.md
* Ensure performance (<5s load time) and security.
* Achieve 100% critical bug resolution.
* Confirm WCAG 2.1 accessibility.
* Prepare for course submission.

## **3. Test Cases and Scenarios**

### **3.1 Functional Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Description** | **Steps** | **Expected Result** | **Pass/Fail** |
| TC001 | Successful login | 1. Open web/src/pages/Login.js. 2. Enter valid email ([test@example.com](mailto:test@example.com)) and password. 3. Click "Login". | Redirect to web/src/pages/Dash-board.js, JWT stored. | Pass if dashboard loads. |
| TC002 | Failed login | 1. Open login page. 2. Enter invalid credentials. 3. Click "Login". | Error: "Invalid credentials". | Pass if error displayed. |
| TC003 | Register new user | 1. Open web/src/pages/Register.js. 2. Enter unique email, password. 3. Click "Register". | User added to SQL database, redirect to login. | Pass if registration succeeds. |
| TC004 | Add transaction | 1. Log in. 2. Navigate to web/src/pages/Transactions.js. 3. Enter amount (50), category (Food), date. 4. Click "Add". | Transaction saved via POST /transactions, visible in table. | Pass if transaction appears. |
| TC005 | Budget exceeded alert | 1. Set budget (Food: $100) via web/src/pages/Budgets.js. 2. Add transactions totaling $120. 3. View budget. | Alert: "Budget exceeded by $20". | Pass if alert shown. |
| TC006 | Generate report | 1. Log in. 2. Navigate to web/src/pages/Reports.js. 3. Select monthly report. 4. Click "Generate". | Chart rendered with spending data. | Pass if chart displays correctly. |
| TC007 | Update transaction | 1. Log in. 2. Navigate to web/src/pages/Transactions.js. 3. Click "Edit" on a transaction, update amount to 60. 4. Click "Save". | Transaction updated via PUT /transactions/:id. | Pass if transaction updated. |
| TC008 | Delete transaction | 1. Log in. 2. Navigate to web/src/pages/Transactions.js. 3. Click "Delete" on a transaction. | Transactions removed via DELETE /transactions/:id. | Pass if transaction deleted. |
| TC009 | View dashboard | 1. Log in. 2. Navigate to web/src/pages/Dashboard.js. | Displays balance, transactions, budgets, charts. | Pass if data loads correctly. |
| TC010 | Set budget | 1. Log in. 2. Navigate to web/src/pages/Budgets.js. 3. Enter category (Entertainment), amount (200). 4. Click "Set Budget". | Budget saved via POST /budgets. | Pass if budget set. |
| TC011 | View profile (new) | 1. Log in. 2. Navigate to web/src/pages/Profile.js (if exists). 3. View user details (email, name). | Displays user email, name. | Pass if profile loads (pending web.zip check). |

### **3.2 Non-Functional Test Cases**

#### **3.2.1 Performance**

* **TC012**: Page load time <5s with 1000 transactions (tool: JMeter, test web/build/index.html).
* **TC013**: API response time <2s for GET /transactions (tool: Postman).
* **TC014**: Server handles 100 concurrent users (tool: JMeter).

#### **3.2.2 Security**

* **TC015**: Prevent SQL injection in POST /login (tool: OWASP ZAP).
* **TC016**: Verify JWT token expiration in server/handlers/auth.go.
* **TC017**: Ensure HTTPS for API calls in server/main.go.

#### **3.2.3 Usability**

* **TC018**: New user adds transaction in <2 min (manual, test web/src/pages/Transactions.js).
* **TC019**: Screen reader compatibility for dashboard (tool: NVDA, test web/src/pages/Dashboard.js).
* **TC020**: WCAG 2.1 compliance (contrast, navigation) across pages (tool: WAVE).

#### **3.2.4 Compatibility**

* **TC021**: Login works on Chrome, Firefox, Safari, Edge (manual, test web/build/).
* **TC022**: Responsive design on macOS, Windows, iOS, Android (tool: BrowserStack).
* **TC023**: Functionality on screen sizes (1920x1080, 1366x768, 375x667).

## **4. Resource Allocation**

### **4.1 Team Roles and Responsibilities**

|  |  |  |
| --- | --- | --- |
| **Role** | **Responsibilities** | **Time Commitment** |
| Test Lead | Oversee testing, verify web/src/ components, manage docs/test\_plan.md. | 22 hrs/week |
| QA Tester | Execute test cases, log bugs in GitHub Issues, test UI in web/build/. | 18 hrs/week |
| Backend Developer | Write tests for server/handlers/, fix bugs, support API testing. | 10 hrs/week |
| Frontend Developer | Write tests for web/src/components/, ensure WCAG 2.1 compliance. | 12 hrs/week |
| UI/UX Designer | Validate usability, verify styling in web/src/styles/. | 8 hrs/week |
| Product Owner | Clarify requirements, approve UAT for UI components. | 5 hrs/week |

### **4.2 Tools and Software**

* **Test Management**: GitHub Wiki, Testiny.
* **Bug Tracking**: GitHub Issues (<https://github.com/Ivan103099/CSCI3100-Project/issues>).
* **Automation**: Go testing, Jest for web/src/\_\_tests\_\_/, Selenium for UI.
* **Performance**: JMeter, Postman.
* **Compatibility**: BrowserStack.
* **Accessibility**: NVDA, WAVE.

### **4.3 Testing Environments**

|  |  |  |
| --- | --- | --- |
| **Environment** | **Purpose** | **Owner** |
| Development | Unit testing (server/main.go, web/src/). | Our team member |
| Testing | System, integration tests with SQL data. | Out team member |
| Staging | UAT, mirrors production. | Our team member |

### **4.4 Time Allocation**

|  |  |  |
| --- | --- | --- |
| **Activity** | **Effort (%)** | **Responsible** |
| Test Planning | 10% | Our team member |
| Test Case Creation | 15% | Our team member |
| Test Execution | 40% | Our team member |
| Bug Reporting/Retesting | 20% | Our team member |
| Regression Testing | 15% | Our team member |

## **5. Testing Approach**

### **5.1 Testing Types**

* **Unit Testing**: Test server/handlers/ and web/src/components/ using Go testing and Jest.
* **Integration Testing**: Validate APIs with SQL database.
* **System Testing**: End-to-end workflows using Selenium.
* **Regression Testing**: Automated tests in GitHub Actions.
* **UAT**: Feedback from 5-10 users on UI components.

### **5.2 Methodologies**

* **Manual Testing**: UI validation in web/src/pages/.
* **Automated Testing**: Tests in web/src/\_\_tests\_\_/.
* **Exploratory Testing**: Edge cases in transactions and budgets.

## **6. Timeline and Schedule**

**Agile Model (5 Weeks, 3 Sprints)**

### **6.1 Sprint 1: Setup and Unit Testing (Apr 18 - Apr 25)**

* Update test cases in docs/test\_cases.md.
* Unit tests for web/src/pages/Login.js, server/handlers/auth.go.
* Integration tests for POST /transactions.
* Bug fixes, initial regression tests.

### **6.2 Sprint 2: System and Non-Functional Testing (Apr 25 - Apr 30)**

* System tests for UI workflows (e.g., login to budgeting).
* Performance (JMeter) and security (OWASP ZAP) tests.
* Usability (WAVE) and compatibility (BrowserStack) tests.
* Bug fixes, regression tests.

### **6.3 Sprint 3: UAT and Finalization (May 1 - May 9)**

* UAT with 5-10 users, focus on web/src/pages/.
* Address feedback, final regression tests.
* Prepare Group B7 Test\_plan.docx.

## **7. Risk Assessment and Mitigation**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Description** | **Mitigation** |
| UI Bugs | Issues in web/src/pages/ (e.g., chart rendering). | Early UI testing with Selenium. |
| API Delays | Slow development of server/handlers/. | Prioritize endpoints, daily standups. |
| SQL Injection | Vulnerabilities in server/handlers/. | Use GORM prepared statements, OWASP ZAP. |
| Compatibility | UI issues on mobile in web/build/. | BrowserStack testing in Sprint 2. |
| Scope Creep | New features from inaccessible link. | Freeze requirements post-Sprint 1. |

## **8. Success Criteria**

* 100% critical bug resolution.
* 90% unit test coverage (web/src/\_\_tests\_\_/, server/test/).
* Performance: Load time <5s, API response <2s.
* Security: No OWASP Top 10 vulnerabilities.
* Usability: 80%+ positive UAT feedback, WCAG 2.1 compliance.
* Compatibility: Functional on Chrome, Firefox, Safari, Edge.
* Approval by product owner and instructor.

## **9. Reporting Requirements**

* **Test Cases**: Stored in docs/test\_cases.md.
* **Bugs**: Tracked in GitHub Issues.
* **Progress**: Weekly meetings emailed summaries.
* **Final Report**: Group B7 Test\_plan.docx (due May 9, 2025).