Integration Phase

Date	15 April 2025
Team ID	SWTID1744094910
Project Title	Personal Expense Tracker App
Maximum Marks	2 Marks

End-to-End Testing

Objective: Validate the entire application workflow from user registration to expense tracking, budget management, and reporting.
Scope:

- Authentication (registration/login)
- Expense CRUD operations
- Budget setup and alerts
- Analytics and report generation
- Multi-user collaboration

Test Environment Setup

- Frontend: React/React Native app (web/mobile).
- Backend: Node.js/Express.js API.
- Database: PostgreSQL (test database with seeded data).
- **Tools:**
    - Cypress/Playwright: For web E2E testing.
    - Detox: For React Native mobile testing.
    - Postman/Newman: API testing and monitoring.
    - BrowserStack: Cross-browser/device testing.

Critical Test Scenarios

1. Authentication & User Registration

Test Case 1.1: Successful User Registration

- Steps:
    - Navigate to the registration page.
    - Enter valid email, password, and name.

- Submit the form.
- Expected Result:
  - Confirmation message: "Registration successful!"
  - JWT token stored in cookies/local storage.
  - User added to the database.

Test Case 1.2: Registration with Existing Email

- Steps:
  - Use an email already registered in the system.
  - Submit the form.
- Expected Result:
  - Error message: "Email already exists."

2. Expense Management

Test Case 2.1: Add a New Expense

- Prerequisite: Logged-in user.
- Steps:
  - Navigate to "Add Expense."
  - Enter amount ($50), category (Food), date (today), and notes.
  - Submit.
- Expected Result:
  - Expense appears in the dashboard.
  - Database record created in `expenses` table.

Test Case 2.2: Edit an Existing Expense

- Steps:
  - Open an existing expense.
  - Update the amount to $60.
  - Save changes.
- Expected Result:
  - Dashboard reflects the updated amount.
  - Database `expenses.amount` updated to 60.

3. Budget Management

Test Case 3.1: Set a Monthly Budget

- Steps:
  - Navigate to "Budgets."
  - Set a $500 monthly budget for "Food."
  - Add $450 in Food expenses.
- Expected Result:
  - Dashboard shows "You've spent $450 of $500."
  - Alert notification: "Approaching budget limit!"

Test Case 3.2: Exceed Budget Limit

- Steps:
  - Add another $100 Food expense.
- Expected Result:
  - Alert notification: "Budget exceeded for Food!"
  - Email/SMS sent (if configured).

4. Analytics & Reports

Test Case 4.1: Generate Monthly Report

- Steps:
  - Navigate to "Reports."
  - Select "April 2024" and export as PDF.
- Expected Result:
  - PDF report downloaded with correct totals.
  - Data matches database aggregates.

5. Multi-User Collaboration

Test Case 5.1: Share Expense in a Group

- Prerequisite: Two registered users (User A and User B).
- Steps:
  - User A creates a group "Family Budget" and invites User B.
  - User A adds a $200 expense and assigns 50% to User B.
- Expected Result:
  - User B's dashboard shows "You owe $100."

- `expense_shares` table updated with User B's share.

Security & Error Handling Tests

Test Case 6.1: Invalid Token Access

- Steps:
  - Use an invalid JWT token to access `/api/expenses`.
- Expected Result:
  - `401 Unauthorized` error.

Test Case 6.2: SQL Injection Attempt

- Steps:
  - Enter `' OR 1=1--` in the login email field.
- Expected Result:
  - Validation error: "Invalid email format."

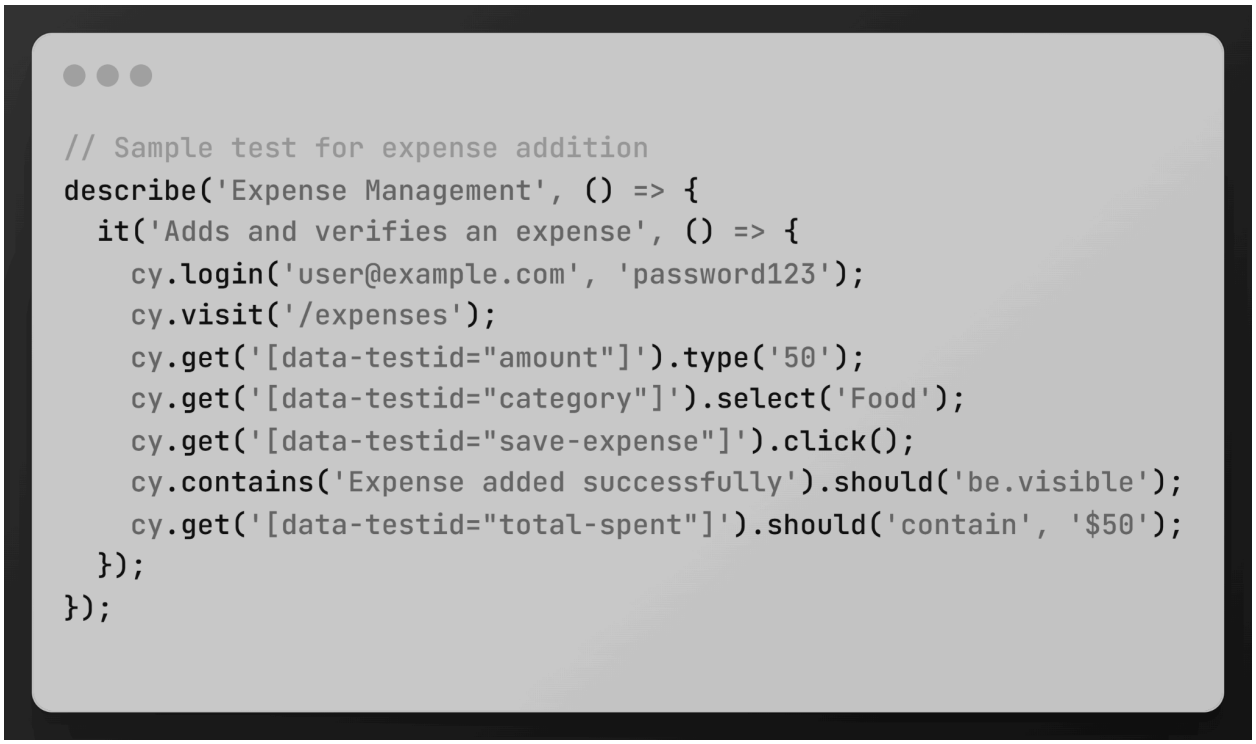
Performance & Load Testing

- Tool: K6/Locust.
- Scenario: Simulate 100 users adding expenses concurrently.
- Success Criteria:
  - API response time < 500ms.
  - 0% error rate.

Accessibility Testing

- Tool: Axe DevTools.
- Checks:
  - Keyboard navigation.
  - Screen reader compatibility.
  - Color contrast ratios.

Test Automation Script Example (Cypress)



```
// Sample test for expense addition
describe('Expense Management', () => {
  it('Adds and verifies an expense', () => {
    cy.login('user@example.com', 'password123');
    cy.visit('/expenses');
    cy.get('[data-testid="amount"]').type('50');
    cy.get('[data-testid="category"]').select('Food');
    cy.get('[data-testid="save-expense"]').click();
    cy.contains('Expense added successfully').should('be.visible');
    cy.get('[data-testid="total-spent"]').should('contain', '$50');
  });
});
```

Best Practices

- Test Data Management: Use factories/fixtures to seed test data.
- Parallel Execution: Run tests in parallel to reduce feedback time.
- CI/CD Integration: Trigger E2E tests on every PR using GitHub Actions.
- Visual Testing: Use Percy/Applitools to detect UI regressions.

Conclusion

End-to-end testing ensures all components of your Expense Tracker App work seamlessly together, catching integration issues early and validating real-world user workflows.