Personal Expense Tracker Project Timeline

This timeline outlines the tasks, learning phases, and project development steps for building a web-based personal expense tracker.

Week 1: December 13–19 (Planning, Learning Basics, and Setup)

Day 1 (Dec 13): Setup and Learning PHP Basics

Tasks:

- Set up UniServer, PHP, and MySQL environment.
- Familiarize with PHP syntax: variables, arrays, functions, and basic CRUD.
- Learn about `$_POST`, `$_GET`, and form processing.

Files to Create:

- `database/schema/init.sql`: Initial database schema.
- `backend/config/database.php`: Database connection script.

Learning Resources:

- PHP.net documentation
- W3Schools PHP tutorials

Day 2 (Dec 14): Database Design and PHP Integration

Tasks:

- Detailed MySQL database planning.
- Create tables for expenses and categories.
- Learn MySQL and PHP database integration.

Files to Create:

- `database/schema/expenses_table.sql`
- `backend/models/Expense.php`
- `backend/controllers/ExpenseController.php`

Learning Focus:

- MySQL table design
- PHP database connection methods

Day 3 (Dec 15): CSS Flexbox and Grid Fundamentals

Tasks:

- Learn CSS Flexbox layout techniques.
- Practice creating responsive layouts without frameworks.
- Understand CSS Grid for complex layouts.

Files to Create:

- `frontend/assets/css/styles.css`

Learning Resources:

- MDN Web Docs: CSS Flexbox
- CSS-Tricks Flexbox Guide

Day 4 (Dec 16): PHP Form Handling and Frontend Design

Tasks:

- Implement expense input form using custom CSS.
- Create form validation in PHP.
- Design a responsive layout with Flexbox.

Files to Create:

- `frontend/public/index.php`: Main expense entry page.
- `backend/services/ValidationService.php`

Day 5 (Dec 17): AJAX and JavaScript Basics

Tasks:

- Learn AJAX for dynamic form submissions.
- Implement client-side form validation.
- Create smooth user interactions without page reloads.

Files to Create:

- `frontend/assets/js/ajax.js`
- `frontend/assets/js/validation.js`

Day 6 (Dec 18): Implementing CRUD Operations

Tasks:

- Develop view expenses functionality.
- Create methods for listing, adding, and editing expenses.
- Enhance frontend display with Flexbox.

Files to Create:

- `backend/controllers/ExpenseController.php`
- `frontend/templates/expense-list.php`

Day 7 (Dec 19): Refining CRUD and Error Handling

Tasks:

- Implement edit and delete expense operations.
- Add error handling and user feedback.
- Improve overall application robustness.

Files to Create:

- `backend/controllers/EditExpenseController.php`
- `backend/controllers/DeleteExpenseController.php`

Week 2: December 20–26 (Core Functionality Development)

Day 8 (Dec 20): Chart.js Data Visualization

Tasks:

- Learn Chart.js for expense visualization.
- Create initial charts showing spending patterns.

Files to Create:

- `frontend/assets/js/chart-config.js`
- Update `backend/controllers/ExpenseController.php`.

Day 9 (Dec 21): Dynamic Chart Integration

Tasks:

- Fetch real expense data for charts.
- Implement dynamic category and time-based visualizations.

Day 10 (Dec 22): Data Export Functionality

Tasks:

- Implement CSV and JSON export features.
- Create backend export methods.

Day 11 (Dec 23): Advanced Form Validation

Tasks:

- Implement comprehensive form validation.

Day 12 (Dec 24): Comprehensive Testing

Tasks:

- Thoroughly test all features.

Week 3: December 27–31 (Enhancements, Final Features, and Debugging)

Day 15 (Dec 27): Advanced Chart.js Integration

Tasks:

- Add advanced spending visualizations (e.g., monthly trends, pie charts for categories).
- Fetch dynamic data from the database for visualizations.

Files to Create/Update:

- `frontend/assets/js/chart-config.js`
- Update `backend/controllers/ExpenseController.php`.

Day 16 (Dec 28): Profile Management (Optional)

Tasks:

- Add a user profile page for updating email and password.
- Validate profile updates both on the client and server sides.

Files to Create/Update:

- `frontend/public/profile.php`: Profile management form.
- `backend/controllers/UserController.php`: Add methods for updating user data.
- `backend/models/User.php`: Add methods for database updates.

Day 17 (Dec 29): Finalizing Export Features

Tasks:

- Complete CSV and JSON export functionality.
- Ensure the exported files contain relevant filters (e.g., date range, categories).

Files to Create/Update:

- `backend/services/ExportService.php`
- `backend/controllers/ExportController.php`
- Update `frontend/public/index.php` for export buttons.

Day 18 (Dec 30): Comprehensive Feature Testing

Tasks:

- Test all implemented features, including CRUD, charts, authentication, and export.
- Debug any issues and fix edge cases.

Day 19 (Dec 31): Performance Optimization

Tasks:

- Optimize database queries for faster performance.
- Minify CSS and JavaScript files for quicker load times.
- Optimize Chart.js rendering for large datasets.

Week 4: January 1–5 (Final Testing, Deployment, and Submission)

Day 20 (Jan 1): Deployment Setup

Tasks:

- Prepare the application for deployment.
- Set up the project on a local server (UniServer) or a web hosting platform.
- Test database connectivity on the deployment environment.

Files to Update:

- `backend/config/database.php`: Update with deployment database credentials.
- `.htaccess`: Add for URL rewriting (if using Apache).

Day 21 (Jan 2): Cross-Browser and Device Testing

Tasks:

- Test the application on different browsers (Chrome, Firefox, Edge, Safari).
- Test the responsiveness of the application on various devices.

Day 22 (Jan 3): Final Code Review and Cleanup

Tasks:

- Review all code for readability and maintainability.
- Add comments and remove unused files or code.

Day 23 (Jan 4): Documentation

Tasks:

- Write comprehensive documentation for the project.
- Include setup instructions, features, and usage examples.

Files to Create/Update:

- `README.md`: Add installation steps, project overview, and features.

Day 24 (Jan 5): Final Submission or Presentation

Tasks:

- Submit the project or prepare for presentation.
- If presenting, prepare a live demo of the application.