Expense Tracker NAKISA

ISHAAN BAJAJ ishaanbajaj12@gmail.com

Contents

Implementation Overview

✅ **Advanced Filtering and Pagination**

- Backend: Used JpaSpecificationExecutor to support filtering by category, date range, and amount. Implemented pagination using Pageable.
- Frontend: UI controls for filter form (dropdowns, range slider, date pickers). Integrated pagination with page indicators.

- **Decision**: JpaSpecification provides flexible query building for future enhancements like sorting.

✅ **Validation and Error Handling**

- Backend: Used JSR-303 annotations (@NotBlank, @DecimalMin, etc.) with global exception handling via @ControllerAdvice.
- Frontend: Dynamic display of field-specific validation errors under input fields.

- **Decision**: Clean separation of concerns and DRY error management.

Soft Delete & Archiving

✅ **Soft Delete Implementation**

- DELETE endpoint marks `deleted=true` instead of hard deleting.
- Added `/api/expenses/archived` to return soft-deleted expenses.
- Scheduled auto-archiving using `@**Scheduled**` for expenses older than 30 days.

✅ **Frontend Changes**

- Replaced "Delete" with "Archive" button.
- Created tab layout: "Current Expenses" vs "Archived Expenses".

Decision: Soft deletes enable reversibility and traceability; scheduled jobs ensure aging data is cleaned.

Testing Strategy

✅ **Backend Testing**

- Unit Tests (ExpenseServiceTest): Covered both valid and invalid inputs (nulls, negative amounts, blank fields).
- Integration Tests (ExpenseControllerTest):Validated full request lifecycle with TestRestTemplate.

✅ **Frontend Testing**

- Used Vue Test Utils to simulate form submission in `ExpenseForm.vue`.
- Confirmed that `expense-added` event is emitted correctly on success.
- Axios was mocked for isolated test.

Decision: Mixed testing approach ensures both logic correctness and API contract integrity.

Challenges & Solutions

🚧 Challenges Faced

- Frontend testing with Vue 2 + Jest + Babel was fragile and required overrides.
- Ensuring global error messages map to the right frontend fields.

✅ Solutions

- Used scoped `babel-jest` config and Vue aliasing to resolve test errors.
- Validated server-side errors using JSON keys and mapped them accordingly in the form.

Improvements & Recommendations

🔧 **Suggested Improvement**

- Introduce caching (e.g., Spring Cache): to reduce DB hits for repeated filters or archived data.
  - Benefit: Reduces server load & improves frontend speed.

💡 **Additional Feature**

- Add Export to CSV or PDF on frontend for expense records.
- Useful for tax filing or sharing with accountants.

For details about the suggestions, contact: ishaanbajaj12@gmail.com

Summary

✅ All four exercises completed: filtering, validation, soft delete, and testing.

👨‍💻 Code follows clean architecture, DRY principles, and is modular for extensibility.

🚀 System is now production-ready with scalable backend and responsive frontend.

Name: Ishaan Bajaj
Email: ishaanbajaj12@gmail.com
Phone: +1(438)725-2685
Website: ishaanbajaj.com