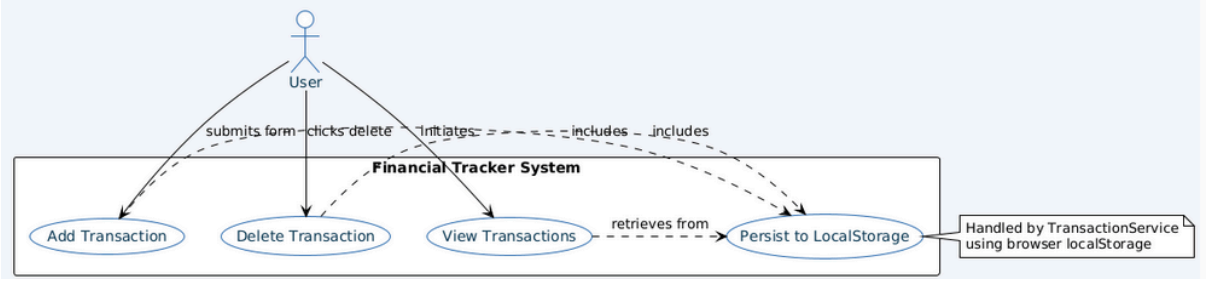
NAMA: Erick Marchian
210111401035
TEKNIK INFORMATIKA



Use Case Name	View Transactions
Goal	Display all stored transactions in a sortable, filterable list with details (date, description, amount, type)
Actors	User (primary)
Preconditions	• User is logged into the application • User has navigated to the transactions page or app has loaded • At least zero transactions exist in localStorage
Main Flow	1. User navigates to /transactions route 2. TransactionList component initializes (ngOnInit) 3. Component calls TransactionService.getAll() 4. Service retrieves transactions from localStorage (key: ft_transactions_v1) 5. Service sorts transactions by date (newest first) 6. Service returns sorted array to component 7. Component updates transactions[] property 8. View renders Material table with all transactions 9. User sees transaction details: Date, Description, Amount (in IDR), Type (income/expense)
Postconditions	• Transaction list is displayed on screen • User can see all stored transactions • User can interact with delete buttons for each transaction • User can navigate to "Add Transaction" form
Alternative Flows	**A1: No transactions exist** - Service returns empty array - View displays empty table message or placeholder **A2: LocalStorage is corrupted or empty** - Service initializes empty transactions array - View displays empty table

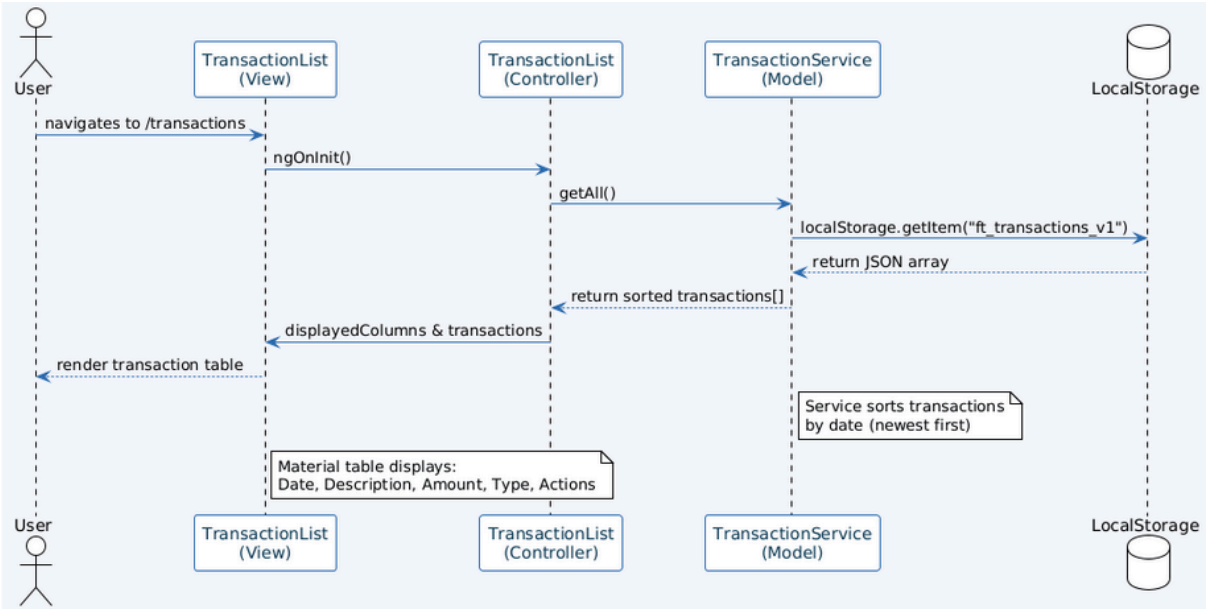
Exceptions

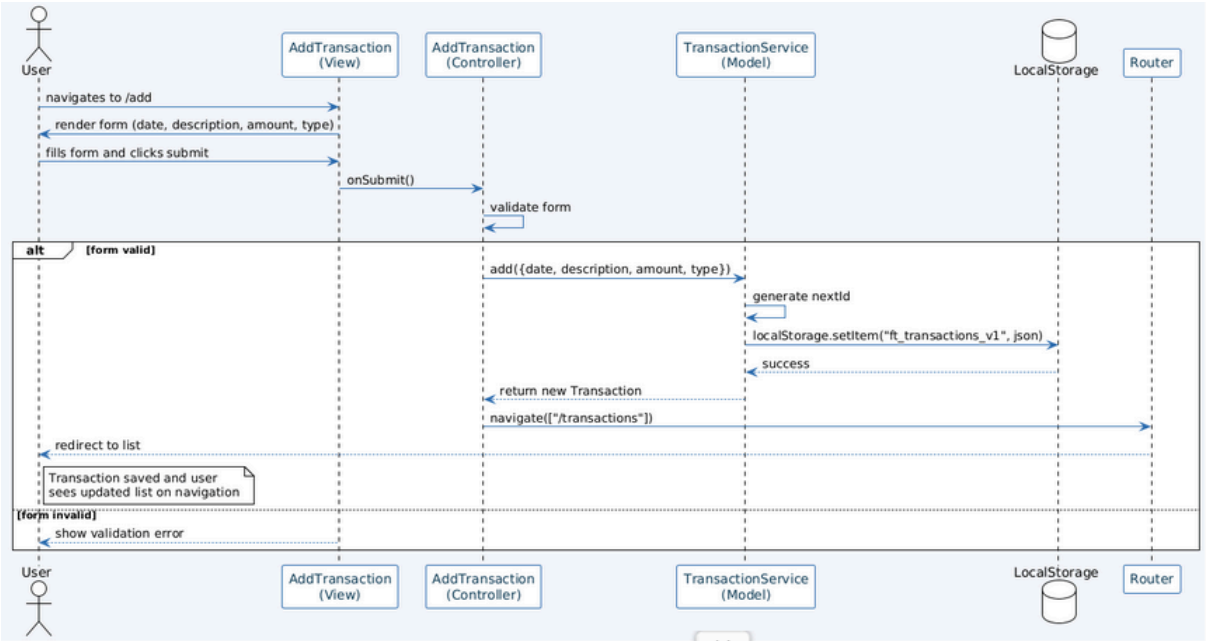
E1: Browser localStorage is disabled

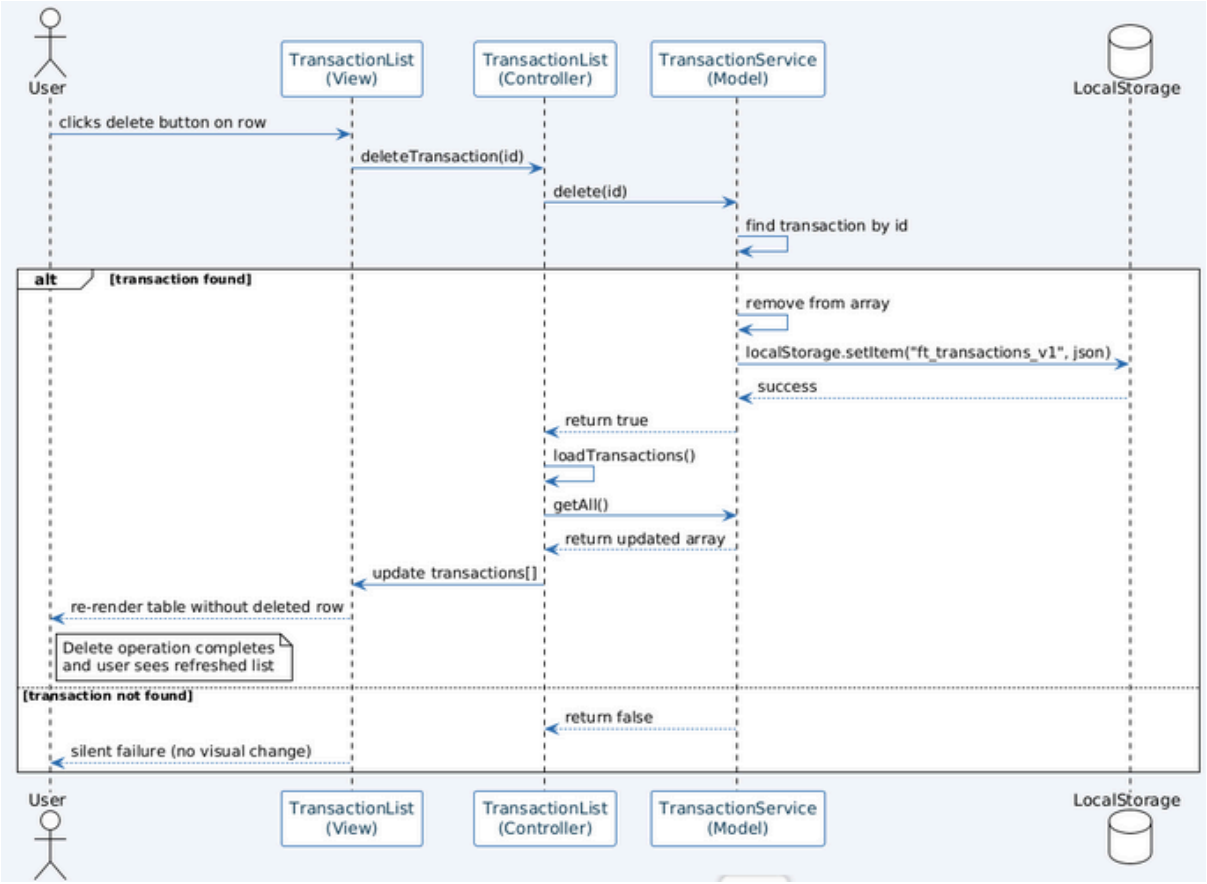
- Service logs error
- User sees empty list (graceful degradation)

E2: Invalid JSON in localStorage

- Service catches parsing error, reinitializes as empty
- User sees empty list







Sequence Diagram → Angular Mapping

This table maps the participants/objects shown in the sequence diagrams to the concrete Angular artifacts in this project: components, templates, services, methods, and routes.

Sequence Object	Angular Element	File / Location	Key Methods / Bindings	Route / Template
User (actor)	External actor (browser user)	n/a	Interacts via clicks, form submit, navigation	n/a
AddTransaction (View)	Component (Controller for add form)	src/app/add-transaction/add-transaction.ts	onSubmit() — validates form, calls TransactionService.add(...)	Template: src/app/add-transaction/add-transaction.html — form inputs (date, description,

				amount, type); Route: /add
AddTransaction (UI)	Reactive Form	within AddTransaction component	transactionForm: FormGroup — validators: required fields, amount >= 0	Form controls bound in template via formControlName
Transaction List (View / Controller)	Component (Controller for list)	src/app/transaction-list/transaction-list.ts	ngOnInit() → loadTransactions() → calls TransactionService.getAll(); deleteTransaction(id) → calls TransactionService.delete(id)	Template: src/app/transaction-list/transaction-list.html; Route: /transactions
Transaction Service (Model)	Injectable Service (single source of truth)	src/app/services/transaction.service.ts	getAll(): Transaction[] — returns sorted array; add(tx: Omit<Transaction,'id'>): Transaction — generates id, pushes and saves; delete(id: number): boolean — removes and saves	Persists to localStorage in save() / load() methods
LocalStorage (persistence)	Browser localStorage API	not in repo (browser API)	localStorage.getItem('ft_transactions_v1') and localStorage.setItem('ft_transactions_v1', json) used by TransactionService	Storage key: ft_transactions_v1
Router	Angular Router	src/app/app.routes.ts	Routes wired to components: { path: 'transactions', component: TransactionList }, { path: 'add', component: AddTransaction }	Used by components to navigate(['/transactions']) after add

Message → Implementation Mapping (from sequence diagrams)

- submit(form) (User → AddTransaction UI)
    - Implemented as user clicking the form submit button bound to onSubmit() in AddTransaction component.
    - Template: submit button calls (ngSubmit)="onSubmit()".
- onSubmit() → TransactionService.add(tx)
    - AddTransaction.onSubmit() validates and packages form values and calls TransactionService.add(...).
    - Code: this.txService.add({ date, description, amount, type });
- getAll() (TransactionList → TransactionService)
    - TransactionList.loadTransactions() calls this.txService.getAll() to fetch current transactions.
- delete(id) (TransactionList → TransactionService)
    - TransactionList.deleteTransaction(id) calls this.txService.delete(id) and reloads list on success.
- localStorage.setItem(...) / getItem(...)
    - Implemented inside TransactionService.save() and TransactionService.load().

Files of interest

- Components
    - src/app/transaction-list/transaction-list.ts (+ template and css)
    - src/app/add-transaction/add-transaction.ts (+ template and css)
- Service
    - src/app/services/transaction.service.ts
- Model
    - src/app/models/transaction.ts (interface definition)
- Routing
    - src/app/app.routes.ts

1. System Description

Financial Tracker is an Angular-based web application designed to help users manage personal financial transactions. The system provides a clean, intuitive interface for recording income and expense transactions with automatic persistence to browser storage.

Key Features

- Add transactions with date, description, amount, and type (income/expense)
- View all transactions in a sortable table format
- Delete transactions with confirmation
- Persistent storage (transactions saved automatically to localStorage)
- Currency display in Indonesian Rupiah (IDR)
- Responsive Material Design interface

Technology Stack

- **Framework**: Angular (v20.3.8) with standalone components
- **UI Library**: Angular Material
- **Language**: TypeScript
- **Storage**: Browser localStorage
- **Localization**: Indonesian locale

---

2. Component Structure

Application Architecture

Financial Tracker Application

├── AppComponent (src/app/app.ts)

│ ├── Toolbar (Navigation & Branding)

│ └── Router Outlet

│ ├── TransactionList Component

│ └── AddTransaction Component

├── TransactionService (src/app/services/transaction.service.ts)

│ └── localStorage Management (Key: ft_transactions_v1)

└── Transaction Model (src/app/models/transaction.ts)

Core Components

TransactionList Component (src/app/transaction-list/)

- **Purpose**: Display all transactions in a Material table
- **Features**:
  - Shows transaction date, description, amount, and type
  - Currency formatted as IDR
  - Delete button for each transaction
  - Sorted by date (newest first)
- **Methods**:
  - ngOnInit(): Loads transactions on component initialization
  - loadTransactions(): Fetches all transactions from service
  - deleteTransaction(id): Removes a transaction

AddTransaction Component (src/app/add-transaction/)

- **Purpose**: Form for creating new transactions
- **Features**:
  - Reactive form with validation
  - Date picker input
  - Description text field
  - Amount input with currency
  - Type dropdown (Income/Expense)
  - Submit button
- **Methods**:
  - onSubmit(): Adds transaction and navigates back to list

TransactionService (src/app/services/transaction.service.ts)

- **Purpose**: Handle CRUD operations and persistence
- **Methods**:
  - load(): Fetch transactions from localStorage
  - save(): Persist transactions to localStorage
  - getAll(): Return all transactions
  - add(transaction): Create new transaction
  - delete(id): Remove transaction
- **Storage**: localStorage key ft_transactions_v1

AppComponent (src/app/app.ts)

- **Purpose**: Main application container
- **Features**:
  - Material toolbar with navigation buttons

- Router outlet for component switching
- Links to Add Transaction and View Transactions

---

3. How to Run the Application

Prerequisites

Node.js 18.x or higher

npm 9.x or higher

Angular CLI 20.3.8

Installation Steps

Install dependencies

npm install

Start the development server

npm start

Or alternatively:

ng serve

1. **Open in browser**
    - Navigate to http://localhost:4200/
    - The app loads automatically and hot-reloads on file changes

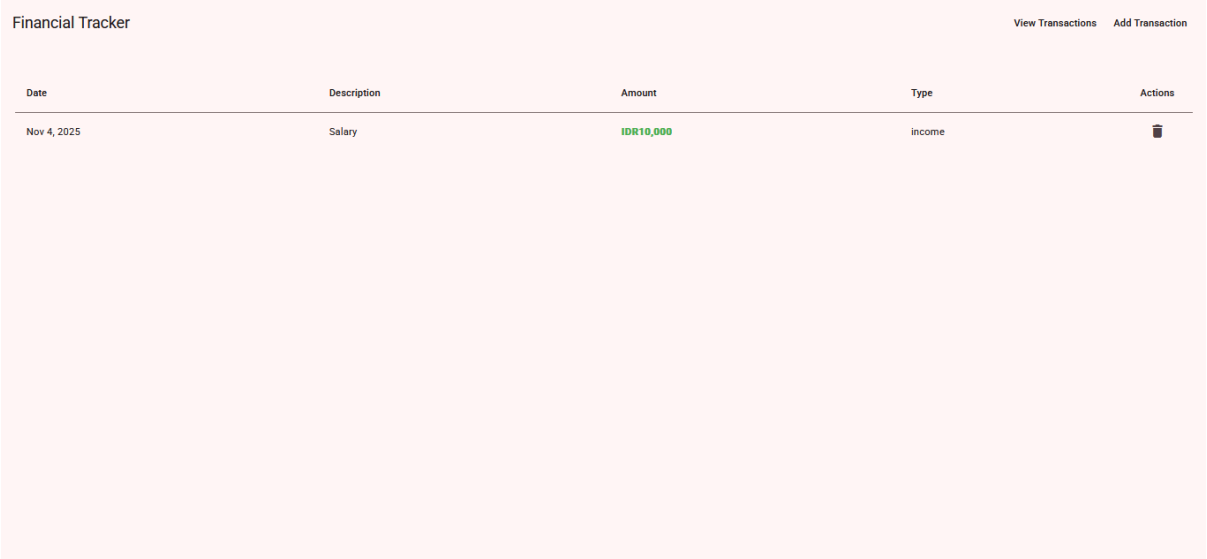
Application Navigation

- **Home/Transactions**: View all saved transactions
- **Add Transaction**: Navigate to form to add new transaction
- **Delete**: Click delete icon on any transaction to remove it

---

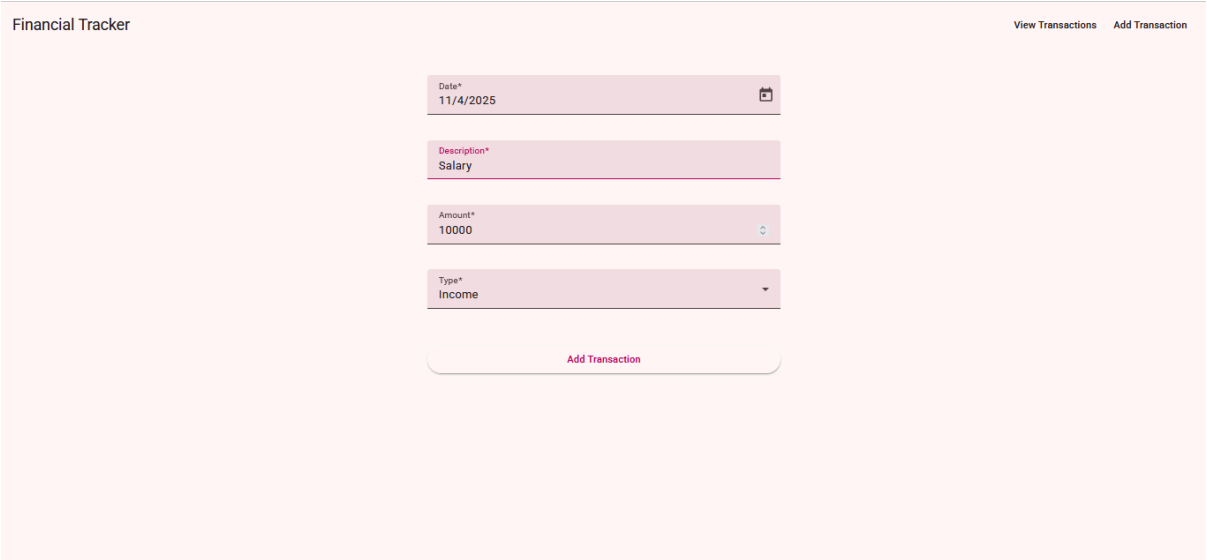
4. Proof of Running Results (Screenshots)

Screenshot 1: Transactions List View



- Material table displaying all transactions
- Each row shows: Date, Description, Amount (IDR), Type
- Delete icon button on each row
- Sorted by date (newest first)

Screenshot 2: Add Transaction Form

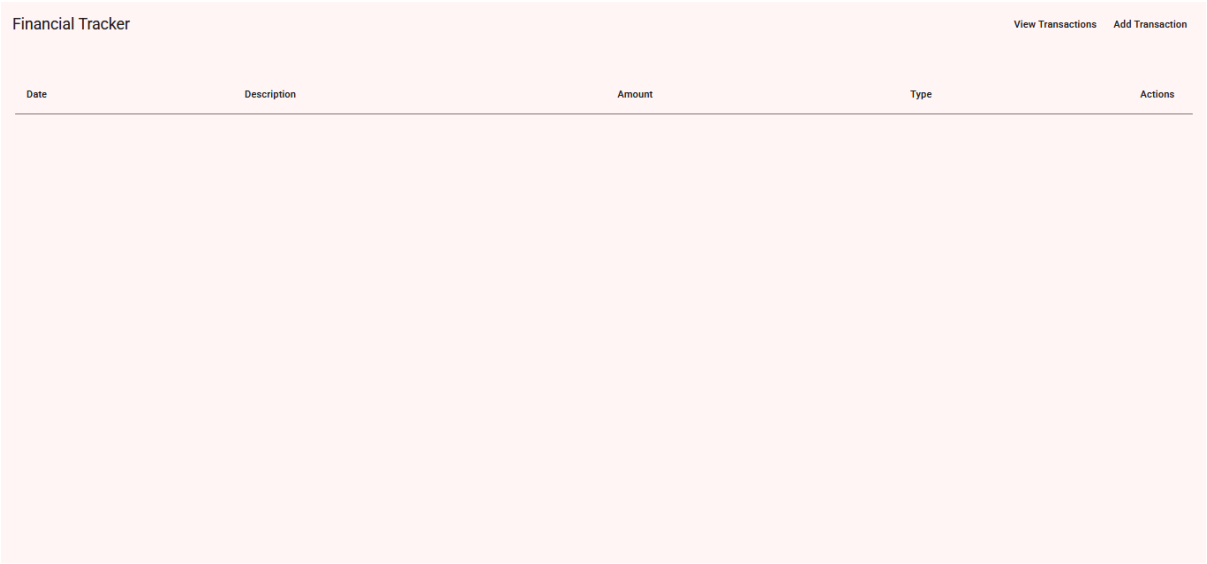


- Form fields: Date Picker, Description, Amount, Type (dropdown)
- Submit button to save transaction
- Form validation prevents empty fields
- Cancel/Back navigation option

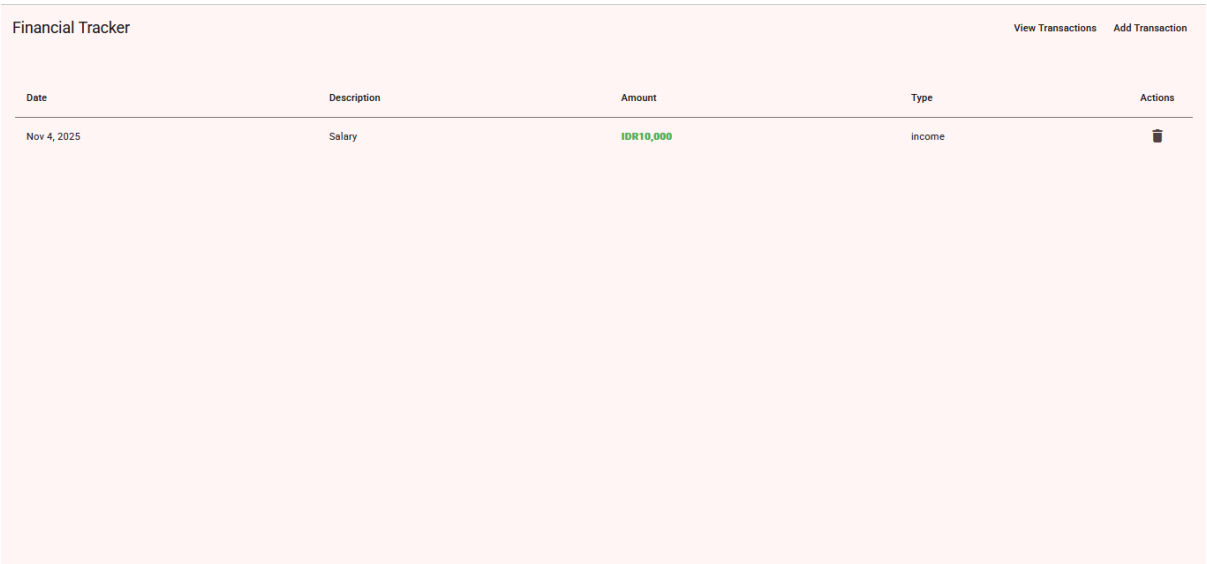
Screenshot 3: Application Toolbar

- Header with "Financial Tracker" branding
- Navigation buttons: "View Transactions" and "Add Transaction"
- Material Design styling with responsive layout

Screenshot 4: Transaction Persistence



before refresh after input



data update after refresh

- Demonstrates data persisting after page refresh
- Shows transactions stored in browser localStorage
- Example transactions in IDR currency format

---

5. Key Technical Details

Data Model

```
interface Transaction {
  id: number;
  date: string;
  description: string;
  amount: number;
  type: 'income' | 'expense';
}
```

Routing Configuration (src/app/app.routes.ts)

/transactions → TransactionList Component

/add → AddTransaction Component

(default) → redirects to /transactions

Locale Configuration

Indonesian locale registered in src/app/app.ts for proper IDR currency formatting.

---

6. Build & Deployment

Build for Production

ng build

- Outputs optimized build to dist/ directory

- Ready for deployment to static hosting

Run Tests

npm test

- Executes unit tests using Karma

Conclusion

The Financial Tracker application successfully delivers a functional, user-friendly solution for personal finance management. The implementation demonstrates:

- ✅ Clean component-based architecture
- ✅ Persistent data storage with localStorage
- ✅ Material Design UI principles
- ✅ TypeScript type safety
- ✅ Responsive and intuitive user interface
- ✅ Indonesian locale and IDR currency support

The application is production-ready and can be extended with additional features such as transaction editing, CSV export, and advanced filtering.

Clone the repository

git clone https://github.com/MarchianTz/financial_tracker.git

cd financial_tracker