💰Money Manager – Full Development Checklist

(React + Spring Boot + MySQL)

This document is a **complete, step-by-step development plan** describing **what the developer has to do** to build the project from scratch.

> All generated code was **reviewed, modified, tested, and integrated manually**.

This project **follows the same feature scope and learning objectives** as a full-length guided tutorial, but was **implemented independently using a structured plan**.

🏮 Development Principles

- Project scope defined before coding
- Backend-first development approach
- Each module implemented and tested independently
- Frontend developed after stable APIs
- Code written, tested, and maintained by the developer

✂️Backend – Project Introduction & Initialization

Project Scope

The application includes:

- User authentication and profile management
- Income & expense tracking
- Category-based analytics
- Dashboard insights
- Email notifications and reports
- File upload and data export

✂️Backend – Project Initialization

Step 1.1 Create Spring Boot Project

Developer instruction:

```
// Create a Spring Boot project with Web, Security, JPA, MySQL, Lombok
```

Dependencies:

- Spring Web
- Spring Security
- Spring Data JPA
- MySQL Driver
- Lombok
- Validation

✒️Backend – Base Configuration

Step 2.1 application.yml

Developer instruction:

```
# Configure MySQL datasource, JPA, Hibernate, server port
```

🛏️Backend – Entity Layer (MODEL)

Step 3.1 User Entity

File: `User.java`

Developer instruction:

```
// JPA entity for User with id, name, email, password, createdAt
```

Step 3.2 Category Entity

File: `Category.java`

Developer instruction:

```
// JPA entity for Category with id, name, type(INCOME, EXPENSE), user
relationship
```

Step 3.3 Transaction Entity

File: `Transaction.java`

Developer instruction:

```
// JPA entity for Transaction with amount, description, type, date, user,
category
```

🌂 Backend – Repository Layer

Step 4.1 UserRepository

Developer instruction:

```
// JPA repository for User with findByEmail method
```

Step 4.2 CategoryRepository

Developer instruction:

```
// JPA repository for Category filtered by user
```

Step 4.3 TransactionRepository

Developer instruction:

```
// JPA repository for Transaction filtered by user and date
```

🌡️ Backend – DTO Layer

Step 5.1 Auth DTOs

Developer instruction:

```
// DTOs for LoginRequest, RegisterRequest, AuthResponse
```

Step 5.2 Transaction DTOs

Developer instruction:

```
// DTOs for creating and returning Transaction data
```

🎒Backend – Security

Step 6.1 JWT Utility

Developer instruction:

```
// Utility class for generating and validating JWT tokens
```

Step 6.2 Security Configuration

Developer instruction:

```
// Spring Security configuration with JWT filter and stateless session
```

🎓Backend – Service Layer

Step 7.1 AuthService

Developer instruction:

```
// Service for user registration and login with password encryption
```

Step 7.2 CategoryService

Developer instruction:

```
// Service for CRUD operations on categories for logged-in user
```

Step 7.3 TransactionService

Developer instruction:

```
// Service for CRUD operations on transactions and summary calculations
```

🎩Backend – Controller Layer

Step 8.1 AuthController

Developer instruction:

```
// REST controller for registration, login, JWT authentication
```

Step 8.2 CategoryController

Developer instruction:

```
// REST controller for category CRUD APIs
```

Step 8.3 TransactionController

Developer instruction:

```
// REST controller for income and expense APIs with validation
```

Step 8.4 DashboardController

Developer instruction:

```
// REST controller that returns dashboard analytics and summaries
```

Step 8.5 FilterController

Developer instruction:

```
// REST controller for filtering transactions by date, type, category
```

Step 8.6 NotificationController

Developer instruction:

```
// REST controller for email notifications and daily reminders
```

Step 8.2 CategoryController

Developer instruction:

```
// REST controller for category CRUD APIs
```

Step 8.3 TransactionController

Developer instruction:

```
// REST controller for transaction CRUD APIs
```

Step 8.4 DashboardController

Developer instruction:

```
// REST controller that returns income, expense, balance summary
```

🎽 Frontend – React Initialization

Frontend Libraries Used

- Axios
- Tailwind CSS
- React Hot Toast
- Lucide React Icons
- Emoji Picker
- Chart Library (Recharts / Chart.js)

🎽 Frontend – React Initialization

Step 9.1 Create React App

Developer instruction:

```
// Initialize React app with routing and axios
```

🕐 Frontend – Auth Flow

Step 10.1 Auth Context

Developer instruction:

```
// React Context for authentication with JWT storage
```

Step 10.2 Login Page

Developer instruction:

```
// Login form that calls backend auth API
```

Step 10.3 Register Page

Developer instruction:

```
// Registration form connected to backend
```

✂️ Frontend – Core Pages

Dashboard Page

Developer instruction:

```
// Dashboard page showing totals and charts
```

Transactions Page

Developer instruction:

```
// Page for listing, adding, editing, deleting transactions
```

Categories Page

Developer instruction:

```
// Page for managing income and expense categories
```

✂️✒️ Charts & Analytics

Developer instruction:

```
// Charts for income vs expense, category-wise breakdown, monthly analysis
```

✂️✒️ Additional Advanced Features

File Export & Email

Developer instruction:

```
// Backend API to export transactions as Excel and send via email
```

```
// Frontend UI to download Excel and trigger email sending
```

Profile Features

Developer instruction:

```
// Upload profile picture using cloud storage
```

Notifications

Developer instruction:

```
// Scheduled daily email reminders for expense tracking
```

✂️🛏️Finalization

- Route protection
- Error handling
- Logout
- UI cleanup

🔗Project Summary

This project demonstrates:

- Full-stack system design
- Secure authentication using JWT
- RESTful backend APIs
- Data validation and filtering
- Real-world features like email, file export, and notifications
- Clean frontend architecture with reusable components

Feature Coverage

The project includes:

- Backend authentication & security
- Category, income, expense, and dashboard APIs
- Filtering, notifications, and reporting
- Complete React frontend with charts, modals, and UX enhancements
- Cloud-based deployment

📌*This documentation describes the exact steps followed by the developer to build the project.*