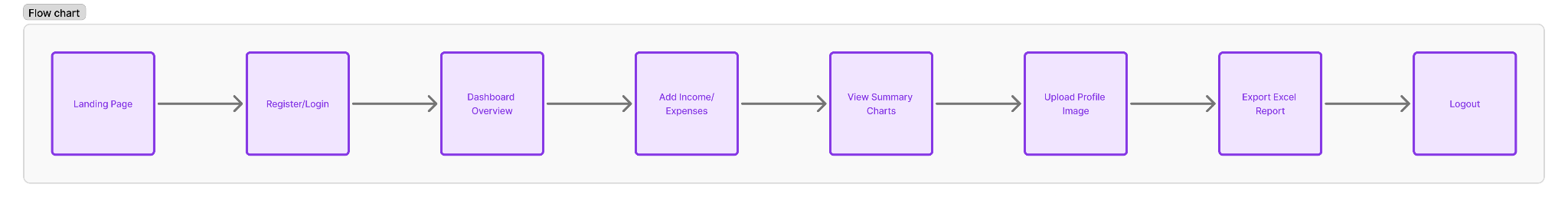
# REQUIREMENT ANALYSIS

## 3.1 Customer Journey Map

The customer journey is structured to provide a seamless experience for individuals tracking their personal finances. The following map outlines each touchpoint from account creation to data analysis and logout.

1. Landing Page → 2. Register/Login → 3. Dashboard Overview → 4. Add Income/Expenses → 5. View Summary Charts → 6. Upload Profile Image → 7. Export Excel Report → 8. Logout  
  
  


This flow is designed to ensure ease of access, visual clarity, and secure handling of sensitive data.  
  
3.2 Solution Requirements

The solution is broken into functional and non-functional requirements:

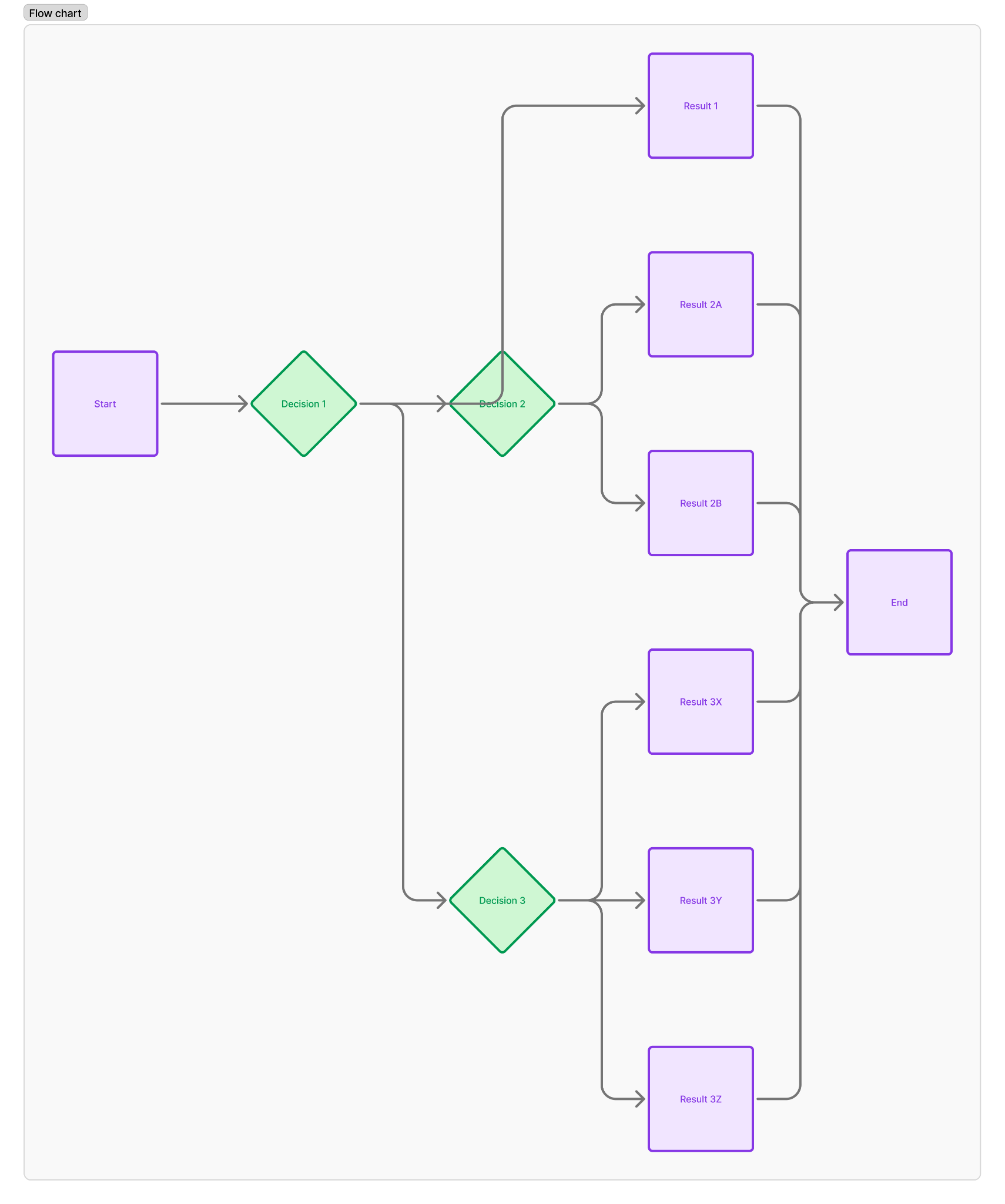
### Functional Requirements

- User Authentication: Secure login/logout using JWT.  
- Dashboard Interface: Real-time overview of income, expenses, and balance.  
- Transaction Management: Add, edit, delete income and expense entries.  
- File Upload: Profile image support using multer.  
- Data Export: Generate Excel reports of financial data.  
- Responsiveness: Mobile-friendly and adaptive UI layout.

### Non-Functional Requirements

- Security: All user data should be encrypted, and routes protected with middleware.  
- Performance: Application should load within 2 seconds.  
- Scalability: Modular codebase to support future multi-user role management.  
- Reliability: Backend API must handle failure gracefully and show relevant errors to users.  
- Maintainability: Clean code structure using MVC patterns and environment configurations.

## 3.3 Data Flow Diagram

The following describes the simplified data flow in the Expense Tracker system:  
  
1. User initiates login via React frontend → API call to Express backend  
2. Backend validates credentials using bcrypt → JWT token issued  
3. User dashboard makes multiple calls to fetch expense/income summaries  
4. Backend aggregates MongoDB collections and sends back data  
5. User performs transactions → API updates MongoDB  
6. Requests like Excel export or image upload follow their dedicated endpoints  
  
This modular flow ensures separation of concerns and scalability.  
  
  


## 3.4 Technology Stack

### Frontend

- React.js: SPA architecture using Vite  
- Tailwind CSS: Utility-first CSS for responsive layout  
- Recharts: For rendering dynamic charts  
- Axios: API requests with interceptor support

### Backend

- Node.js & Express.js: REST API framework  
- MongoDB with Mongoose: NoSQL document storage  
- Multer: Middleware for handling multipart/form-data (file uploads)  
- JSON Web Tokens (JWT): Stateless authentication  
- XLSX: For reading and writing Excel sheets

### DevOps & Utilities

- Postman: API testing and validation  
- GitHub: Version control and collaboration  
- MongoDB Compass: GUI for managing and visualizing MongoDB data  
- Moment.js: Date formatting for transactions