Oversight

By

Zain Afzal

Rafia Anwar

Ali Shariatmadari

Table of Contents

Requirements Specification

1. Introduction

Oversight is a comprehensive financial tracking application designed for Canadian users. It allows users to monitor and analyze their financial assets, track spending patterns, and plan for financial scenarios. This document provides a detailed technical specification of the system, including its architecture, key features, technologies used, and data model.

2. Purpose and Scope

Oversight is a standalone personal finance management tool that integrates seamlessly with financial institutions through secure API connections. Unlike existing tools, Oversight prioritizes transparency, customizability, and real-time insights. It combines advanced financial modeling with user-friendly interfaces to provide actionable recommendations tailored to individual financial goals.

2.1 Key Features

Oversight provides the following core functionalities:

- Securely link financial accounts and retrieve account balances.
- Visualize net worth breakdowns with detailed categorizations.
- Categorize spending to highlight habits and trends.
- Simulate the impact of major purchases on future net worth.
- Optimize debt repayment strategies to minimize interest.
- Generate short-term and long-term financial forecasts.

2.2 User Characteristics

Oversight is targeted toward:

- Individuals with basic financial literacy who want to improve their financial decision making.

- Users with multiple financial accounts seeking a consolidated view of their finances.
- People interested in optimizing their debt and asset allocation for minimal interest.
- Users who prefer intuitive, visually appealing interfaces for financial analysis.
- Tech-savvy individuals comfortable with connecting accounts and managing digital financial tools.

3. System Architecture

Oversight is built using a modern tech stack with a focus on security, scalability, and performance.

3.1 Technology Stack

- Frontend: Next.js (React/TypeScript) with a responsive dark-themed UI
- Backend: Supabase (PostgreSQL database with authentication and API)
- Authentication: Supabase Auth with row-level security policies
- Data Storage: PostgreSQL database with structured tables for financial data
- State Management: React Context for global state management
- Styling: Tailwind CSS with custom styling for dark theme
- UI Components: Radix UI for accessible components
- Data Visualization: Recharts for charting financial data

3.2 Architecture Components

- Client-Side Application: Next.js frontend that communicates directly with Supabase
- Database: Structured PostgreSQL database with row-level security policies
- Authentication Layer: Supabase Auth for secure user authentication
- API Layer: Supabase API for database operations

4. Data Model

The data model is designed to store and organize financial data in a structured way.

4.1 Main Entities

4.1.1 Users

- Stores user account information including personal details
- Linked to auth.users table in Supabase

4.1.2 Accounts

- Central table representing various financial accounts
- Contains common fields like account name, type, and balance
- Used for high-level account management

4.1.3 Specialized Account Types

- Bank Accounts: Traditional financial accounts with transactions
- Crypto Wallets: Cryptocurrency holdings
- Investment Accounts: Stock, ETF, and other investment assets
- Vehicles: Vehicle assets with depreciation tracking
- Real Estate: Property assets with appreciation tracking

4.1.4 Transactions

- Detailed financial transactions across all account types
- Different transaction tables for different account types:
  - Bank account transactions
  - Crypto wallet transactions
  - Investment transactions

4.2 Relationships

- One-to-many relationship between Users and Accounts
- One-to-many relationship between Accounts and Transactions
- Each specialized account type references the main Accounts table
- All entities are secured with row-level security policies

5. Functional Requirements

5.1 Authentication & User Management

- Secure user registration and login

- Password reset and account management
- Row-level security to ensure users only access their own data

5.2 Account Management

- Create, update, and delete financial accounts
- Support for multiple account types (bank, crypto, investment, etc.)
- View account details and balances

5.3 Transaction Management

- Import transactions via CSV upload
- Manual transaction entry
- Transaction categorization
- View and filter transaction history

5.4 Asset Tracking

- Track vehicles with depreciation modeling
- Track real estate with appreciation modeling
- Record and update asset values

5.5 Financial Analysis

- Calculate net worth across all assets
- Visualize financial trends over time

5.6 Dashboard

- Comprehensive dashboard with key financial indicators
- Customizable time-range views (1M, 3M, 6M, 1Y, 2Y, ALL)
- Visual representations of financial data

6. Non-Functional Requirements

6.1 Security

- Row-level security policies on all database tables
- Secure authentication using Supabase Auth
- HTTPS for all client-server communications
- Encryption of sensitive financial data

6.2 Performance

- Responsive UI design for all device sizes
- Efficient database queries for large transaction sets
- Optimized data loading with pagination where appropriate

6.3 Usability

- Intuitive navigation and interaction patterns
- Clear data visualization
- Responsive design for desktop and mobile use

6.4 Privacy & Compliance

- Data minimization and purpose limitation
- User control over their data with export/delete options

6.5 Scalability

- Architecture designed to handle growing user base
- Database structure optimized for increasing data volume
- Efficient data storage and retrieval mechanisms

7. User Interface

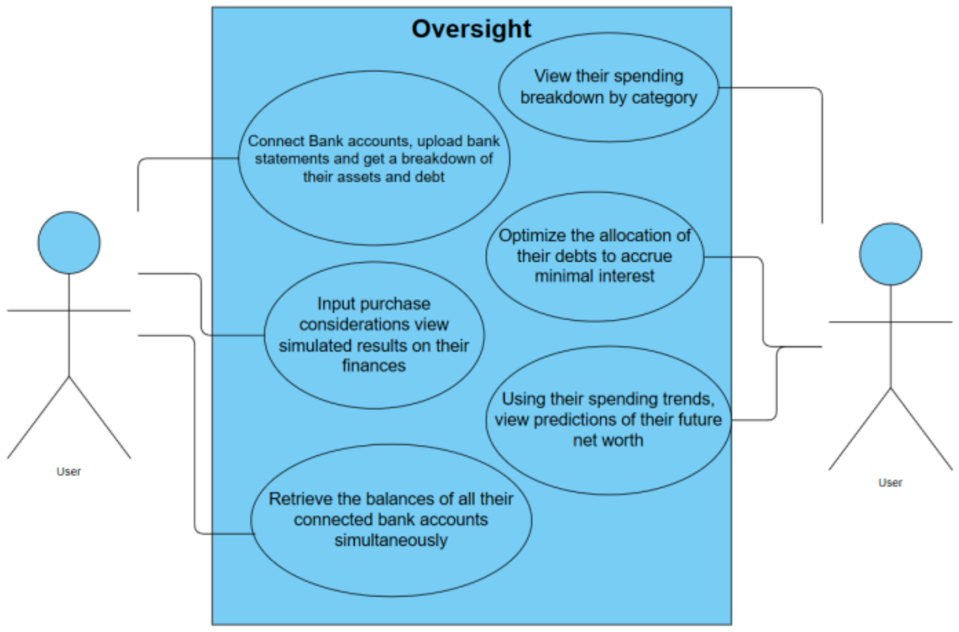
7.1 Design Principles

- Data-dense visualizations with clear information hierarchy
- Consistent navigation and interaction patterns
- Responsive layout for all device sizes

7.2 Key Screens

- Dashboard with financial overview
- Account management screens
- Transaction history and filters
- Asset management for vehicles and real estate
- Financial analytics and visualizations

8. Final Use Case Diagram


Oversight
View their spending breakdown by category
Connect Bank accounts, upload bank statements and get a breakdown of their assets and debt
Optimize the allocation of their debts to accrue minimal interest
Input purchase considerations view simulated results on their finances
Using their spending trends, view predictions of their future net worth
Retrieve the balances of all their connected bank accounts simultaneously
User
User

9. Assumptions, Exceptions, and Variations

9.1 Assumptions

1. Users have online access to their bank accounts and financial institutions support secure API connections.
2. Users possess the necessary credentials and permissions to link their financial accounts with the app.
3. Financial institutions provide consistent and reliable data formats for account information retrieval.
4. Users can accurately input details about their assets such as cars and properties.
5. The app can securely store and process users' financial data in compliance with data protection laws.
6. Users consent to sharing their financial information with the app for analysis and optimization purposes.

9.2 Exceptions

1. Users are unable to connect their bank accounts due to unsupported financial institutions or lack of integration capabilities.

2. API downtime or errors from financial institutions prevent data retrieval and synchronization.
3. Users provide incomplete or incorrect information about their assets, leading to inaccurate analyses.

9.3 Variations
1. Users manually input their financial data instead of connecting accounts automatically.
2. Some users focus solely on tracking net worth without utilizing debt optimization features.
3. Users may prefer different visualization formats for their financial data (e.g., charts, graphs, tables).

10. Future Enhancements

10.1 Planned Features

- Integration with banking APIs for automatic transaction import
- AI-powered transaction categorization
- Advanced financial scenario simulation
- Mobile application
- Expanded investment tracking and performance analysis

Technical Design Documentation

1. System Architecture

1.1 High-Level Architecture

Oversight follows a modern client-server architecture with Next.js as the frontend framework and Supabase as the backend-as-a-service platform.

1.2 Component Architecture

The application follows a component-based architecture with several key subsystems:

1.2.1 Frontend Components

- Authentication Module: Handles user login, registration, and session management
- Dashboard Module: Provides overview of user's financial situation
- Account Management Module: Handles various financial accounts
- Transaction Management Module: Processes and displays financial transactions
- Asset Tracking Module: Manages vehicles, real estate, and other assets
- Analytics Engine: Generates financial reports and visualizations
- Budget Management Module: Handles creation and tracking of budgets

1.2.2 Backend Components

- Supabase Authentication: Manages user identity and access tokens
- Database Layer: PostgreSQL with row-level security policies
- API Layer: RESTful endpoints exposed via Supabase

1.3 Data Flow

1. User authenticates via Supabase Auth

2. Client receives JWT token for subsequent authenticated requests

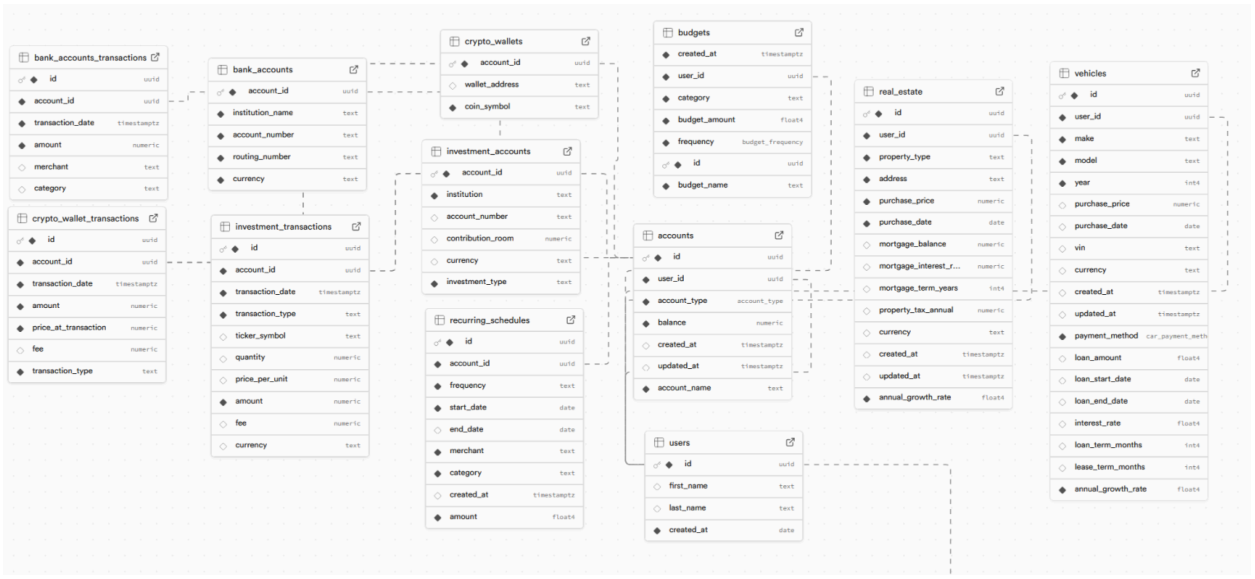
3. Client interacts with Supabase APIs directly using React hooks and contexts

4. Data is fetched from or written to PostgreSQL database via Supabase

5. Row-level security policies ensure data isolation between users

2. Database Design

2.1 Entity Relationship Diagram



2.2 Table Definitions

2.2.1 Users Table

The Users table stores basic information about each user and establishes a connection to Supabase's authentication system. It includes fields for the user's unique identifier, first name, and last name. The user ID serves as the primary key and references the built-in Supabase auth.users table, ensuring that when a user is deleted from the authentication system, their data is also removed from our application.

2.2.2 Accounts Table

The Accounts table functions as a central registry for all financial accounts in the system. Each record represents a user's financial account and contains information such as account name, type (bank, crypto, credit, etc.), and current balance. The table includes creation and update timestamps to track when changes occur. Every account is linked to a specific user via the user_id field, which is crucial for implementing row-level security.

2.2.3 Bank Accounts Table

The Bank Accounts table extends the core Accounts table with bank-specific details. Each bank account references an entry in the main Accounts table and adds fields like institution name, account number, routing number, and currency (defaulting to CAD for Canadian users). This table maintains a copy of the current balance for quick access, though the authoritative balance is calculated from transactions.

2.2.4 Bank Account Transactions Table

The Bank Account Transactions table records individual financial transactions for bank accounts. Each transaction includes details such as the date it occurred, the amount (positive for deposits, negative for withdrawals), merchant information, and a category for expense classification. Each transaction is linked to a specific bank account, allowing for account-specific transaction histories and balance calculations.

Similar specialized tables exist for other account types like crypto wallets and investment accounts, each with fields tailored to the specific characteristics of those financial instruments.

2.3 Row-Level Security Policies

Every table in the database implements row-level security (RLS) policies to ensure that users can only access their own data. For the Accounts table, a policy ensures that users can only view, add, modify, or delete accounts that they own, as determined by the user_id field matching their authenticated user ID. For tables that don't directly contain a user_id field (like transactions), policies use EXISTS clauses to check the ownership chain through parent tables, ensuring that transactions can only be accessed if they belong to accounts owned by the authenticated user.

3. Key Interaction Patterns

Time Range Selection: Consistent pattern across charts using time range selectors (1M, 3M, 6M, 1Y, 2Y, ALL)

Data Filtering: Dropdown and toggle filters for transactions and reports

Modal Forms: For adding/editing accounts, transactions, and assets

Context Menus: For actions on accounts, transactions, etc.

Toast Notifications: For operation feedback (success/error messages)

4. Frontend Implementation

4.1 State Management

The application uses React Context for global state management:

AuthContext: Manages user authentication state, including login status, user profile, and authentication methods.

AccountsContext: Manages financial accounts data, providing functions to fetch, create, update, and delete accounts across all types.

AssetsContext: Manages vehicle and real estate assets, with methods to calculate value based on purchase price and depreciation/appreciation rates.

TransactionsContext: Manages transaction data across account types, enabling filtering, sorting, and categorization capabilities.

4.2 Key Components

4.2.1 Networth Component

The Networth component displays the user's total financial worth over time. It combines a headline figure showing the current net worth with a visualization of how that value has changed over the selected time period. The component includes:

- A card layout with a clear title and the net worth amount prominently displayed
- A time range selector allowing users to view data over different periods (1M, 3M, 6M, 1Y, 2Y, ALL)
- An area chart that visualizes net worth fluctuations, with tooltips showing specific values on hover
- Color coding to indicate positive and negative trends

The component uses the data transformation utilities to calculate net worth by combining all account balances, asset values, and adjusting for liabilities.

4.2.2 Transaction Table Component

The Transaction Table component provides a detailed view of financial transactions across accounts. It features:

- A sortable, filterable table of transactions
- Column headers for date, merchant/description, amount, category, and account

- Filter controls allowing users to narrow transactions by date range, amount, category, or search text
- Color-coded amounts (green for income, red for expenses)
- Pagination for handling large transaction sets
- Action buttons for editing, categorizing, or deleting transactions

The component connects to the TransactionsContext to retrieve transaction data and apply filters, ensuring efficient data loading and rendering.

4.3 Data Fetching and Mutation

The application uses custom hooks to interact with Supabase. For example, the bank accounts hook handles:

1. Fetching bank accounts data for the authenticated user

2. Managing loading and error states during data retrieval

3. Establishing relationships between bank accounts and their parent account records

4. Providing methods to create, update, and delete bank accounts

5. Refreshing data when underlying records change

The hook encapsulates the database query logic, forming a clean boundary between the UI components and the data layer. Similar hooks exist for all major data entities, each tailored to the specific data structure and operations needed for that entity type.

5. Backend Implementation

5.1 Supabase Configuration

5.1.1 Authentication

Supabase Auth is configured with:

- Email/password authentication
- JWT token-based sessions
- Row-level security policies for data access

5.1.2 Database Triggers

The database includes several triggers to maintain data consistency. For example, the account balance update trigger automatically recalculates and updates an account's balance whenever transactions are added, modified, or deleted. This ensures that account balances always reflect the sum of their associated transactions without requiring manual updates.

Other triggers handle tasks like:

Updating the "updated_at" timestamp when records change

Maintaining referential integrity across related tables

Implementing business logic rules (e.g., ensuring transaction dates fall within valid periods)

5.2 API Endpoints

All API endpoints are managed through Supabase's RESTful interface:

5.2.1 Account Endpoints

The system provides endpoints for managing accounts:

- Retrieving all accounts for the authenticated user
- Creating new accounts with basic information
- Updating existing account details
- Deleting accounts when no longer needed

Similar endpoints exist for bank_*accounts, crypto*_wallets, and other specialized account types, each with appropriate validations and business logic.

5.2.2 Transaction Endpoints

Transaction management endpoints include:

- Retrieving transactions with filtering options
- Adding new transactions to accounts
- Updating transaction details (amount, date, category, etc.)
- Deleting transactions when needed

These endpoints enforce business rules like preventing negative balances in certain account types or ensuring transaction categories are valid.

6. Security Implementation

6.1 Authentication Flow

The authentication process follows these steps:

1. User registers or logs in using email/password through Supabase Auth

2. Supabase generates a JWT token containing the user's ID and role

3. JWT token is stored securely in the client (httpOnly cookie)

4. Token is used to authenticate all subsequent API requests

5. Token includes user ID which is used by RLS policies to filter data

6.2 Row-Level Security Details

The database implements a comprehensive security model using row-level security policies. For the Users table, a simple policy ensures users can only access their own profile information by checking that the record's ID matches the authenticated user's ID.

For related tables like transactions, more complex policies are used. These policies check the ownership chain - verifying that the transaction belongs to an account, which in turn belongs to the authenticated user. This creates a security boundary where users can never access data they don't own, even if they attempt to manipulate API requests.

7. Future Technical Roadmap

7.1 API Integrations

Future integration plans include:

- Banking APIs: Direct connections to financial institutions, using Plaid

- Cryptocurrency APIs: Live data from exchanges and blockchain
- Property Valuation APIs: Automated real estate value updates

7.2 Advanced Features

Planned technical improvements:

- Machine Learning: For transaction categorization and spending predictions
- Financial Planning Models: For sophisticated scenario planning
- Data Export/Import: For interoperability with financial software
- Mobile Application: React Native version of the application

7.3 External Dependencies

Authentication: Supabase Auth

Database: Supabase PostgreSQL

Storage: Supabase Storage

Hosting: Vercel (Next.js)

Verification and Validation

This section outlines the verification and validation processes for the Oversight financial tracking application. It details the methodologies, test strategies, and quality assurance processes employed to ensure the application meets its functional requirements, performance standards, and security objectives.

1. Verification Methodology

1.1 Verification Approach

The Oversight verification process follows these principles:

- Traceability: All features are traceable to requirements
- Incremental Testing: Testing occurs at each development phase
- Automation: Automated tests are prioritized where feasible
- Coverage Analysis: Code coverage metrics are monitored

1.2 Testing Levels

1.2.1 Unit Testing

- Focus Areas:
  - Utility functions
  - Data transformations
  - Service methods
  - Component logic

1.2.2 Integration Testing

- Focus Areas:
  - Component interactions
  - Context provider functionality
  - API endpoint integration
  - Database operations

1.2.3 End-to-End Testing

- Focus Areas:
  - User registration and login
  - Account creation and management
  - Transaction import and categorization

2. Validation Methodology

2.1 Validation Approach

Validation ensures that the right product is built for user needs. The validation process includes:

- User Acceptance Testing: Testing with real users
- Requirement Validation: Ensuring all requirements are met
- Usability Testing: Evaluating user experience and interface design
- Performance Validation: Confirming the application meets performance targets

2.2 User Acceptance Testing

- Participant Selection: Mix of financially savvy people and typical end users
- Test Scenarios: Real-world financial management tasks
- Feedback Collection: Structured surveys and open-ended feedback
- Success Criteria: User satisfaction ratings, task completion metrics

2.3 Requirements Validation

Each requirement is validated through:

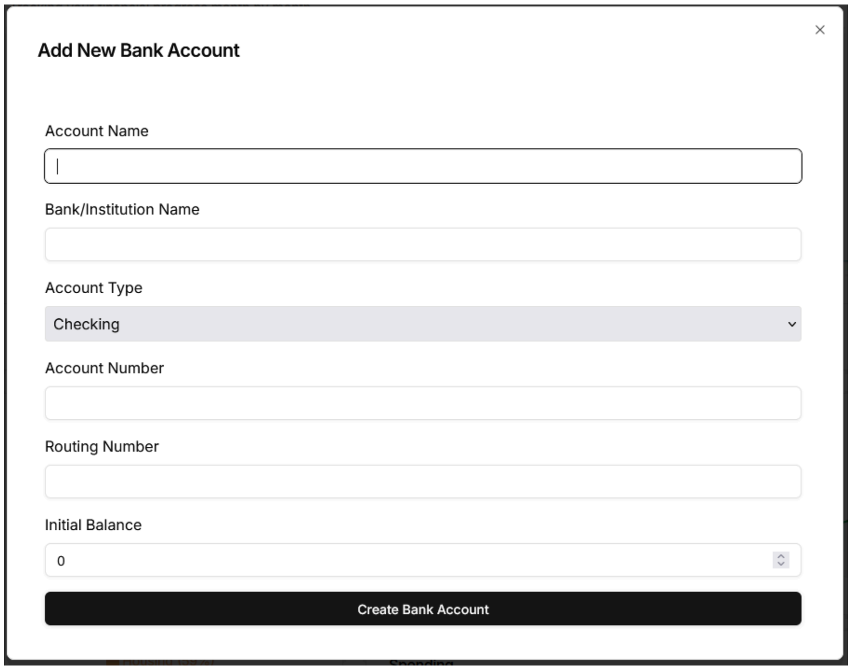
1. Functional Testing: Ensures features work as specified
2. Edge Case Testing: Tests boundary conditions and exceptions
3. Cross-Browser/Device Testing: Validates functionality across platforms

3. User Stories, Scenarios, and Test Cases

3.1.1 User Story: Connect Financial Data and View Net Worth Breakdown

Description: As a user, I want to securely add my financial accounts to the app to view a breakdown of my net worth.

User Interface:



Add New Bank Account

Account Name

Bank/Institution Name

Account Type

Checking

Account Number

Routing Number

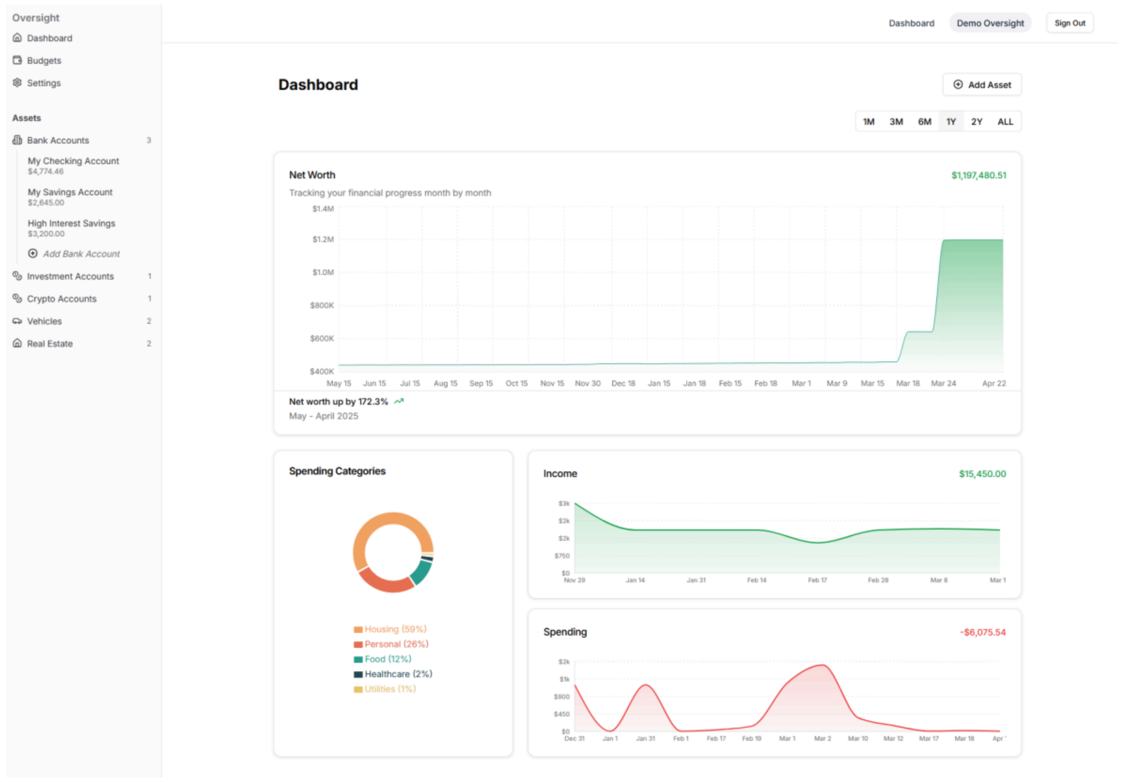
Initial Balance

0

Description: The account adding page allows users to provide bank credentials, initiate a connection, and displays success or error messages.

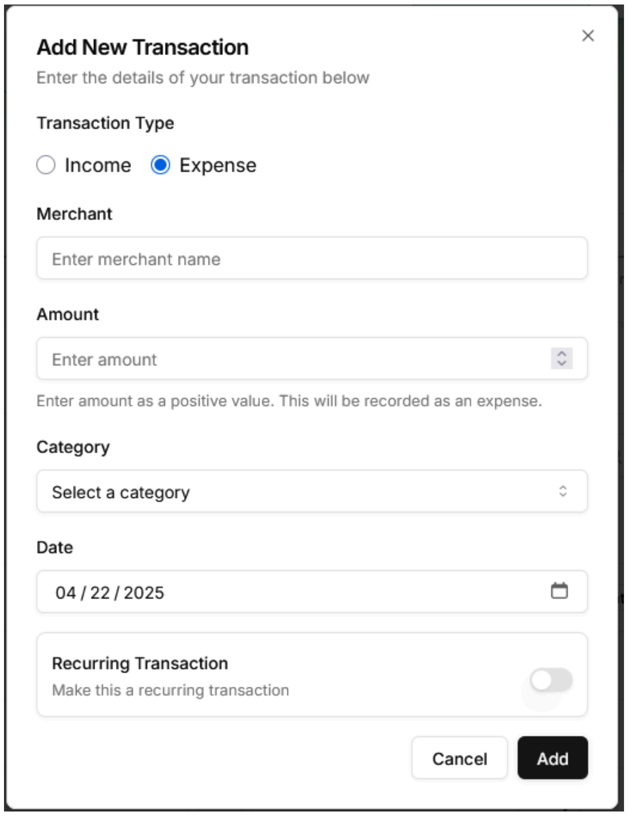
3.1.2 User Story: Retrieve Balances for All Connected Institutions
Description: As a user, I want to view the balances of all my connected financial accounts in one place.

User Interface:



Description: The dashboard shows recent transactions, net worth graphs, and account balances, with options to manually edit or upload data.

3.1.3 User Story: Add Purchase Decisions and Simulate Future Net Worth
Description: As a user, I want to simulate how potential purchases will impact my future net worth.
User Interface:


Add New Transaction

Enter the details of your transaction below

Transaction Type

Income [✓ Expense]

Merchant

Enter merchant name

Amount

Enter amount

Enter amount as a positive value. This will be recorded as an expense.

Category

Select a category

Date

04 / 22 / 2025

Recurring Transaction

Make this a recurring transaction

Description: The simulation page provides input fields for purchase details and displays graphical representations of the impact on net worth.

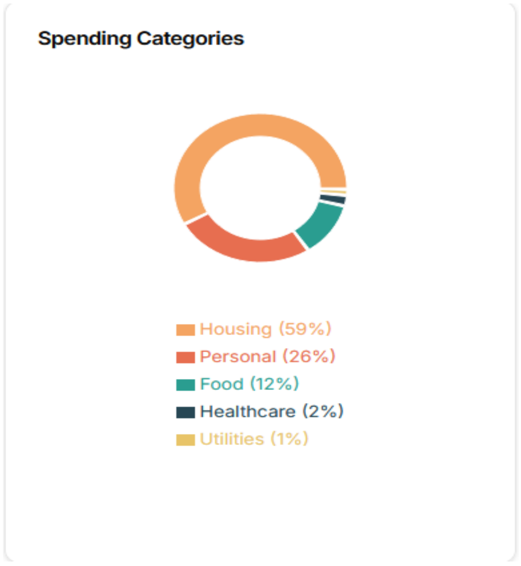
Exceptional Scenarios:

- Inputted data format is incorrect.
  System Behavior: Highlight the input fields in red and notify the user that their data format is invalid and prompt them to input proper data.

3.1.4 User Story: Categorize Spending

Description: As a user, I want to categorize my transactions to better understand my spending habits.

User Interface:



Description: The spending categorization interface displays transactions with dropdown menus for category selection, pre-filled with suggestions.

3.2 Test Cases:

3.2.1 User Authentication and Authorization

Description	Steps	Expected Results
User Registration	1. Navigate to signup page 2. Enter valid credentials 3. Submit form	Account created successfully, user redirected to dashboard
User Login	1. Navigate to login page 2. Enter valid credentials 3. Submit form	User authenticated and redirected to dashboard
Password Reset	1. Click "Forgot Password" 2. Enter email 3. Follow reset instructions	User receives email and can reset password
Unauthorized Access	1. Attempt to access protected route without authentication	User redirected to login page

3.2.1 Account Management

Description	Steps	Expected Results
Create Bank Account	1. Navigate to Accounts 2. Click "Add Account" 3. Enter details 4. Submit form	New account appears in accounts list
Delete Account	1. Select account 2. Click "Delete" 3. Confirm deletion	Account removed, related transactions handled per settings
Account Balance Update	1. Add transactions to account 2. View account details	Balance reflects transaction total correctly

3.2.3 Transaction Management

Description	Steps	Expected Results
Add Manual Transaction	1. Navigate to Transactions 2. Click "Add Transaction" 3. Complete form 4. Submit	Transaction added to account and appears in list
Import CSV Transactions	1. Prepare valid CSV 2. Use import feature 3. Map columns 4. Complete import	All transactions imported correctly
Categorize Transaction	1. Select transaction 2. Change category 3. Save changes	Transaction category updated, analytics reflect change

3.2.4 Asset Tracking

Description	Steps	Expected Results

Add Vehicle Asset	1. Go to Assets 2. Add vehicle with details 3. Submit form	Vehicle added to assets list
Add Real Estate	1. Go to Assets 2. Add property with details 3. Submit form	Property added to assets list
Asset Depreciation	1. Add vehicle with depreciation rate 2. View over time	Value decreases according to rate

3.2.5 Budget Management

Description	Steps	Expected Results
Create Budget	1. Go to Budgets 2. Create new budget 3. Set amount and category 4. Save	Budget created and appears in list
Track Budget Progress	1. Create budget 2. Add transactions in category 3. View	Progress bar updates correctly

3.2.6 Reporting and Analytics

Description	Steps	Expected Results
Net Worth Calculation	1. Add various accounts and assets 2. View dashboard	Net worth calculated correctly
Spending by Category	1. Add categorized transactions 2. View spending report	Categories show correct totals
Time Range Filtering	1. Add transactions across dates 2. Select different time ranges	Data filtered correctly per selection

User Manual and Deployment Information

1. System Requirements

1.1 Server Requirements

- Node.js: v16.14.0 or higher
- PostgreSQL: Latest version (provided by Supabase)
- Storage: Minimum 1GB for application, database size will grow with user data
- Memory: Minimum 2GB RAM recommended

1.2 Client Requirements

Supported Browsers:

- Chrome (latest 2 versions)
- Firefox (latest 2 versions)
- Safari (latest 2 versions)
- Edge (latest 2 versions)

2. Deployment Guide

2.1 Environment Setup

1. Clone the Repository

- *git clone*
- *cd oversight*

2. Set Up Supabase

- Create a new project in Supabase (https://supabase.com)
- Note your project URL and API keys
- Apply the database schema using the SQL files in the `schema/` directory

3. Environment Configuration

- Create a `.env.local` file in the frontend directory based on `.env.example`

- Add your Supabase URL and anon key:

  NEXT_PUBLIC_SUPABASE_URL=your-project-url

  NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key

4. Install Dependencies

- *cd frontend*
- *npm install*

2.2 Local Development

1. Start Development Server

- *npm run dev*

2. Access the Application

- Open http://localhost:3000 in your browser

3 Production Deployment

3.1 Deploying with Vercel

1. Connect Repository to Vercel

- Create an account on Vercel (https://vercel.com)
- Import your GitHub repository
- Configure environment variables (same as in `.env.local`)

2. Deploy

- Vercel will automatically build and deploy your application
- Each push to main branch will trigger a new deployment

3.2 Manual Deployment

1. Build the Application

- npm run build

2. Deploy the Build Output

- Deploy the `.next` directory to your hosting provider
- Configure the server to route all requests to the Next.js application

3. Start the Server

- npm run start

2.4 Database Migration

- Database migrations are managed through SQL scripts in the `schema/` directory
- Apply migrations in order using Supabase's SQL editor
- For schema updates, create new migration files and run them on deployment

3. User Guide

3.1 Account Management

3.1.1 User Registration and Login

1. Registration

- Navigate to the Oversight login page
- Click "Sign Up" to create a new account
- Enter your email and create a secure password
- Complete the verification process via email

2. Login

- Enter your email and password on the login page

3. Password Recovery

- Click "Forgot Password" on the login page
- Follow the instructions sent to your email
- Create a new password when prompted

3.1.2 Profile Management

1. Accessing Your Profile

- Click your name/avatar in the top right corner
- Select "Profile" from the dropdown menu

2. Updating Profile Information

- Edit your name, email, and other details
- Click "Save Changes" to update your profile

3. Changing Password

- Go to profile settings
- Click "Change Password"
- Enter your current password and new password
- Click "Update Password"

3.2 Dashboard Navigation

3.2.1 Main Dashboard

The dashboard provides an overview of your financial situation with several key sections:

1. Net Worth Display

- Shows total net worth calculation
- Includes graphical representation over time
- Use time range selector (1M, 3M, 6M, 1Y, 2Y, ALL) to adjust view

2. Asset Allocation

- Pie chart showing distribution of assets by type
- Hover over sections to see details

3. Monthly Spending

- Bar chart of spending by month and category
- Toggle between absolute values and percentage views

4. Recent Transactions
    - List of most recent financial transactions
    - Click "View All" to see complete transaction history

3.2.2 Sidebar Navigation

The sidebar provides access to all major sections of the application:

- Dashboard: Overview of financial status
- Accounts: Manage financial accounts
- Transactions: View and manage all transactions
- Assets: Track vehicles, real estate, and other assets
- Settings: Configure application preferences

3.3 Account and Transaction Management

3.3.1 Adding Financial Accounts

1. Navigate to Accounts

    - Click "Accounts" in the sidebar

2. Add New Account

    - Click "+ Add Account" button
    - Select account type (Bank, Credit, Investment, etc.)
    - Enter account details (name, institution, balance)
    - Click "Create Account"

3. Account Types
    - Bank Accounts: Checking, savings accounts
    - Credit Cards: Credit accounts with balances
    - Investment Accounts: Stocks, ETFs, retirement accounts
    - Crypto Wallets: Cryptocurrency holdings
    - Other: Custom account types

3.3.2 Managing Transactions

1. Importing Transactions
    - Navigate to the Transactions page

- Click "Import Transactions"
- Select CSV file format
- Map CSV columns to transaction fields
- Review and confirm import

2. Adding Manual Transactions

- Click "+ Add Transaction"
- Select account
- Enter transaction date, amount, merchant, category
- Click "Save Transaction"

3. Editing Transactions

- Click on a transaction in the list
- Update fields as needed
- Click "Save Changes"

4. Categorizing Transactions

- Select transaction
- Click "Categorize" or edit category field
- Select appropriate category from dropdown
- Click "Apply"

3.4 Asset Management

3.4.1 Adding Vehicles

1. Navigate to Assets

- Click "Assets" in the sidebar
- Select "Vehicles" tab

2. Add New Vehicle

- Click "+ Add Vehicle"
- Enter vehicle details (make, model, year, purchase price)
- Specify purchase date and depreciation rate
- Click "Save Vehicle"

3. Updating Vehicle Value

- Select vehicle from list
- Click "Update Value"
- Enter new estimated value
- Click "Save"

3.4.2 Adding Real Estate

1. Navigate to Assets

- Select "Real Estate" tab

2. Add Property

- Click "+ Add Property"
- Enter property details (address, type, purchase price)
- Specify purchase date and appreciation rate
- Add mortgage details if applicable
- Click "Save Property"

3. Updating Property Value

- Select property from list
- Click "Update Value"
- Enter new estimated value
- Click "Save"

3.5 Financial Analytics

3.5.1 Spending Analysis

1. Navigate to Analytics

- Click "Analytics" in the sidebar
- Select "Spending" tab

2. View Spending by Category

- Pie chart shows distribution across categories
- Bar chart shows month-to-month trends
- Table lists top spending categories

3. Filter Options

- Select date range
- Filter by account
- Include/exclude specific categories

3.5.2 Net Worth Tracking

1. View Net Worth Trends

- Line chart shows net worth over time
- Area chart breaks down components (cash, investments, property, etc.)
- Table shows changes month-to-month or year-to-year

2. Analysis Options

- Select time period
- Compare to previous periods
- Exclude specific asset types

4. Troubleshooting

4.1 Common Issues

4.1.1 Authentication Problems

-Issue: Unable to log in

  - Solution: Verify email and password, check for caps lock, try password reset

- Issue: Session expired frequently

  - Solution: Check browser cookie settings, ensure "Remember me" is selected

4.1.2 Data Import Issues

- Issue: CSV import fails

  - Solution: Verify CSV format matches expected format, check for special characters

4.1.3 Performance Issues

- Issue: Slow dashboard loading

  - Solution: Reduce date range, check internet connection, clear browser cache

- Issue: Charts not rendering

  - Solution: Try a different browser, ensure JavaScript is enabled

5. Security Recommendations

5.1 Access Security

- Use strong, unique passwords
- Enable two-factor authentication if available
- Log out when using shared computers
- Regularly review connected devices

5.2 Data Security

- Keep sensitive account information (like account numbers) obscured
- Review transactions regularly for unauthorized activity
- Use the data export feature to create backups periodically

6. Frequently Asked Questions

Q: Is my financial data secure?

A: Yes, Oversight uses industry-standard encryption and security practices. Your data is protected by row-level security in the database, ensuring only you can access your information.

Q: Can I export my data?

A: Yes, you can export your financial data in CSV or PDF format from the Settings page.

Q: How often should I update asset values?

A: For accuracy, update vehicle values quarterly and real estate values annually, or whenever you have new valuation information.

Q: What if my bank isn't supported for direct import?

A: You can manually import transactions using CSV files exported from your bank's website.

Q: How are my account credentials stored?

A: Oversight uses Supabase Auth for authentication. Passwords are hashed and encrypted, never stored in plain text.

Q: Can I share access to my account with my spouse/partner?

A: Currently, each user needs their own account. Family account sharing is on our roadmap for future updates.

Glossary

Net Worth: The total value of all assets minus liabilities

Asset Allocation: The distribution of investments across different asset classes

Budget: A financial plan allocating expected income toward expected expenses

Transaction: A record of money movement (deposit, withdrawal, payment, etc.)

CSV: Comma-Separated Values, a common format for importing/exporting data

RLS: Row-Level Security, the database feature ensuring user data isolation

API: Application Programming Interface, used for data exchange.

Oversight: The name of the personal finance management tool.

Debt optimization: Strategies for efficiently paying off loans.