Personalized Investment Recommendation App

1. Project Overview

Title: Personalized Investment Recommendation App for Bank Customers

Objective: Develop a smart, user-friendly application that analyzes customer financial behavior, assesses risk profiles, and recommends suitable mutual funds and stocks. The app allows users to simulate portfolios, track performance, and improve financial literacy through tips and suggestions.

---

2. Architecture Overview

- **Frontend:** React (styled-components, React Router)
- **Backend:** Flask (Python), modular blueprints
- **Database:** MongoDB (users, transactions, investments, options, recommendations)
- **ML Models:** Scikit-learn (Random Forest, multi-label)

---

3. Frontend Pages & Functionality

3.1 Login Page (`/login`)

- User authentication (email, password)
- Redirects to dashboard or risk profile based on completion

3.2 Register Page (`/register`)

- User registration (name, email, password)
- Redirects to risk profiling questionnaire

3.3 Risk Profiling Questionnaire (`/risk-profile`)

- 7-question form to assess risk level
- Dropdown for investment goal
- Calculates risk score and level (Low/Medium/High)
- Submits profile to backend

3.4 Dashboard (`/dashboard`)

- Shows user overview: name, virtual balance, invested amount, returns
- Displays portfolio table (type, amount, expected return, date)

3.5 Bank Data Upload (`/bank-upload`)

- Upload CSV of bank transactions
- Classifies user as Saver/Spender/Investor
- Displays all transactions in a table

3.6 Investment Page (`/investment`)

- Add new virtual investments (type, company, amount, expected return)
- Company dropdown auto-fills expected return
- List and sell current investments

3.7 Recommendation Page (`/recommendation`)

- **Section 1:** Get recommendations based on financial behavior, risk profile, and goal
- **Section 2:** Get recommendations based on risk, tenure, and capital (with input validation)
- Both sections display results in a unified table

3.8 NotFound Page

- 404 fallback for undefined routes

---

4. Backend Structure

- **app.py:** Flask app initialization, JWT setup, MongoDB connection
- **auth.py:** User login and registration endpoints
- **user.py:** User profile and risk profile endpoints
- **investment.py:** Add, list, and sell investments
- **transaction.py:** Upload and fetch bank transactions, classify financial behavior
- **recommendation.py:**
  - `/api/recommend`: ML-based recommendations (profile/behavior/goal)
  - `/api/recommend-rtc`: ML-based recommendations (risk, tenure, capital)
- **insert_investment_options.py:** Inserts investment options into DB
- **ML Training Scripts:**
  - `train_recommendation_model.py`
  - `train_risk_tenure_capital_model.py`

---

5. Database (MongoDB) Schema

- **users:** User info, risk profile, investment goal
- **bank_transactions:** Credit/debit/investment transactions
- **investment_options:** Stocks and mutual funds metadata
- **investments:** User's virtual investments
- **recommendations:** Saved recommendations
- **financial_tips:** Tips by risk level (optional)

---

6. Machine Learning Model Integration

6.1 Main Recommendation Model

- **Algorithm:** Random Forest (OneVsRestClassifier)
- **Features:** Risk profile, financial behavior, investment goal
- **Endpoint:** `/api/recommend`
- **Training Script:** `train_recommendation_model.py`

6.2 Risk-Tenure-Capital (RTC) Model

- **Algorithm:** Random Forest (OneVsRestClassifier)
- **Features:** Risk level, tenure, capital, capital bin
- **Endpoint:** `/api/recommend-rtc`
- **Training Script:** `train_risk_tenure_capital_model.py`
- **Threshold optimization** for each label

---

7. Features & Logic

- **Risk Profiling:** 7-question form, score-based risk level
- **Financial Behavior Classification:** Rule-based on bank data (Saver/Spender/Investor)
- **Investment Recommendations:** ML-driven, two models (profile/goal and risk/tenure/capital)
- **Simulated Portfolio:** Add/sell investments, track returns
- **Financial Tips:** Personalized tips by risk level (optional)

---

8. How to Run

1. **Backend:**
    - Change directory to backend: `cd backend`
    - Install requirements: `pip install -r requirements.txt`

    - Start Flask app: `python app.py`
2. **Frontend:**
    - Change directory to frontend: `cd frontend`
    - Install dependencies: `npm install`
    - Start React app: `npm start`
3. **MongoDB:**
    - Ensure MongoDB is running and accessible

---

9. Credits

- Frontend: Omkar & Akshay
- Backend: Jayanth & Veena
- Database: Jayanth
- AI/ML: Jayanth & Lakshmi Manasa
