Tech Stack Document

Predictive Analytics for Personal Expenses & Financial Planning

---

1. Introduction

This document defines the **complete technology stack** required to build the *Predictive Analytics for Personal Expenses and Financial Planning* system. The tech stack is designed by referring to: - The **PRD** (functional & non-functional requirements) - The **Design Document & UI inspiration dashboard** - The overall **data analytics and ML-driven nature** of the project

The stack considers **scalability, performance, security, and future enhancements**, while remaining suitable for **academic, portfolio, and production-level implementation**.

---

2. System Architecture Overview

High-Level Architecture

- **Frontend (UI Layer)** – User interaction & visualization
- **Backend (Application Layer)** – Business logic & APIs
- **Analytics & ML Layer** – Predictive modeling & insights
- **Database Layer** – Secure data storage
- **Deployment & DevOps Layer** – Hosting & monitoring

---

3. Frontend Technology Stack

3.1 Core Technologies

- **HTML5** – Page structure
- **CSS3 / Tailwind CSS** – Responsive and modern UI styling
- **JavaScript (ES6+)** – Client-side logic

3.2 Frontend Framework (Recommended)

- **React.js**
- Component-based architecture
- Efficient state management
- Easy integration with analytics dashboards

3.3 UI & Visualization Libraries

- **Chart.js / Recharts / D3.js** – Graphs & charts
- **Material UI / Ant Design** – UI components
- **Lucide / Font Awesome** – Icons

3.4 Frontend Responsibilities

- Dashboard rendering
- Expense & income visualization
- User input forms
- Real-time data updates
- Forecast visualization

---

4. Backend Technology Stack

4.1 Backend Framework

- **Python (Primary Backend Language)**
- **Flask / FastAPI** (Recommended)
- Lightweight and fast
- Ideal for ML-based APIs
- RESTful service support

4.2 API Layer

- REST APIs for:
- Expense management
- Budget planning
- Predictive analytics results
- User authentication

4.3 Backend Responsibilities

- Business logic execution
- Data preprocessing
- Model inference handling
- Secure data access

---

5. Machine Learning & Analytics Stack

5.1 Core Libraries

- **NumPy** – Numerical computation
- **Pandas** – Data manipulation
- **Matplotlib / Seaborn** – Exploratory visualization

5.2 Predictive Modeling

- **Scikit-learn** – Regression & forecasting models
- **Statsmodels** – Time-series analysis (ARIMA, SARIMA)
- **Prophet (Optional)** – Seasonality-based forecasting

5.3 ML Use Cases

- Expense trend prediction
- Monthly & yearly expense forecasting
- Budget deviation detection
- Savings goal completion prediction

---

6. Database Technology Stack

6.1 Relational Database (Primary)

- **PostgreSQL / MySQL**
- Structured financial data storage
- Strong data integrity

6.2 NoSQL Database (Optional)

- **MongoDB**
- Flexible schema for logs & analytics

6.3 Data Stored

- User profiles
- Transactions & expenses
- Budget & goal data
- Prediction results

---

7. Authentication & Security Stack

7.1 Authentication

- **JWT (JSON Web Tokens)**
- **OAuth 2.0** (Future enhancement)

7.2 Security Measures

- Data encryption (at rest & in transit)
- Secure API endpoints
- Role-based access control
- Input validation

---

8. Cloud & Deployment Stack

8.1 Hosting & Deployment Platform (Primary)

- **Vercel**
- Frontend-first cloud platform

- Ideal for React / Next.js deployments
- Automatic CI/CD via GitHub integration
- Global CDN for fast content delivery

8.2 Backend Deployment on Vercel

- **Python APIs via Serverless Functions**
- FastAPI / Flask adapted as serverless endpoints
- Handles business logic & ML inference

8.3 Why Vercel

- Zero infrastructure management
- Fast deployments & rollbacks
- Seamless frontend + backend integration
- Suitable for academic and production demos

---

9. DevOps & Monitoring Tools

9.1 Version Control

- **Git & GitHub** – Source code management

9.2 CI/CD

- **GitHub Actions** – Automated build & deployment

9.3 Monitoring & Logging

- **Prometheus / Grafana** (Optional)
- **CloudWatch / Log Management Tools**

---

10. Testing & Quality Assurance

10.1 Testing Tools

- **PyTest** – Backend testing
- **Postman** – API testing
- **Jest** – Frontend testing

10.2 Testing Types

- Unit testing
- Integration testing
- Model performance testing

---

11. Data Pipeline & Workflow

1. User inputs expense data
2. Backend validates and stores data
3. ML pipeline preprocesses data
4. Predictive models generate forecasts
5. Results are sent to frontend
6. Dashboards update dynamically

---

12. Tech Stack Mapping to PRD

PRD Feature	Technology
Expense tracking	React + Flask + PostgreSQL
Predictive analytics	Python + ML libraries
Budget planning	Backend logic + dashboards
Financial insights	Charts & ML predictions
Security	JWT + encryption

---

13. Scalability & Future Enhancements

- Microservices architecture
- Real-time data ingestion
- AI-driven recommendation engine
- Mobile app (React Native / Flutter)

---

14. Conclusion

This tech stack supports a **robust, scalable, and analytics-driven financial planning system**. It aligns with the PRD and design goals, ensuring the project is suitable for **academic submission, portfolio demonstration, and real-world deployment**.

End of Tech Stack Document