AI FinanceMaster App – Product Requirements Document (PRD)

1. Executive Summary

AI FinanceMaster App is an AI-powered personal finance manager designed for digitally savvy individuals (ages 18-45), freelancers, families, and small business owners. The app connects securely to users' bank, credit card, and investment accounts, automatically categorizing expenses, detecting subscriptions, and providing actionable insights to reduce monthly costs. Key features include real-time alerts, bill reminders, subscription management, and AI-generated plain-English financial summaries. The platform integrates with Plaid, Stripe Billing, Yahoo Finance, OpenAI, Twilio/WhatsApp, credit score APIs, and bill negotiation services. The primary objective is to deliver tangible savings, peace of mind, and proactive financial optimization, positioning the app as an "AI Money Copilot" that empowers users to make informed, cost-saving decisions.

2. Problem Statement

Managing personal finances is complex and time-consuming, especially for millennials, Gen Z, gig workers, and professionals juggling multiple accounts and subscriptions. According to a 2023 Bankrate survey, 74% of Americans have a budget, but 65% fail to stick to it. Users often lose money to duplicate subscriptions, late fees, and suboptimal bill rates. Existing solutions (e.g., Mint, Rocket Money, Credit Karma) are fragmented, reactive, and lack proactive, AI-driven recommendations. Users need a unified, intelligent platform that automates financial tracking, identifies waste, and actively helps reduce costs.

3. Solution Overview

AI FinanceMaster App is an integrated, AI-driven platform that aggregates financial data in real time, categorizes expenses, manages subscriptions, and delivers personalized, actionable recommendations. Unique features include:

- **Automatic Expense Categorization & Budget Tracking**
- **Subscription Detection & Management (with cancellation/negotiation)**
- **Real-time Alerts & Bill Reminders (customizable, via in-app, WhatsApp, SMS)**
- **Natural Language Financial Summaries (OpenAI-powered)**
- **Predictive Spend Warnings & Personalized Budgeting**
- **Automated Bill Negotiation (via BillShark API or similar)**
- **Continuous Learning from User Feedback**
- **Downloadable PDF Reports**
- **Compliance with PCI DSS & GDPR**

The app's proactive, action-oriented approach moves beyond passive reporting, empowering users to optimize and automate their financial lives.

---

4. Stakeholder Analysis

Stakeholder	Role/Responsibility	Influence
End Users	Primary users; provide feedback and drive adoption	High
Product Manager	Vision, roadmap, requirements, prioritization	High
UX/UI Designer	User flows, onboarding, accessibility, usability	High
Engineering Team	Architecture, development, integration, QA	High

Stakeholder	Role/Responsibility	Influence
Compliance Officer	Ensures PCI DSS, GDPR, and financial regulations	High
Marketing Team	User acquisition, onboarding, retention campaigns	Medium
Customer Support	User assistance, issue resolution, feedback channel	Medium
Banking Partners	Data access via Plaid, Stripe, Yahoo Finance	Medium
Regulatory Bodies	Oversight, audits, compliance enforcement	High
Bill Negotiation Partners	Service integration, negotiation workflow	Medium

5. User Personas

Persona 1: "Millennial Urban Professional"

- **Name:** Alex Chen
- **Age/Role:** 29, Product Manager
- **Demographics:** Lives in NYC, works at a tech startup, $90k/year, single, rents apartment
- **Goals:** Automate budgeting, avoid late fees, optimize spending, grow investments
- **Pain Points:** Multiple accounts, forgotten subscriptions, lack of time for manual tracking
- **Use Cases:** Connects all accounts, receives real-time alerts, manages subscriptions, reviews AI-generated summaries weekly
- **Tech Comfort:** High; uses fintech apps, expects seamless UX
- **Behavioral Patterns:** Engages with app daily, prefers in-app and WhatsApp notifications, values actionable insights

Persona 2: “Freelancer/Gig Worker”

- **Name:** Jordan Rivera
- **Age/Role:** 34, Freelance Designer
- **Demographics:** Los Angeles, variable income, multiple clients, uses personal and business cards
- **Goals:** Track irregular income/expenses, avoid overdrafts, minimize recurring costs
- **Pain Points:** Unpredictable cash flow, duplicate subscriptions, missed bill payments
- **Use Cases:** Uses subscription management to cancel unused services, relies on predictive spend warnings, exports PDF reports for taxes
- **Tech Comfort:** Medium-High; uses mobile banking, open to new tools if intuitive
- **Behavioral Patterns:** Checks app before major purchases, customizes alert frequency, prefers SMS for urgent alerts

Persona 3: “Young Family CFO”

- **Name:** Priya Patel
- **Age/Role:** 38, Parent & Project Manager
- **Demographics:** Chicago suburb, married, two kids, manages household finances
- **Goals:** Stay within family budget, avoid late fees, optimize household bills
- **Pain Points:** Multiple shared subscriptions, complex bill schedules, lack of consolidated view
- **Use Cases:** Sets up shared account access, receives bill reminders, uses bill negotiation for utilities, reviews monthly summaries
- **Tech Comfort:** Medium; uses apps for family coordination, values security and privacy
- **Behavioral Patterns:** Reviews weekly summaries, delegates some tasks to spouse, prefers email and in-app notifications

Persona 4: “Small Business Owner”

- **Name:** Samir Gupta

- **Age/Role:** 45, Owner of a boutique marketing agency
- **Demographics:** Boston, 10 employees, manages business and personal finances
- **Goals:** Streamline expense tracking, separate business/personal costs, improve cash flow
- **Pain Points:** Manual reconciliation, missed business expense deductions, fragmented tools
- **Use Cases:** Connects business accounts, categorizes expenses, downloads reports for accountant, uses AI recommendations for cost savings
- **Tech Comfort:** Medium; uses accounting software, expects integrations
- **Behavioral Patterns:** Monthly deep dives, prefers desktop access, values PDF exports

---

6. Technical Requirements

Architecture Alignment:

- **Frontend:** React 18.x (SPA), Material-UI, Redux Toolkit for state management
- **Backend:** Node.js 18.x with Express, RESTful API design, OpenAI API integration
- **Infrastructure:** AWS (EC2, RDS PostgreSQL 15+, S3 for document storage), Docker containers, Terraform for IaC
- **Security:** OAuth 2.0 (Plaid, Stripe), JWT for session management, 256-bit encryption (TLS 1.3), PCI DSS & GDPR compliance
- **Integration:**
  - Plaid API (banking data)
  - Stripe Billing API (subscription detection)
  - Yahoo Finance API (investment tracking)
  - OpenAI API (NLP summaries)
  - Twilio/WhatsApp API (alerts/notifications)
  - Credit Score API (risk analysis)
  - BillShark API (bill negotiation)

- **Database:** PostgreSQL 15+ (see Data Architecture for schema)
- **Component Architecture:**
  - Auth Service
  - Account Aggregation Service
  - AI Recommendation Engine
  - Notification Service
  - Subscription Management Module
  - Bill Negotiation Module
  - Reporting Service
- **Performance:** Support 10,000 concurrent users, <1s response time for dashboard
- **Configuration:** Feature toggles for demo mode, alert customization, and AI feedback loop

Frontend:

- React 18.x, Material-UI, Redux Toolkit, Axios for API calls
- Responsive web app (mobile-first), PWA support
- Accessibility: WCAG 2.1 AA compliance

Backend:

- Node.js 18.x, Express, RESTful API, OpenAI SDK
- Plaid, Stripe, Yahoo Finance, Twilio, BillShark SDKs
- Rate limiting (API Gateway), centralized logging (AWS CloudWatch)

Infrastructure:

- AWS EC2 (auto-scaling), RDS PostgreSQL, S3, CloudFront
- Docker, Terraform, CI/CD (GitHub Actions)

Security:

- End-to-end encryption (TLS 1.3), PCI DSS controls, GDPR data minimization
- 2FA (email/SMS), audit logging, role-based access control

Integration:

- OAuth 2.0 for third-party APIs
- Webhooks for real-time updates (Plaid, Stripe, Twilio)

7. Non-Functional Requirements

Attribute	Requirement
Performance	<1s dashboard load, <2s for report generation, 99.99% uptime, 10,000 concurrent users
Scalability	Horizontal scaling (AWS Auto Scaling), stateless microservices
Reliability	99.99% uptime, automatic failover (multi-AZ), daily backups
Availability	Multi-region AWS deployment, health checks, blue-green deployment
Security	PCI DSS, GDPR, 256-bit encryption, regular penetration testing, 2FA, audit logs
Privacy	Data minimization, user consent management, right to be forgotten, data masking
Compliance	PCI DSS, GDPR, CCPA, SOC 2 Type II
Accessibility	WCAG 2.1 AA
Maintainability	Modular codebase, CI/CD pipeline, automated tests, clear documentation
Usability	NPS > 60, CSAT > 4.5/5, onboarding completion > 80%

8. Success Metrics & KPIs

Metric	Target/Goal	Measurement Method
User Onboarding Completion Rate	>80%	Analytics on onboarding flow
Monthly Active Users (MAU)	50,000+ in 12 months	User activity logs
Subscription Cancellations Initiated	10,000/month	Subscription module logs
Bill Negotiation Success Rate	>30% of initiated requests	BillShark API callbacks
Average Monthly Savings per User	$50+	User-reported + system calc
Alert Engagement Rate	>60%	Notification click-throughs
User Feedback Submission Rate	>20% of active users	Feedback module analytics
NPS (Net Promoter Score)	>60	In-app survey
CSAT (Customer Satisfaction)	>4.5/5	Post-interaction survey
System Uptime	99.99%	AWS CloudWatch

9. Risks

Risk Category	Description	Mitigation Strategy
Security Breach	Exposure of sensitive financial data	PCI DSS controls, regular audits, encryption, 2FA

Risk Category	Description	Mitigation Strategy
Regulatory Non-Compliance	Failure to meet PCI DSS, GDPR, CCPA, SOC 2	Continuous compliance monitoring, legal reviews
Market Adoption	Low user adoption or engagement	User-centric onboarding, demo mode, targeted marketing
System Performance	Downtime or slow response under load	Auto-scaling, load testing, AWS multi-region
Third-Party API Failure	Outages or changes in Plaid, Stripe, BillShark, etc.	Graceful degradation, fallback flows, API monitoring
Data Quality	Inaccurate categorization or recommendations	User feedback loop, AI retraining, manual correction
User Trust	Users hesitant to connect accounts	Transparent security messaging, demo mode, privacy
Integration Complexity	Issues with integrating multiple APIs	Modular architecture, robust API versioning

10. Assumptions

- Users are digitally literate (primary target: 18-45, urban, tech-savvy)
- Plaid, Stripe, Yahoo Finance, BillShark, and other APIs provide reliable, stable integrations
- Users will connect at least one financial account for full functionality
- Users value transparency and control (approval required for automated actions)

- PCI DSS & GDPR compliance is mandatory for all data flows
- User feedback will be sufficient to improve AI recommendations
- Demo mode will increase conversion rates to full account creation
- AWS infrastructure is available and scalable as required
- Users prefer a mix of in-app and external notifications (WhatsApp/SMS)
- Bill negotiation partners (e.g., BillShark) are available in target markets

---

11. Compliance & Regulatory Requirements

- **PCI DSS:** End-to-end encryption, access controls, quarterly ASV scans, annual assessments, incident response plan, data masking, no storage of sensitive authentication data after authorization ([PCI-DSS Compliance][1])
- **GDPR:** User consent, right to be forgotten, data minimization, breach notification, privacy policy ([Blockchain Technology in Financial Services][2])
- **CCPA:** California user data rights, opt-out mechanisms
- **SOC 2 Type II:** Security, availability, processing integrity, confidentiality, privacy
- **Other:** Regular compliance audits, legal review of all integrations, privacy policy disclosures

---

12. Security & Privacy Requirements

- **Data Encryption:** 256-bit AES for data at rest, TLS 1.3 for data in transit
- **Authentication:** OAuth 2.0 for third-party APIs, JWT for sessions, 2FA for user logins
- **Access Control:** Role-based access, least privilege, unique user IDs
- **Audit Logging:** All access and actions on sensitive data logged and monitored
- **Incident Response:** Documented plan, regular drills, rapid breach notification
- **Data Minimization:** Only store data strictly necessary for functionality

- **User Consent:** Explicit opt-in for data collection, clear privacy controls
- **Data Masking:** PAN masking, no storage of sensitive authentication data post-authorization
- **Vulnerability Management:** Regular scans, patch management, penetration testing

---

13. Integration Requirements

- **Plaid API:** Secure banking data aggregation (OAuth 2.0)
- **Stripe Billing API:** Subscription detection and management
- **Yahoo Finance API:** Investment account aggregation and tracking
- **OpenAI API:** Natural language summaries and recommendations
- **Twilio/WhatsApp API:** Real-time alerts and notifications
- **Credit Score API:** Credit risk analysis and monitoring
- **BillShark API:** Automated bill negotiation
- **Webhooks:** For real-time updates from Plaid, Stripe, Twilio
- **API Gateway:** Centralized management, rate limiting, monitoring
- **OAuth 2.0:** For all third-party integrations

---

14. Data Architecture

Architecture Alignment:

- **Database:** PostgreSQL 15+
- **Schema:**
  - `users` (id, email, hashed_password, created_at, last_login, consent_status, notification_preferences)
  - `accounts` (id, user_id, provider, account_type, account_number_masked, balance, currency, last_synced)
  - `transactions` (id, account_id, date, amount, currency, merchant, category, is_subscription, is_unusual, notes)
  - `subscriptions` (id, user_id, merchant, amount, frequency, status, detected_on, last_action)

- `bills` (id, user_id, merchant, due_date, amount, status, negotiation_status)
- `alerts` (id, user_id, type, message, sent_via, sent_at, read_at)
- `ai_recommendations` (id, user_id, recommendation_type, content, created_at, feedback)
- `feedback` (id, user_id, recommendation_id, rating, comment, submitted_at)
- `reports` (id, user_id, period, pdf_url, generated_at)

- **Relationships:**
  - `users` 1:N `accounts`, `subscriptions`, `bills`, `alerts`, `ai_recommendations`, `feedback`, `reports`
  - `accounts` 1:N `transactions`
- **Indexes:**
  - On `user_id` for all tables, on `date` for `transactions`, on `status` for `subscriptions` and `bills`
- **Storage:**
  - S3 for PDF reports and document storage
- **Data Flow:**
  - Plaid/Stripe/Yahoo Finance → Account Aggregation Service → Database
  - AI Recommendation Engine → `ai_recommendations` → User Feedback → Retraining pipeline
  - Notification Service → `alerts` → Twilio/WhatsApp/SMS
  - Reporting Service → `reports` → S3

---

15. API Specifications

API Versioning: `/api/v1/`
Authentication: OAuth 2.0 (third-party), JWT (user sessions)
Rate Limiting: 500 requests/min/user
Error Handling: Standardized error codes, descriptive messages
All endpoints return JSON; all request bodies validated with Joi

Feature-to-Endpoint Mapping

User Authentication & Onboarding

- `POST /api/v1/auth/register`
  - Request: `{ email, password }`
  - Response: `{ userId, token }`
- `POST /api/v1/auth/login`
  - Request: `{ email, password }`
  - Response: `{ userId, token }`
- `POST /api/v1/auth/logout`
  - Request: `{ token }`
  - Response: `{ success }`
- `POST /api/v1/auth/forgot-password`
  - Request: `{ email }`
  - Response: `{ success }`
- `GET /api/v1/demo-mode`
  - Response: `{ features: [ ... ] }`

Account Aggregation

- `POST /api/v1/accounts/connect`
  - Request: `{ provider, oauth_token }`
  - Response: `{ accountId, status }`
- `GET /api/v1/accounts`
  - Response: `{ accounts: [ ... ] }`
- `DELETE /api/v1/accounts/:id`
  - Response: `{ success }`
- `GET /api/v1/accounts/:id/transactions`
  - Query: `?from=YYYY-MM-DD&to=YYYY-MM-DD`
  - Response: `{ transactions: [ ... ] }`

Expense Categorization & Budget Tracking

- `GET /api/v1/transactions`
  - Query: `?category=food&from=YYYY-MM-DD`

  - Response: `{ transactions: [ ... ] }`
- `POST /api/v1/transactions/categorize`
  - Request: `{ transactionId, category }`
  - Response: `{ success }`
- `GET /api/v1/budgets`
  - Response: `{ budgets: [ ... ] }`
- `POST /api/v1/budgets`
  - Request: `{ category, amount, period }`
  - Response: `{ budgetId }`
- `PUT /api/v1/budgets/:id`
  - Request: `{ amount }`
  - Response: `{ success }`

Subscription Detection & Management

- `GET /api/v1/subscriptions`
  - Response: `{ subscriptions: [ ... ] }`
- `POST /api/v1/subscriptions/cancel`
  - Request: `{ subscriptionId }`
  - Response: `{ status, confirmationRequired }`
- `POST /api/v1/subscriptions/confirm-cancellation`
  - Request: `{ subscriptionId, userConfirmation }`
  - Response: `{ status }`
- `GET /api/v1/subscriptions/duplicates`
  - Response: `{ duplicates: [ ... ] }`

Bill Negotiation

- `GET /api/v1/bills`
  - Response: `{ bills: [ ... ] }`
- `POST /api/v1/bills/negotiate`
  - Request: `{ billId }`
  - Response: `{ negotiationStatus }`
- `GET /api/v1/bills/negotiation-status/:id`

    - Response: `{ status, savings }`

Alerts & Notifications

- `GET /api/v1/alerts`
    - Response: `{ alerts: [ ... ] }`
- `POST /api/v1/alerts/preferences`
    - Request: `{ type, channel, frequency }`
    - Response: `{ success }`
- `POST /api/v1/alerts/mark-read`
    - Request: `{ alertId }`
    - Response: `{ success }`

AI Recommendations & Summaries

- `GET /api/v1/ai/recommendations`
    - Response: `{ recommendations: [ ... ] }`
- `POST /api/v1/ai/feedback`
    - Request: `{ recommendationId, rating, comment }`
    - Response: `{ success }`
- `GET /api/v1/ai/summary`
    - Response: `{ summary: "Your finances are on track..." }`

Investment Tracking

- `GET /api/v1/investments`
    - Response: `{ investments: [ ... ] }`
- `GET /api/v1/investments/performance`
    - Response: `{ performance: [ ... ] }`

Credit Score Monitoring

- `GET /api/v1/credit-score`
    - Response: `{ score, factors }`

Reporting

- `GET /api/v1/reports`
    - Response: `{ reports: [ ... ] }`
- `POST /api/v1/reports/generate`

  - Request: `{ period }`
  - Response: `{ reportId, pdfUrl }`

User Feedback

- `POST /api/v1/feedback`
  - Request: `{ type, message }`
  - Response: `{ success }`

Example Request/Response Schema

```
// POST /api/v1/ai/feedback
{
  "recommendationId": "rec_123456",
  "rating": 4,
  "comment": "Helpful but missed one subscription."
}
```

```
// Response
{
  "success": true
}
```

---

16. Testing Strategy

- **Unit Testing:** Jest (backend), React Testing Library (frontend)
- **Integration Testing:** Supertest (API), mock third-party APIs
- **E2E Testing:** Cypress, Selenium (critical flows: onboarding, account linking, subscription management)
- **Performance Testing:** JMeter, k6 (simulate 10,000 concurrent users)
- **Security Testing:** OWASP ZAP, regular penetration tests, dependency scanning
- **Accessibility Testing:** axe-core, manual WCAG 2.1 AA checks
- **Regression Testing:** Automated CI/CD pipeline triggers on each release

---

17. Deployment Strategy

- **Phased Rollout:**

- Internal alpha → closed beta → public beta → GA

- **Blue-Green Deployments:**
  - Zero-downtime, quick rollback

- **Rollback Procedures:**
  - Automated rollback on failed health checks

- **CI/CD:**
  - GitHub Actions, AWS CodeDeploy

- **Canary Releases:**
  - Gradual feature exposure, monitor error rates

---

18. Monitoring & Observability

- **Metrics:**
  - API latency, error rates, user activity, third-party API health, notification delivery

- **Dashboards:**
  - Grafana, AWS CloudWatch dashboards for real-time monitoring

- **Alerts:**
  - PagerDuty integration for critical errors, anomaly detection

- **Logging:**
  - Centralized (AWS CloudWatch), structured logs, audit trails

- **User Behavior Analytics:**
  - Mixpanel/Amplitude for onboarding, feature usage, funnel analysis

---

19. Timeline & Phases

Phase	Deliverables	Timeline	Dependencies
Discovery & Design	Wireframes, user flows, technical architecture	1 month	Stakeholder input

Phase	Deliverables	Timeline	Dependencies
MVP Build	Core features: onboarding, aggregation, alerts	2 months	Plaid, Stripe integration
Beta Release	Subscription mgmt, bill negotiation, AI summaries	1 month	OpenAI, BillShark API
Public Launch	Investment, credit, reporting, feedback loop	1 month	All integrations
Continuous Delivery	Feature enhancements, AI retraining, compliance	Ongoing	User feedback, compliance

20. Resource Requirements

- **Team Composition:**
  - 1 Product Manager
  - 2 UX/UI Designers (web/mobile, accessibility)
  - 4 Backend Engineers (Node.js, integrations)
  - 3 Frontend Engineers (React, PWA)
  - 2 QA Engineers (automation, security)
  - 1 DevOps Engineer (AWS, CI/CD, monitoring)
  - 1 Compliance Officer (PCI DSS, GDPR)
  - 1 Customer Support Lead
- **Skills Needed:**
  - Fintech integrations, AI/ML (OpenAI), security, UX for finance, accessibility, AWS infrastructure
- **Budget Considerations:**
  - Third-party API fees (Plaid, Stripe, OpenAI, BillShark)
  - AWS hosting and scaling

- Compliance audits and penetration testing
- Marketing and user acquisition

---

21. Change Management Plan

- **User Adoption:**
  - In-app guided tours, demo mode, onboarding tutorials
- **Training:**
  - Knowledge base, video walkthroughs, FAQ
- **Communication:**
  - Regular release notes, in-app announcements, email newsletters
- **Feedback Loop:**
  - In-app feedback forms, NPS surveys, user interviews
- **Support:**
  - Live chat, ticketing system, escalation procedures

References:

- All compliance and technical requirements are aligned with [PCI-DSS Compliance][1] and [Blockchain Technology in Financial Services][2] knowledge base entries.
- User personas, flows, and features are directly extracted and synthesized from the provided user answers and project context.

End of PRD – AI FinanceMaster App