Bill Management System - Product Requirements Document (PRD)

1. Executive Summary

Project Name: Bill Management System
Version: 1.0
Date: August 30, 2025
Author: Development Team

What is this project?

A digital system to help users track, manage, and pay their bills efficiently. This replaces manual bill tracking with an automated, user-friendly application.

Why are we building this?

- People often forget to pay bills on time, leading to late fees
- Manual tracking is error-prone and time-consuming
- Users need a centralized place to manage all their financial obligations

---

2. Problem Statement

Current Pain Points:

- Users forget bill due dates and incur late payment fees
- No central location to view all upcoming bills
- Manual tracking through spreadsheets or notes is inefficient
- Difficulty in budgeting for recurring expenses
- No alerts or reminders for upcoming payments

Target Users:

- Working professionals aged 25-45
- Homeowners and renters with multiple recurring bills
- Small business owners managing business expenses
- Anyone looking to improve their financial organization

---

3. Goals & Success Metrics

Primary Goals

1. **Reduce Late Payments**: Help users pay bills on time
2. **Centralize Bill Management**: Single platform for all bills
3. **Improve Financial Planning**: Better visibility into expenses

Success Metrics

- 90% reduction in user-reported late payments
- 80% user retention rate after 3 months
- Average user manages 8+ bills in the system
- 4.5+ star rating in app stores

---

4. User Stories & Requirements

4.1 Core User Stories

As a user, I want to:

1. **Add Bills**
   - Add new bills with details (amount, due date, category)
   - Set up recurring bills (monthly, quarterly, yearly)
   - Upload bill documents/photos
2. **View Bills**
   - See all upcoming bills in a dashboard
   - View bill history and payment records
   - Filter bills by category, date, or status
3. **Get Reminders**
   - Receive notifications before bill due dates
   - Set custom reminder preferences (1 day, 3 days, 1 week before)
   - Get alerts for overdue bills
4. **Track Payments**
   - Mark bills as paid
   - Record payment method and confirmation numbers

- View payment history and patterns

5. **Budget Planning**
    - See monthly/yearly bill totals
    - View spending by category
    - Get insights on bill trends

4.2 Technical Requirements

Must Have (Priority 1):

- User authentication (login/signup)
- Add/edit/delete bills
- Dashboard with upcoming bills
- Basic notifications
- Mobile-responsive design

Should Have (Priority 2):

- Email/SMS reminders
- Bill categories and filtering
- Payment tracking
- Data export functionality
- Search functionality

Could Have (Priority 3):

- Integration with banking apps
- Bill splitting for roommates
- Advanced analytics
- Dark mode
- Multi-language support

---

5. User Experience (UX) Requirements

5.1 Key User Flows

New User Onboarding:

1. User signs up → 2. Completes profile → 3. Adds first bill → 4. Sets up notifications → 5. Views dashboard

Adding a Bill:

1. Click "Add Bill" → 2. Enter bill details → 3. Set recurrence → 4. Save → 5. Confirmation

Daily Usage:

1. Open app → 2. View dashboard → 3. See upcoming bills → 4. Mark as paid if needed → 5. Check next due dates

5.2 Design Principles

- **Simple**: Clean, uncluttered interface
- **Fast**: Quick loading times and easy navigation
- **Reliable**: Accurate data and consistent notifications
- **Accessible**: Works for users with different abilities

---

6. Technical Specifications

6.1 System Architecture

- **Frontend**: Web application (React/Vue.js) + Mobile app (React Native/Flutter)
- **Backend**: REST API (Node.js/Python/Java)
- **Database**: PostgreSQL or MongoDB
- **Hosting**: Cloud platform (AWS/Google Cloud/Azure)

6.2 Key Features Implementation

Database Schema (Simplified):

```
Users Table:
- user_id, email, password, name, created_at

Bills Table:
- bill_id, user_id, bill_name, amount, due_date, category, is_recurring, status

Payments Table:
- payment_id, bill_id, payment_date, amount, payment_method

Notifications Table:
- notification_id, user_id, bill_id, notification_date, type, sent
```

6.3 Security Requirements

- Encrypted password storage
- HTTPS for all communications
- User data protection (GDPR compliance)
- Secure API endpoints with authentication

---

7. Development Phases

Phase 1: MVP (Minimum Viable Product) - 8 weeks

Features:

- User registration/login
- Add/edit/delete bills
- Basic dashboard
- Email notifications
- Mark bills as paid

Success Criteria:

- Users can successfully manage their bills
- Basic notification system works
- Secure user authentication

Phase 2: Enhanced Features - 6 weeks

Features:

- Mobile app
- Advanced filtering and search
- Bill categories
- Payment history
- SMS notifications

Phase 3: Advanced Features - 8 weeks

Features:

- Analytics and insights
- Bill sharing
- Banking integration
- Advanced reporting

---

8. Business Requirements

8.1 Monetization Strategy

- **Freemium Model**: Basic features free, premium features paid
- **Premium Features**: Advanced analytics, unlimited bills, priority support
- **Subscription**: $4.99/month or $49.99/year

8.2 Compliance & Legal

- Data privacy compliance (GDPR, CCPA)
- Terms of service and privacy policy
- Financial data handling regulations

---

9. Risk Assessment

High-Risk Items

- **Data Security**: User financial information must be protected

- **Notification Reliability**: Failed reminders defeat the app's purpose
- **User Adoption**: Getting users to switch from existing methods

Mitigation Strategies

- Implement robust security measures and regular audits
- Build redundant notification systems
- Focus on excellent user experience and onboarding

---

10. Testing Strategy

Testing Types

1. **Unit Testing**: Individual functions and components
2. **Integration Testing**: API endpoints and database connections
3. **User Acceptance Testing**: Real users testing key workflows
4. **Security Testing**: Vulnerability assessments
5. **Performance Testing**: Load testing for scalability

Key Test Scenarios

- User can add and manage bills successfully
- Notifications are sent at correct times
- Payment tracking works accurately
- Data synchronizes properly across devices

---

11. Launch Plan

Pre-Launch (2 weeks before)

- Beta testing with 50 selected users
- Final bug fixes and optimizations
- App store submission (if mobile)
- Marketing materials preparation

Launch Week

- Soft launch to limited audience

- Monitor system performance
- Collect user feedback
- Address critical issues

Post-Launch (First month)

- User feedback analysis
- Performance monitoring
- Feature usage analytics
- Plan next iteration

---

12. Future Enhancements

Potential Features for Version 2.0

- AI-powered bill prediction
- Voice commands for adding bills
- Integration with smart home devices
- Collaborative family bill management
- Investment and savings tracking

---

13. Glossary

PRD: Product Requirements Document - this document
MVP: Minimum Viable Product - basic version with core features
API: Application Programming Interface - how different software components communicate
UI/UX: User Interface/User Experience - how the app looks and feels
Backend: Server-side components that handle data and logic
Frontend: User-facing parts of the application