Functional Requirements Document (FRD)

Document Information

Project: FlowTrack - Inventory Management System
Version: 2.0 (Enhancement Phase)
Date: February 6, 2026
Status: In Development
Audience: Developers, Product Managers, Stakeholders

Executive Summary

FlowTrack is an asset lending management system currently in **Phase 1 (Academic/MVP)** with core functionality complete. This document outlines requirements for **Phase 2 (Production Features)** including payment processing, notifications, enhanced business logic, UI improvements, and DevOps infrastructure.

User Personas

Persona 1: Alex - The Borrower

- **Role:** Student/Employee
- **Age:** 20-35
- **Tech Savviness:** Medium
- **Goals:**
  - Borrow equipment quickly
  - Know when items are due
  - Pay fines online
  - Extend loans if needed
- **Pain Points:**
  - Forgets return dates
  - Doesn't want to visit office to pay fines
  - Unclear what's available right now
  - Can't book items in advance

Persona 2: Sarah - The Staff Member

- **Role:** Front desk staff
- **Age:** 25-40
- **Tech Savviness:** Medium-High
- **Goals:**
  - Process loans/returns quickly
  - Minimize manual data entry
  - Track who has what
  - Generate reports for management
- **Pain Points:**

- Too many manual emails to send
- Difficult to find borrower history
- Can't track payment status
- Repetitive tasks take too much time

Persona 3: Michael - The Manager 👔

- **Role:** Inventory Manager
- **Age:** 30-50
- **Tech Savviness:** Medium-High
- **Goals:**
  - See real-time inventory status
  - Identify problem borrowers
  - Optimize inventory usage
  - Track costs (purchases, repairs, losses)
  - Make data-driven decisions
- **Pain Points:**
  - Can't see trends over time
  - No alerts for critical situations
  - Manual report generation
  - No way to forecast inventory needs

Persona 4: Priya - The Admin 👩‍💻

- **Role:** System Administrator
- **Age:** 25-45
- **Tech Savviness:** High
- **Goals:**
  - Manage users and permissions
  - Ensure system security
  - Monitor system health
  - Troubleshoot issues
  - Configure system settings
- **Pain Points:**
  - No audit trail of changes
  - Can't see system metrics
  - Manual user management
  - No API for integrations

🔢 Functional Requirements

Module 1: Payment Gateway Integration

FR-PAY-001: Payment Method Configuration

Priority: High
User Story: As an admin, I want to configure payment gateway settings so the organization can accept online

payments.

Requirements:

- Support multiple payment providers (Stripe, PayPal, Razorpay)
- Secure API key storage (encrypted, not in source code)
- Configure payment currency
- Set up webhook endpoints
- Test mode for development/staging

Acceptance Criteria:

- ☐ Admin can select payment provider from dropdown
- ☐ API keys stored encrypted in database
- ☐ Test transactions work in sandbox mode
- ☐ Webhook verification implemented
- ☐ Error logging for failed configurations

FR-PAY-002: Fine Payment

Priority: High
User Story: As a borrower, I want to pay my fines online so I don't have to visit the office.

Requirements:

- Display all outstanding fines
- Support full payment
- Support partial payment
- Multiple payment methods (Credit Card, Debit Card, UPI, Wallet)
- Payment receipt generation
- Payment confirmation email
- Update fine status in real-time

Acceptance Criteria:

- ☐ Borrower can view all unpaid fines
- ☐ Payment form validates card details
- ☐ Payment processes through gateway
- ☐ Receipt sent to borrower email
- ☐ Fine marked as "Paid" in database
- ☐ Transaction ID stored in database
- ☐ Failed payments show error message
- ☐ Payment history visible to borrower

Database Changes:

```
CREATE TABLE payments (
    payment_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    fine_id BIGINT,
```

```
    amount DECIMAL(10,2),
    payment_method VARCHAR(50),
    transaction_id VARCHAR(255),
    payment_status ENUM('PENDING', 'SUCCESS', 'FAILED', 'REFUNDED'),
    payment_date TIMESTAMP,
    gateway_response TEXT,
    FOREIGN KEY (fine_id) REFERENCES fines(fine_id)
);
```

FR-PAY-003: Payment History & Receipts

Priority: Medium
User Story: As a borrower, I want to view my payment history and download receipts.

Requirements:

- List all payment transactions
- Filter by date range, status
- Download receipt PDF
- Email receipt
- View payment details

Acceptance Criteria:

- ☐ Payment history page shows all transactions
- ☐ Filters work correctly
- ☐ PDF receipt includes all transaction details
- ☐ Receipt can be emailed
- ☐ Receipt includes organization branding

FR-PAY-004: Refund Management

Priority: Low
User Story: As a manager, I want to process refunds for incorrect charges.

Requirements:

- Initiate refund for payment
- Full or partial refund
- Refund reason tracking
- Automatic fine adjustment
- Refund notification to borrower

Acceptance Criteria:

- ☐ Manager can select payment to refund
- ☐ Refund processes through gateway
- ☐ Fine amount adjusted automatically

- [ ] Borrower receives refund notification
- [ ] Refund recorded in payment history

Module 2: Notification System

FR-NOT-001: Email Notification Infrastructure

Priority: High
User Story: As an admin, I want to configure email settings so the system can send notifications.

Requirements:

- Support SMTP configuration
- Support SendGrid/AWS SES/Mailgun
- Email templates management
- Test email functionality
- Email queue for reliability
- Retry logic for failed emails

Acceptance Criteria:

- [ ] Admin can configure SMTP settings
- [ ] Email templates editable in UI
- [ ] Template variables work ({{borrower_name}}, {{item_name}})
- [ ] Test email sends successfully
- [ ] Failed emails logged with reason
- [ ] Email queue processing in background

Database Changes:

```
CREATE TABLE notification_preferences (
    pref_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT,
    email_enabled BOOLEAN DEFAULT TRUE,
    sms_enabled BOOLEAN DEFAULT FALSE,
    loan_reminders BOOLEAN DEFAULT TRUE,
    payment_alerts BOOLEAN DEFAULT TRUE,
    system_updates BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE notification_log (
    log_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT,
    notification_type VARCHAR(50),
    channel ENUM('EMAIL', 'SMS', 'IN_APP'),
    status ENUM('SENT', 'FAILED', 'PENDING'),
    sent_at TIMESTAMP,
    error_message TEXT,
```

```
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

FR-NOT-002: Loan Due Reminders

Priority: High
User Story: As a borrower, I want to receive reminders before my item is due so I don't forget.

Requirements:

- Send reminder 1 day before due date
- Send reminder on due date
- Send overdue notification (daily for first 3 days)
- Configurable reminder schedule
- Batch processing (send all reminders at once)

Acceptance Criteria:

- ☐ Scheduled job runs daily at 9 AM
- ☐ Identifies loans due tomorrow/today/overdue
- ☐ Sends personalized email to each borrower
- ☐ Email includes item details, due date, late fee info
- ☐ Notification logged in database
- ☐ Admin can view notification history

Email Template Example:

```
Subject: Reminder: {{item_name}} due {{due_date}}

Hi {{borrower_name}},

This is a friendly reminder that {{item_name}} is due back {{due_date}}.

Item: {{item_name}}
Borrowed on: {{issue_date}}
Due date: {{due_date}}
Fine rate if late: ₹{{fine_rate}} per day

Please return the item on time to avoid late fees.

[View My Loans]

Thanks,
FlowTrack Team
```

FR-NOT-003: Fine Notifications

Priority: High
User Story: As a borrower, I want to be notified when a fine is charged so I can pay it promptly.

Requirements:

- Send notification when fine is generated
- Include fine amount, calculation breakdown
- Link to payment page
- Weekly reminder for unpaid fines
- Escalation to admin after 30 days

Acceptance Criteria:

- ☐ Notification sent immediately after return with fine
- ☐ Email shows fine calculation (days late × rate)
- ☐ "Pay Now" button links to payment page
- ☐ Weekly reminders for unpaid fines
- ☐ Admin notification for fines >30 days old

FR-NOT-004: In-App Notifications

Priority: Medium
User Story: As a user, I want to see notifications in the app without checking email.

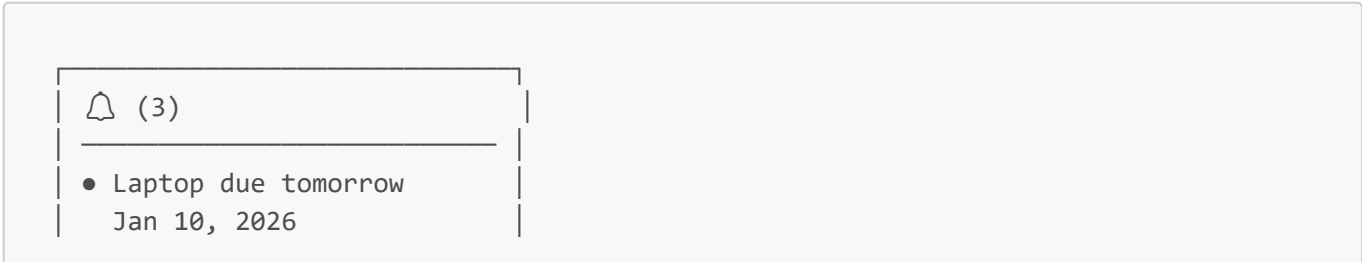
Requirements:

- Notification bell icon in navbar
- Badge showing unread count
- Dropdown list of recent notifications
- Mark as read functionality
- Link to relevant page (loan details, payment page)
- Notification preferences

Acceptance Criteria:

- ☐ Notification icon shows unread count
- ☐ Clicking icon shows dropdown
- ☐ Notifications sorted by date (newest first)
- ☐ Clicking notification marks as read
- ☐ Link navigates to correct page
- ☐ User can enable/disable notification types

UI Mockup:



```
│ ──────────────────────────── │
│ ○ Fine charged: ₹50          │
│   Jan 9, 2026                │
│ ──────────────────────────── │
│ ○ Mouse returned             │
│   Jan 8, 2026                │
│ ──────────────────────────── │
│ [View All Notifications]     │
└──────────────────────────────┘
```

FR-NOT-005: Admin Alert Notifications

Priority: Medium
User Story: As a manager, I want to receive alerts for critical situations.

Requirements:

- Low stock alert (quantity < threshold)
- Item not returned after 7 days overdue
- High-value item issued
- Item marked for repair
- Daily summary of pending actions

Acceptance Criteria:

- ☐ Admin receives email for each alert type
- ☐ Alert thresholds configurable
- ☐ Daily summary at 8 AM with all pending items
- ☐ Alert includes action buttons
- ☐ Alert can be dismissed/acknowledged

Module 3: Enhanced Business Logic

FR-BUS-001: Item Reservation System

Priority: High
User Story: As a borrower, I want to reserve an item in advance so I'm guaranteed availability.

Requirements:

- Book item for future date range
- System checks availability before confirming
- Reservation expires if not picked up
- Pickup reminder notification
- Manager can cancel reservations

Acceptance Criteria:

- ☐ Borrower can browse available items

- [ ] Calendar shows availability
- [ ] Reservation created only if available
- [ ] Borrower receives confirmation email
- [ ] Reminder sent 1 day before pickup
- [ ] Reservation auto-cancels if not picked up within X hours

Database Changes:

```
CREATE TABLE reservations (
    reservation_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    item_id BIGINT,
    borrower_id BIGINT,
    reservation_date TIMESTAMP,
    start_date DATE,
    end_date DATE,
    status ENUM('PENDING', 'CONFIRMED', 'FULFILLED', 'CANCELLED', 'EXPIRED'),
    pickup_deadline TIMESTAMP,
    created_at TIMESTAMP,
    FOREIGN KEY (item_id) REFERENCES Inventory_item(item_id),
    FOREIGN KEY (borrower_id) REFERENCES Borrower(borrower_id)
);
```

FR-BUS-002: Waiting List

Priority: Medium
User Story: As a borrower, I want to join a waiting list for unavailable items so I'm notified when available.

Requirements:

- Join waiting list for out-of-stock items
- FIFO queue management
- Automatic notification when available
- Option to decline (goes to next person)
- Time limit to accept (24 hours)

Acceptance Criteria:

- [ ] "Join Waiting List" button shown for unavailable items
- [ ] User added to queue
- [ ] When item returned, first person in queue notified
- [ ] Notification includes "Reserve Now" link
- [ ] If not reserved within 24h, next person notified

FR-BUS-003: Item Kits/Bundles

Priority: Medium
User Story: As a manager, I want to create item bundles so related items are issued together.

Requirements:

- Create kit with multiple items
- Issue entire kit with one transaction
- Track kit availability (all items must be available)
- Return entire kit together
- Kit fine calculated as sum of individual fines

Acceptance Criteria:

- ☐ Manager can create kit (e.g., "Laptop Kit" = Laptop + Charger + Bag)
- ☐ Kit shows as unavailable if any component unavailable
- ☐ Issuing kit creates loan for each item
- ☐ Returning kit processes all items
- ☐ Report shows kit as single unit

Example Kits:

- **Photography Kit:** Camera + Lens + Tripod + Memory Card
- **Laptop Kit:** Laptop + Charger + Mouse + Bag
- **Lab Kit:** Microscope + Slides + Gloves

FR-BUS-004: Loan Extensions

Priority: Medium
User Story: As a borrower, I want to extend my loan period if I need the item longer.

Requirements:

- Request extension before due date
- Maximum 2 extensions per loan
- Extension only if no reservations exist
- Manager approval required (optional config)
- New due date calculated

Acceptance Criteria:

- ☐ "Request Extension" button on loan details page
- ☐ Extension request only allowed before due date
- ☐ System checks for reservations
- ☐ If auto-approve enabled, extends immediately
- ☐ If approval required, manager receives notification
- ☐ Borrower notified of approval/rejection

FR-BUS-005: Bulk Operations

Priority: Low
User Story: As staff, I want to issue multiple items at once to save time.

Requirements:

- Select multiple items
- Issue all to same borrower
- Single due date for all
- Bulk return processing
- Bulk fine calculation

Acceptance Criteria:

- ☐ Checkbox selection on item list
- ☐ "Issue Selected Items" button
- ☐ Single form with borrower and due date
- ☐ Creates separate loan for each item
- ☐ Return page allows selecting multiple loans
- ☐ Return processes all selected loans

FR-BUS-006: Dynamic Fine Rules

Priority: Low
User Story: As a manager, I want to configure fine rules per category so expensive items have higher penalties.

Requirements:

- Per-category fine rates
- Grace period before fine starts
- Maximum fine cap
- First-time offender discount
- Weekend/holiday exclusions

Acceptance Criteria:

- ☐ Fine rate configurable per item type
- ☐ Grace period setting (default 1 day)
- ☐ Maximum fine setting (e.g., 3x item price)
- ☐ 50% discount for first late return
- ☐ Weekends not counted in days late

Fine Calculation Logic:

```
Days Late = Return Date - Due Date - Grace Period - Weekends/Holidays

If First Offense:
    Fine = Days Late × Fine Rate × 0.5
Else:
    Fine = Days Late × Fine Rate
```

```
If Fine > Max Fine (3 × Item Price):
    Fine = 3 × Item Price
```

Module 4: UI/UX Improvements

FR-UI-001: Responsive Mobile Design

Priority: High
User Story: As a borrower, I want to use the system on my phone for convenient access.

Requirements:

- Mobile-first responsive design
- Touch-friendly buttons (min 44px)
- Simplified navigation for mobile
- Fast loading on 3G/4G
- Progressive Web App (PWA) capabilities

Acceptance Criteria:

- ☐ All pages responsive (320px to 1920px)
- ☐ Navigation collapses to hamburger menu on mobile
- ☐ Forms usable on mobile (proper input types)
- ☐ Images optimized for mobile
- ☐ PWA installable on mobile home screen

FR-UI-002: Advanced Search & Filters

Priority: High
User Story: As a borrower, I want to search and filter items easily to find what I need.

Requirements:

- Full-text search across items
- Fuzzy search (handle typos)
- Filters: Category, Availability, Price Range
- Sort: Name, Price, Popularity
- Search suggestions as you type
- Save search preferences

Acceptance Criteria:

- ☐ Search bar prominent on item list page
- ☐ Search works on name, description, category
- ☐ Typos handled (e.g., "latop" finds "laptop")
- ☐ Filters update results without page reload
- ☐ Sort dropdown updates instantly
- ☐ Autocomplete shows suggestions

UI Mockup:

```
┌─────────────────────────────────────────┐
│ Search: [laptop_____________] 🔍        │
│                                         │
│ Filters:                                │
│ Category: [Electronics ▼]               │
│ Availability: [Available ▼]             │
│ Price: ₹[0] - ₹[5000]                   │
│ Sort by: [Popularity ▼]                 │
│                                         │
│ Results (12):                           │
│ ┌──────────────────┐                    │
│ │ [Laptop Image]   │ Laptop             │
│ │                  │ ₹800 | Available: 5 │
│ └──────────────────┘ [View Details]     │
└─────────────────────────────────────────┘
```

FR-UI-003: Interactive Dashboard

Priority: Medium
User Story: As a manager, I want an interactive dashboard with real-time data visualization.

Requirements:

- Real-time charts (auto-refresh every 30s)
- Drill-down capabilities
- Date range selection
- Export charts as images
- Customizable widgets

Charts to Include:

- Inventory utilization (pie chart)
- Loans over time (line chart)
- Top borrowed items (bar chart)
- Fine collection (area chart)
- Items by category (donut chart)

Acceptance Criteria:

- ☐ Dashboard loads within 2 seconds
- ☐ Charts interactive (hover for details)
- ☐ Date range filter affects all charts
- ☐ Charts exportable as PNG/SVG
- ☐ Dashboard responsive on mobile

FR-UI-004: Dark Mode

Priority: Low
User Story: As a user, I want a dark mode option for comfortable viewing at night.

Requirements:

- Toggle between light/dark mode
- Preference saved per user
- System preference detection
- Smooth transition animation
- WCAG contrast compliance

Acceptance Criteria:

- ☐ Toggle in user profile
- ☐ Preference saved in database
- ☐ Auto-detect system preference on first visit
- ☐ All pages support dark mode
- ☐ Contrast ratio meets WCAG AA standards

FR-UI-005: Accessibility (WCAG 2.1 AA)

Priority: Medium
User Story: As a user with disabilities, I want the system to be accessible.

Requirements:

- Keyboard navigation for all features
- Screen reader compatibility
- Alt text for all images
- Focus indicators visible
- Sufficient color contrast
- Text resizable to 200%
- Error messages clear and specific

Acceptance Criteria:

- ☐ All forms navigable by keyboard
- ☐ Screen reader announces all actions
- ☐ Images have descriptive alt text
- ☐ Focus outline visible (3px blue border)
- ☐ Contrast ratio ≥ 4.5:1
- ☐ Text resizes without breaking layout
- ☐ Passed WAVE accessibility checker

Module 5: DevOps & Infrastructure

FR-DEV-001: Containerization

Priority: High
User Story: As a DevOps engineer, I want to containerize the application for consistent deployments.

Requirements:

- Multi-stage Dockerfile
- Docker Compose for local development
- Health check endpoints
- Environment-specific configurations
- Volume management for uploads

Acceptance Criteria:

- ☐ Dockerfile builds successfully
- ☐ Image size < 500MB
- ☐ Docker Compose starts all services
- ☐ Health endpoint returns 200 OK
- ☐ Environment variables configurable
- ☐ Database data persists across restarts

Dockerfile Example:

```
FROM openjdk:11-jdk-slim AS build
WORKDIR /app
COPY pom.xml .
COPY src ./src
RUN mvn clean package -DskipTests

FROM openjdk:11-jre-slim
WORKDIR /app
COPY --from=build /app/target/*.jar app.jar
EXPOSE 8080
HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost:8080/actuator/health || exit 1
ENTRYPOINT ["java", "-jar", "app.jar"]
```

FR-DEV-002: CI/CD Pipeline

Priority: High
User Story: As a developer, I want automated builds and deployments to save time.

Requirements:

- GitHub Actions pipeline
- Automated testing on PR
- Build Docker image on merge
- Deploy to staging automatically
- Manual approval for production

- Rollback capability

Acceptance Criteria:

- ☐ Pipeline runs on every push
- ☐ Tests must pass before merge
- ☐ Image pushed to container registry
- ☐ Staging deployed automatically
- ☐ Production requires approval
- ☐ Rollback script available

GitHub Actions Workflow:

```yaml
name: CI/CD Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 11
        uses: actions/setup-java@v3
        with:
          java-version: '11'
      - name: Run tests
        run: mvn test

  build:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - name: Build Docker image
        run: docker build -t flowtrack:latest .
      - name: Push to registry
        run: docker push flowtrack:latest

  deploy-staging:
    needs: build
    runs-on: ubuntu-latest
    environment: staging
    steps:
      - name: Deploy to staging
        run: ./deploy.sh staging
```

FR-DEV-003: Monitoring & Logging

Priority: High
User Story: As an admin, I want to monitor system health and debug issues quickly.

Requirements:

- Application metrics (Prometheus)
- Visualization dashboard (Grafana)
- Centralized logging (ELK or Loki)
- Error tracking (Sentry)
- Uptime monitoring
- Alert configuration

Metrics to Track:

- Request rate, latency, error rate
- Database query performance
- Memory and CPU usage
- Active users
- Loan transactions per hour
- Payment success rate

Acceptance Criteria:

- ☐ Metrics endpoint exposed (/actuator/prometheus)
- ☐ Grafana dashboard shows all metrics
- ☐ Logs aggregated in centralized system
- ☐ Errors sent to Sentry
- ☐ Alerts configured for critical issues
- ☐ Uptime > 99.5% tracked

FR-DEV-004: Automated Backups

Priority: High
User Story: As an admin, I want automatic database backups to prevent data loss.

Requirements:

- Daily automated backups
- Incremental backups every 6 hours
- Backup retention (30 days)
- Encrypted backups
- Backup restoration testing
- Off-site storage

Acceptance Criteria:

- ☐ Cron job runs daily at 2 AM
- ☐ Backup file encrypted with AES-256

- ☐ Backups stored in AWS S3/Azure Blob
- ☐ Restoration tested monthly
- ☐ Old backups auto-deleted after 30 days
- ☐ Backup success/failure notifications

FR-DEV-005: Load Balancing & Auto-Scaling

Priority: Medium
User Story: As a DevOps engineer, I want the system to handle traffic spikes automatically.

Requirements:

- Load balancer configuration
- Auto-scaling based on CPU/memory
- Minimum 2 instances
- Maximum 10 instances
- Session persistence
- Health check based routing

Acceptance Criteria:

- ☐ Load balancer distributes traffic evenly
- ☐ Scales up when CPU > 70%
- ☐ Scales down when CPU < 30%
- ☐ Sessions maintained on scale events
- ☐ Unhealthy instances removed automatically
- ☐ System handles 1000 concurrent users

📊 Non-Functional Requirements

Performance

- Page load time: < 2 seconds
- API response time: < 500ms (95th percentile)
- Database query time: < 100ms
- Support 500 concurrent users
- 99.5% uptime

Security

- HTTPS only (TLS 1.2+)
- OWASP Top 10 vulnerabilities addressed
- SQL injection prevention
- XSS protection
- CSRF tokens
- Rate limiting (100 requests/minute per user)
- Password complexity requirements
- Session timeout (30 minutes)

Scalability

- Horizontal scaling support
- Database connection pooling
- Caching (Redis)
- CDN for static assets
- Async processing for heavy tasks

Reliability

- Zero-downtime deployments
- Database replication
- Automatic failover
- Graceful degradation
- Circuit breakers for external services

Usability

- Intuitive navigation
- Consistent UI patterns
- Error messages clear and actionable
- Help documentation
- Onboarding tutorials

📅 Implementation Priority

Phase 1 (Immediate - 1-2 months):

1. ☑ Payment Gateway (FR-PAY-001 to FR-PAY-003)
2. ☑ Basic Notifications (FR-NOT-001 to FR-NOT-003)
3. ☑ Responsive Mobile Design (FR-UI-001)
4. ☑ Containerization (FR-DEV-001)
5. ☑ CI/CD Pipeline (FR-DEV-002)

Phase 2 (Short-term - 2-4 months):

1. ☑ Item Reservations (FR-BUS-001)
2. ☑ Advanced Search (FR-UI-002)
3. ☑ In-App Notifications (FR-NOT-004)
4. ☑ Monitoring & Logging (FR-DEV-003)
5. ☑ Automated Backups (FR-DEV-004)

Phase 3 (Medium-term - 4-6 months):

1. ☑ Waiting List (FR-BUS-002)
2. ☑ Item Kits (FR-BUS-003)
3. ☑ Loan Extensions (FR-BUS-004)
4. ☑ Interactive Dashboard (FR-UI-003)

5. ☑ Load Balancing (FR-DEV-005)

Phase 4 (Long-term - 6+ months):

1. ☑ Payment Refunds (FR-PAY-004)
2. ☑ Bulk Operations (FR-BUS-005)
3. ☑ Dynamic Fine Rules (FR-BUS-006)
4. ☑ Dark Mode (FR-UI-004)
5. ☑ Accessibility (FR-UI-005)
6. ☑ Admin Alerts (FR-NOT-005)

📋 Acceptance Testing Plan

Payment Gateway Testing:

- ☐ Successful payment flow
- ☐ Declined card handling
- ☐ Network timeout handling
- ☐ Concurrent payment processing
- ☐ Refund processing
- ☐ Receipt generation

Notification Testing:

- ☐ Email delivery success
- ☐ SMS delivery (if implemented)
- ☐ In-app notification display
- ☐ Notification preferences respected
- ☐ Unsubscribe functionality
- ☐ Load testing (1000 emails/minute)

Business Logic Testing:

- ☐ Reservation conflicts handled
- ☐ Item availability calculated correctly
- ☐ Fine calculation accuracy
- ☐ Loan extension logic
- ☐ Kit component tracking
- ☐ Edge cases (same-day return, midnight crossover)

UI/UX Testing:

- ☐ Cross-browser compatibility (Chrome, Firefox, Safari, Edge)
- ☐ Mobile responsiveness (iOS/Android)
- ☐ Touch interactions
- ☐ Keyboard navigation
- ☐ Screen reader testing
- ☐ Performance on slow connections

DevOps Testing:

- ☐ Container startup/shutdown
- ☐ Database migration scripts
- ☐ Backup restoration
- ☐ Load balancer failover
- ☐ Auto-scaling triggers
- ☐ Zero-downtime deployment

🔗 Dependencies & Integrations

External Services:

- **Payment:** Stripe/Razorpay API
- **Email:** SendGrid/AWS SES
- **SMS:** Twilio/AWS SNS (optional)
- **Storage:** AWS S3/Azure Blob (for receipts, reports)
- **Monitoring:** Prometheus + Grafana
- **Error Tracking:** Sentry
- **Analytics:** Google Analytics (optional)

Libraries & Frameworks:

- Spring Boot 2.7.18 → **Upgrade to 3.x** (consider)
- Thymeleaf → **Add REST APIs for future mobile app**
- Spring Security → **Add OAuth2/OIDC** (future)
- JPA/Hibernate → **Optimize queries**
- Chart.js → **Replace with D3.js** for advanced charts

📚 Documentation Requirements

Technical Documentation:

- ☐ API documentation (Swagger/OpenAPI)
- ☐ Database schema diagram
- ☐ Deployment guide
- ☐ Troubleshooting guide
- ☐ Architecture decision records (ADRs)

User Documentation:

- ☐ User manual (PDF)
- ☐ Video tutorials
- ☐ FAQ page
- ☐ Admin guide
- ☐ Quick start guide

☑ Definition of Done

A feature is considered "Done" when:

- ☐ Code written and peer-reviewed
- ☐ Unit tests written (>80% coverage)
- ☐ Integration tests written
- ☐ Documentation updated
- ☐ Tested on staging environment
- ☐ Security review passed
- ☐ Performance benchmarked
- ☐ Accessibility checked
- ☐ Product owner approval
- ☐ Deployed to production

This FRD is a living document and will be updated as requirements evolve.

Last Updated: February 6, 2026
Next Review: March 6, 2026