Product Requirements and Specification Document (PRD)

Project Name

SecurePay - Payment Security Simulator

Overview

SecurePay is an open-source simulation platform enabling finance professionals to test, learn, and improve payment system security. The platform provides hands-on, modular scenarios based on real-world attacks, secure authentication, and detailed analytics, built with Java, Spring, PostgreSQL, React, and OWASP best practices.

---

1. Objectives

Objective	Description
Hands-on Security Training	Enable users to simulate and respond to payment system attacks
Realistic Attack Scenarios	Integrate up-to-date, real-world payment security threats
Secure Authentication	Ensure robust, OWASP-compliant user authentication
Analytics & Reporting	Provide actionable insights on user performance and vulnerabilities
Modular & Extensible Design	Allow easy addition of new scenarios and features
Open-Source	Codebase and documentation are publicly available under an OSI-approved license

---

2. Core Features

Feature	Description
Scenario Simulator	Interactive modules simulating payment system attacks (e.g., MITM, phishing)
User Authentication	Secure login/registration (JWT, OAuth2, 2FA)
Analytics Dashboard	Visualize user progress, scenario outcomes, and security metrics
Scenario Authoring	Admin interface to create/edit attack scenarios
Role Management	User roles: Admin, Instructor, Learner
Audit Logging	Track user actions and scenario results
API-first Architecture	RESTful APIs for all core functionalities

---

3. User Stories

As a…	I want to…	So that…
Learner	Attempt attack scenarios	I can improve my payment security skills
Instructor	Track learner progress and assign scenarios	I can guide and assess learners
Admin	Manage users and scenarios	I can maintain platform integrity
Developer	Extend with new scenarios	The platform remains current and relevant

4. Functional Requirements

ID	Requirement
FR1	Users can register, login, and reset passwords securely (OWASP standards)
FR2	Users can select and launch payment security scenarios
FR3	System simulates attacks (e.g., SQLi, XSS, MITM, phishing) with realistic feedback
FR4	Analytics dashboard displays scenario results, vulnerabilities found, and improvement areas
FR5	Admins can create, edit, and delete scenarios via UI
FR6	Role-based access control for Admin, Instructor, Learner
FR7	All user actions and scenario outcomes are logged and auditable
FR8	RESTful API endpoints for all major functionalities
FR9	Platform supports modular addition of new scenarios

5. Non-Functional Requirements

ID	Requirement
NFR1	System must be OWASP Top 10 compliant
NFR2	All sensitive data encrypted at rest and in transit
NFR3	Platform must support 500 concurrent users
NFR4	Response time < 2 seconds for all user actions
NFR5	Codebase and documentation must be open-source (MIT/Apache 2.0)
NFR6	Modular architecture for easy extensibility
NFR7	Automated test coverage ≥ 80%

6. Technical Specifications

Component	Technology/Standard	Notes
Backend	Java 17+, Spring Boot	RESTful API, security modules
Frontend	React 18+	SPA, responsive design

Database	PostgreSQL 14+	Encrypted storage
AuthN/AuthZ	JWT, OAuth2, 2FA	OWASP best practices
Security	OWASP Dependency-Check	Regular vulnerability scanning
Containerization	Docker	For deployment and local dev
CI/CD	GitHub Actions	Automated build, test, deploy
Documentation	Markdown, OpenAPI (Swagger)	For code and API docs

7. High-Level Architecture

```
[User] <---> [React Frontend] <--REST API--> [Spring Boot Backend] <---> [PostgreSQ
                                         |                                 |
                                         |---> [Scenario Engine]           |
                                         |---> [Analytics Module]          |
```

8. Milestones

Milestone	Target Date
Requirements Finalized	Week 1
MVP Backend & Auth	Week 4
Scenario Engine v1	Week 6
Analytics Dashboard	Week 8
Admin/Authoring Tools	Week 10
Open-Source Release	Week 12

9. Open Issues & Risks

Issue/Risk	Mitigation
Evolving attack vectors	Modular scenario updates, community input
Data privacy compliance	Strict encryption, minimal PII storage
User onboarding complexity	In-app tutorials, clear documentation

10. Acceptance Criteria

- All core features implemented and tested
- OWASP Top 10 compliance verified
- ≥80% automated test coverage
- Documentation complete and open-sourced

- Successfully simulates at least 5 real-world attack scenarios
- Analytics dashboard functional and accurate

End of Document