Business Requirements Document (BRD)

Project Title: VAPT Reporting Portal
Version: 1.0
Date: 9/11/2025
Prepared by: Prajyot Karkade & Shubham Shingare

1. Introduction

1.1 Purpose

The purpose of this project is to design and implement a centralized, interactive, and secure Vulnerability Assessment and Penetration Testing (VAPT) Reporting Portal. This system will replace the current practice of sending reports over email by providing an all-in-one platform where Admins, Pentesters, and Clients can manage projects, vulnerabilities, reports, and communications efficiently.

1.2 Scope

The system will:

- Provide role-based access for Admins, Pentesters, and Clients.
- Enable project lifecycle management (creation → testing → reporting → delivery).
- Support structured vulnerability submission and reporting workflows.
- Provide clients with dashboards, analytics, and downloadable reports.
- Facilitate collaboration through comments, notifications, and retest requests.
- Improve security with strong authentication, role-based permissions, and secure file handling.

2. Business Objectives

1. **Streamline reporting workflow**: Replace email-based report delivery with a centralized system.
2. **Enhance collaboration**: Provide direct communication channels between pentesters and clients.
3. **Increase transparency**: Give clients access to real-time project and vulnerability status.

4. **Improve analytics**: Track vulnerabilities, severity distribution, and remediation progress across projects.
5. **Ensure security compliance**: Enforce secure authentication, encrypted storage, and audit logging.
6. **Scalability & Future-readiness**: Support API integrations, custom branding, and AI-driven insights.

3. Stakeholders

- **Primary Stakeholders:**
    - **Admin (Security Manager / Company Owner)**: Manage users, projects, analytics.
    - **Pentester (Internal Security Analyst / Consultant)**: Upload reports, submit vulnerabilities.
    - **Client (Customer Organization)**: Access dashboards, download reports, request retests.
- **Secondary Stakeholders:**
    - **Developers/IT Team**: Responsible for implementing and maintaining the system.
    - **Compliance Officers**: Ensure reports and workflows meet regulatory standards.

4. Functional Requirements

4.1 Admin Features

- Create, modify, delete Pentester and Client accounts.
- Create and manage projects (name, type, description, client).
- Assign one or more pentesters to projects.
- Track project status: *In Progress, Report Submitted, Delivered*.
- View analytics: total vulnerabilities, common types, severity distribution.
- Manage project templates and predefined scopes.

4.2 Pentester Features

- Secure login with username + password.
- Dashboard showing assigned projects and deadlines.
- Upload reports (PDF, DOCX, Markdown).
- Submit vulnerabilities with details (title, description, severity, CVSS, PoC, remediation).
- Track vulnerability/report status (*Draft → Submitted → Approved → Delivered*).

4.3 Client Features

- Secure login with username + password.
- Dashboard with:
    - Vulnerability pie chart by severity.
    - List of vulnerabilities with filters.
    - Trends of vulnerabilities over time.
- Report management: download latest and historical versions.
- Request retests for specific vulnerabilities.
- Update vulnerability status (*Open → In Progress → Fixed → Verified*).
- Provide comments/feedback on vulnerabilities.

4.4 Workflow Features

- Commenting system for vulnerabilities.
- Retest request and tracking.
- SLA tracking for high/critical issues.
- Email + in-app notifications for project updates, retests, and report deliveries.

4.5 Reporting & Analytics

- Interactive dashboards with charts:
    - Vulnerabilities by severity/type.
    - Vulnerability trends over time.

- Export reports/data in PDF, Excel, CSV.
- Compare new vs old reports for the same project.

4.6 Future-Ready Features (Optional)

- API access for clients.
- White-labeling for client branding.
- Knowledge base of remediation guides.
- AI/ML-driven severity suggestions and recurring vulnerability detection.

5. Non-Functional Requirements

- **Security:** Strong authentication (JWT + 2FA), encrypted storage, input validation, audit logging.
- **Performance:** Must handle concurrent access by multiple clients and pentesters without degradation.
- **Scalability:** Support multiple clients/projects with potential growth in users and reports.
- **Usability:** Role-based dashboards, clean UI with charts.
- **Reliability:** 99.9% uptime target, automatic backups, failover support.

6. Technical Requirements

- **Backend:** Django (Python) or Node.js (Express/NestJS).
- **Frontend:** React + TailwindCSS.
- **Database:** PostgreSQL or MongoDB.
- **Storage:** AWS S3 (preferred) or secure local storage.
- **Authentication:** JWT with refresh tokens + 2FA.
- **Charts/Graphs:** Chart.js / Recharts.

7. Assumptions & Dependencies

- Clients and pentesters will adopt the portal instead of email-based reporting.
- The system will be deployed in a secure cloud environment (AWS/Azure/GCP).
- Internet connectivity is required for all users.
- Dependencies: third-party libraries (Chart.js, AWS SDK).

8. Risks & Mitigations

- **Risk:** Users reluctant to adopt portal → *Mitigation:* Provide training & documentation.
- **Risk:** Data breach of sensitive vulnerability info → *Mitigation:* End-to-end encryption, access logs, security audits.
- **Risk:** Performance bottlenecks with large reports → *Mitigation:* Optimize storage and queries, implement caching.
- **Risk:** SLA violations for critical issues → *Mitigation:* Automated SLA tracking and alerts.

9. Success Criteria

- 100% of new reports delivered through the portal (no email).
- Clients can view vulnerabilities in real-time and request retests.
- Pentesters can upload and track reports without workflow issues.
- Admins have complete visibility of analytics across clients.

10. Deliverables

1. Fully functional **VAPT Reporting Portal** with Admin, Pentester, and Client roles.
2. Documentation:
    - System setup (local + production).
    - API usage guide.
    - User role workflows.
    - Example test data (sample users, projects, reports).

3. Unit and integration test coverage.