Initial Software Requirements Document (SRD) for Company Financial Report System

Version 1.0

1. Introduction

The **Company Financial Report System (CFRS)** is designed to streamline the creation, management, and presentation of monthly financial reports during financial meetings. It will provide stakeholders with real-time insights, interactive visualizations, and secure access to financial data.

2. Objectives

- Automate monthly financial report generation.
- Display financial data in an intuitive, visually engaging format.
- Ensure secure role-based access to sensitive financial information.
- Enable real-time updates during meetings.

3. Functional Requirements

3.1 User Roles and Authentication

- **Roles**:
  - **Executive**: Full access to all reports, raw data, and administrative controls.
  - **Finance Team**: Access to generate/edit reports and upload data.
  - **Guest (Read-Only)**: View finalized reports only.
- **Authentication**:
  - Login via email/password or SSO (e.g., Active Directory).
  - Session timeout after 15 minutes of inactivity.

3.2 Data Management

- **Data Sources**:
  - Import data from Excel, CSV, QuickBooks, or ERP systems (e.g., SAP).
  - Manual entry for adjustments.
- **Data Validation**:
  - Automated checks for missing entries, format errors, or outliers.
  - Alerts for discrepancies.

3.3 Reporting Features

- **Monthly Report Generation**:
  - Pre-built templates for Income Statements, Balance Sheets, Cash Flow, and KPIs.
  - Customizable charts (bar, line, pie) and tables.
- **Dashboard**:
  - Interactive filters (e.g., by department, time period, or project).
  - Drill-down capabilities for granular data.
- **Real-Time Updates**:
  - Reflect live data changes during meetings (e.g., revised forecasts).

3.4 Presentation Mode

- **Meeting Mode**:
  - Full-screen display optimized for projectors.
  - Presenter controls (annotations, spotlight metrics).
- **Export Options**:
  - PDF, Excel, or PowerPoint exports for offline sharing.

3.5 Audit and Compliance

- **Audit Logs**:
  - Track report modifications, data imports, and user activity.
- **Data Encryption**:

- Encrypt data at rest and in transit (TLS 1.3+).

---

4. Non-Functional Requirements

4.1 Performance

- Load reports within **≤2 seconds** for datasets up to 100k rows.
- Support **50+ concurrent users** during meetings.

4.2 Security

- Role-based access control (RBAC) compliant with GDPR/SOX.
- Regular vulnerability scans and penetration testing.

4.3 Usability

- Intuitive UI/UX with minimal training required.
- Mobile-responsive design for pre-meeting reviews.

4.4 Scalability

- Modular architecture to add future integrations (e.g., Power BI).

---

5. UML Overview (Without Diagrams)

5.1 Use Case Diagram (Conceptual)

- **Actors**: Executive, Finance Team, Guest.
- **Key Use Cases**:
  - Authenticate User.
  - Import/Validate Financial Data.
  - Generate Monthly Report.
  - Present Report in Meeting Mode.
  - Export Report.
  - Manage User Permissions.

5.2 Class Diagram (Conceptual)

- **Classes**:
  - **User**: Attributes: `userID`, `role`, `email`; Methods: `login()`, `logout()`.
  - **FinancialData**: Attributes: `month`, `revenue`, `expenses`, `profit`; Methods: `validate()`, `import()`.
  - **Report**: Attributes: `reportID`, `templateType`, `generatedDate`; Methods: `generate()`, `export()`.
  - **Dashboard**: Methods: `renderVisualization()`, `applyFilter()`.

5.3 Sequence Diagram (Conceptual)

- **Generate Report Flow**:
  1. User selects report parameters.
  2. System fetches data from `FinancialData`.
  3. System applies template and generates visualizations.
  4. User approves/rejects the report.

---

6. Initial Design Considerations

6.1 Architecture

- **Frontend**: Web-based (React.js/Angular) for cross-platform compatibility.
- **Backend**: Microservices (Python/Django or Node.js) for scalability.
- **Database**: Relational (PostgreSQL) for structured financial data.

6.2 Integration

- APIs to connect with ERP/accounting software.
- OAuth2 for SSO.

6.3 Deployment

- Cloud-hosted (AWS/Azure) with Docker containers.
- CI/CD pipeline for updates.

---

7. Risks and Mitigations

- **Data Inconsistency**: Implement validation rules and reconciliation workflows.
- **Performance Lag**: Use caching (Redis) for frequently accessed reports.

---

8. Next Steps

1. Finalize UML diagrams (use cases, class, sequence).
2. Develop a prototype for core reporting features.
3. Conduct stakeholder feedback sessions.

Approvals

☐ Product Owner

☐ Technical Lead

This document serves as the foundation for detailed technical specifications and sprint planning. Let me know if you need further refinements!