Software Requirements Specification (SRS)

Human Resource Management System (HRMS)

1 Introduction

1.1 Purpose

This HRMS is crafted to streamline and unify HR operations into a single, user-friendly, cloud-based web application built on Google Firebase. It automates repetitive tasks like managing employee records, tracking attendance, processing payroll, handling leave requests, overseeing recruitment, and evaluating performance. The goal is to reduce manual errors, enhance efficiency, and provide real-time insights for HR teams and employees alike, all while maintaining a seamless and intuitive experience.

1.2 Intended Audience

- **Development Team**: To guide the design and development of a robust system.
- **HR Managers & Staff**: To understand the systems capabilities and leverage its tools effectively.
- **Employees**: To explore self-service features for managing their data and requests.
- **Project Evaluators**: To ensure the system aligns with all specified functional and non-functional requirements.

1.3 Intended Use

- **HR/Admins**: Oversee employee profiles, attendance, leaves, payroll, recruitment, and performance metrics with ease.
- **Employees**: Access a self-service portal to submit leave requests, download payslips, and update personal information.
- **Managers**: Approve leaves, monitor team performance, and access real-time attendance reports.

1.4 Product Scope

This Firebase-powered HRMS delivers:

- Centralized management of employee data.
- Automated attendance and leave tracking with approval workflows.
- Payroll processing with payslip generation.
- Recruitment pipeline management.
- Performance tracking with Key Result Areas (KRAs) and feedback mechanisms.
- Interactive dashboards for actionable insights and reporting.

Designed for organizations of all sizes, it ensures real-time data syncing across devices, scalability, and a responsive interface.

1.5 Definitions & Acronyms

- **HRMS**: Human Resource Management System.
- **KRA**: Key Result Area.
- **ESS**: Employee Self-Service.
- **2FA**: Two-Factor Authentication.
- **FMEA**: Failure Mode and Effects Analysis.
- **Firestore**: Firebases NoSQL document database.
- **Firebase Auth**: Firebase Authentication service.

2 Overall Description

2.1 User Needs

- **HR Staff**: Quick access to employee data, payroll automation, and detailed reports for informed decision-making.
- **Employees**: A straightforward portal to manage leave requests, payslips, and personal details.
- **Managers**: A unified view for approving requests and tracking team performance metrics.
- **Admins**: A secure, scalable, and configurable backend to support organizational needs.

2.2 Assumptions & Dependencies

- Users have reliable internet access and modern devices/browsers.
- The system is hosted on Google Firebase for scalability and performance.
- Firebase Authentication secures user logins.
- Firestore is the primary database for efficient data storage.
- Firebase Cloud Functions power backend logic, such as payroll calculations and notifications.

3 System Requirements

3.1 Functional Requirements

3.1.1 Key Features

1. **Employee Management**: Create, update, and deactivate employee records in Firestore with a user-friendly interface.
2. **Authentication**: Secure login using Firebase Auth (Email/Password, Google Sign-In) with optional 2FA.

3. **Leave Management**: Submit, approve, or reject leave requests, stored in the `leaveRequests` collection.
4. **Attendance Tracking**: Record daily attendance in the `attendance` collection with timestamped logs.
5. **Payroll Processing**: Automate salary calculations and store payslips in the `payroll` collection and Firebase Storage.
6. **Recruitment Tracking**: Manage job postings, applicant statuses, and hiring pipelines in the `recruitment` collection.
7. **Performance Tracking**: Set KRAs, collect feedback, and store ratings in the `performance` collection.
8. **Reporting**: Generate dynamic reports and dashboards from Firestore data for HR and management.

3.1.2 EARS Format

- When an employee submits a leave request, the system shall store it in Firestore and notify the manager instantly.
- When payroll is processed, the system shall calculate salaries, generate payslips, and store them securely in Firebase Storage.
- When a manager updates a recruitment status, the system shall reflect changes in the recruitment dashboard in real time.

3.1.3 BDD / Gherkin Example

```
Feature: Leave Application
  Scenario: Employee submits a leave request
    Given the employee is authenticated via Firebase Auth
    When they submit a leave request form
    Then the request is saved in Firestore and the manager
       receives a notification
```

3.2 Non-Functional Requirements

3.2.1 Performance

- 95% of Firestore read/write operations must complete within 200ms for a smooth user experience.
- Use indexed queries to optimize database performance.
- Implement pagination for efficient handling of large datasets.

3.2.2 Security (Firebase-Specific)

- Enforce Firebase Auth with optional 2FA for enhanced security.
- Firestore Security Rules to ensure role-based access:

  - **Admins/HR**: Full access to all collections.
  - **Managers**: Read/write access to their teams data only.
  - **Employees**: Access restricted to their own records.
- Store sensitive documents (e.g., payslips, contracts) in Firebase Storage with secure, time-limited download URLs.

3.2.3 Usability, Reliability, Compliance

- Responsive, intuitive UI optimized for web and mobile devices.
- Firestores offline persistence for uninterrupted access in low-connectivity scenarios.
- GDPR-compliant data handling to protect user privacy.

3.3 External Interface Requirements

3.3.1 Performance Requirements

- Optimize Firestore queries to minimize reads and control Firebase costs.

3.3.2 Safety Requirements

- Enable Firebase App Check to prevent unauthorized access.
- Schedule daily Firestore backups for data recovery.

3.3.3 Security Requirements

- Limit failed login attempts and notify admins of suspicious activity.
- Store API keys securely in Firebase Config, avoiding exposure in source code.

3.3.4 Software Quality Attributes

- Modular Firebase Cloud Functions for payroll, notifications, and reporting.
- Clear separation of frontend (UI) and backend logic for maintainability.

3.3.5 Business Rules

- Leave requests cannot exceed available leave balance.
- Payroll data locks at month-end to prevent retroactive changes.

3.4 System Features

- Real-time employee data management with instant updates.
- Streamlined leave application and approval workflows.
- Accurate attendance logging with historical data access.
- Automated payroll processing with secure payslip delivery.

- Recruitment tracking with a visual pipeline.
- Performance evaluations with KRA-based feedback.
- Interactive dashboards for HR analytics and reporting.

4 Other Requirements

4.1 Database Requirements (Firebase)

- Firestore collections: `employees`, `attendance`, `leaveRequests`, `payroll`, `recruitment`, `performance`.
- Real-time syncing for seamless UI updates.
- Offline persistence for reliable access on mobile and web.

4.2 Legal & Regulatory Requirements

- Compliance with Indian labor laws and the IT Act.
- GDPR-compliant handling of personal data for global usability.

4.3 Internationalization & Localization

- Support for English and Hindi, with locale-specific date and currency formats.

4.4 Risk Management (FMEA Matrix)

Risk	**Severity**	**Probability**	**Mitigation**
Misconfigured Firestore rules	High	Medium	Conduct regular security audits
Billing spikes from excessive reads	Medium	Medium	Optimize queries and monitor usage
Unauthorized access	High	Low	Implement 2FA and Firebase App Check
Data loss	High	Low	Maintain daily Firestore backups

5 Appendices

5.1 Glossary

- **Firestore**: Firebases scalable NoSQL database.
- **Firebase Auth**: Authentication service for secure user logins.
- **App Check**: Firebase feature to block unauthorized app access.

5.2 Use Cases

- **Leave Request**: Employee submits leave $\rightarrow$ Stored in Firestore $\rightarrow$ Manager notified instantly.
- **Payroll Processing**: Run payroll $\rightarrow$ Cloud Functions calculate salaries $\rightarrow$ Store in Firestore $\rightarrow$ Payslips saved in Storage.

5.3 TBD List

- Finalized UI wireframes for a polished user experience.
- Detailed Firebase Security Rules for role-based access.
- Specific Cloud Function triggers for automation.