Software Requirements Specification (SRS)

1. Scope

1.1 Identification

- Identification No: **CWH-002**
- Title: **Centurion WorkerHub**
- Abbreviation: **CWH**
- Version No: **2.0**
- Release No: **2025**

1.2 System Overview

Centurion WorkerHub is a cloud-based workforce management system built with Flutter mobile frontend and Spring Boot backend. The system provides:

- **Workers:** Selfie-based attendance tracking with GPS verification, task management, real-time chat
- **Managers:** Team oversight, complaint assignment, work progress monitoring, analytics dashboard
- **Public Users:** Complaint submission, feedback provision, resolution tracking

Technology Stack:

- Frontend: Flutter (Cross-platform mobile app)
- Backend: Spring Boot (Java)
- Authentication: Google OAuth 2.0 API
- Database: MySQL 8.0+
- Cloud Platform: Google Cloud Platform (GCP)
- Real-time Communication: WebSocket integration

1.3 Document Overview

This document defines functional requirements, system architecture, and technical specifications for the Centurion WorkerHub project with modern cloud-native architecture and mobile-first approach.

1.4 Requirements of Software/Hardware

Software Requirements:

- Mobile OS: Android 7.0+ (API 24+), iOS 12.0+
- Backend: Spring Boot 3.x, Java 17+
  Database: MySQL 8.0+
- Cloud Services: Google Cloud Platform
- Development Tools: Flutter SDK 3.2+, Android Studio, IntelliJ IDEA
- Authentication: Google Identity Platform

Hardware Requirements:

- Mobile Device: 2GB RAM, 1GB storage, Camera, GPS
- Development: Intel i5/Apple M1+, 8GB+ RAM, 50GB storage
- Cloud Infrastructure: Auto-scaling GCP instances

1.5 Brief Software Functional Description

The system enables:

- GPS-based attendance with selfie verification within 50m radius of Centurion University Vizianagaram
- Role-based dashboards with real-time updates
- Complaint management workflow from submission to resolution
- Real-time chat system between workers and managers
- Work tracking and analytics with performance metrics

1.6 Functional Requirements

Core Functionalities:

- **Authentication & Authorisation**
  - Google OAuth 2.0 integration
  - Role-based access control (Worker/Manager/Public)
  - JWT token management
- **Attendance Management**

    - GPS location verification (18.1124, 83.4316 ±50m)
    - Front camera selfie capture
    - Check-in/check-out timestamps
    - Attendance analytics and reporting
- **Task & Work Management**
    - Task assignment and tracking
    - Work progress monitoring
    - Performance analytics
- **Communication System**
    - Real-time WebSocket chat
    - Push notifications
    - File sharing capabilities
- **Complaint Management**
    - Public complaint submission
    - Manager assignment workflow
    - Status tracking and feedback

1.7 Brief Description of the System

Configuration:

- Frontend: Flutter mobile app with BLoC state management
- Backend: Spring Boot REST APIs with micro-services architecture
- Database: MySQL with normalised schema
- Authentication: Google Identity Platform with OAuth 2.0
- Cloud: Google Cloud Run, Cloud SQL, Cloud Storage

Input Interfaces:

- Mobile App: Touch interface, camera input, GPS location
- Authentication: Google Sign-In integration
- Data Input: Forms, image uploads, location tracking

3. Requirements

3.1 Required Mode of Operation

- Mobile-first: Primary interface through Flutter mobile app
- Cloud-native: Deployed on Google Cloud Platform
- Real-time: WebSocket connections for live updates
- Offline-capable: Local storage for critical functions

3.2 System Capability Requirements

Performance Requirements:

- Response Time: API calls < 2 seconds
- Concurrent Users: 5000+ simultaneous users
- Availability: 99.9% uptime SLA
- Scalability: Auto-scaling based on demand

Security Requirements:

- Authentication: OAuth 2.0 with Google
- Data Encryption: TLS 1.3 in transit, AES-256 at rest
- Privacy: GDPR compliant data handling
- Access Control: Role-based permissions

3.3 Database Backend

MySQL Schema Design:

- Users (id, google_id, email, role, profile_data)
- Attendance (id, user_id, check_in, check_out, location, selfie_url)
- Tasks (id, assigned_to, assigned_by, status, description)
- Complaints (id, submitted_by, assigned_to, status, description)
- Chat_Messages (id, sender_id, receiver_id, message, timestamp)
- Notifications (id, user_id, type, content, read_status)

Software Design Document (SDD)

1. Introduction

1.1 Purpose

This document defines the technical architecture, system design, and implementation details for Centurion WorkerHub using modern cloud-native technologies and mobile-first approach.

1.2 Scope

The system provides comprehensive workforce management through:

- Flutter Mobile App: Cross-platform mobile interface
- Spring Boot Backend: Microservices architecture
- Google Cloud Platform: Scalable cloud infrastructure
- Real-time Features: WebSocket communication and push notifications

1.3 System Overview

- Frontend: Flutter with BLoC pattern, clean architecture
- Backend: Spring Boot with RESTful APIs
- Database: Cloud SQL (MySQL) with connection pooling
- Authentication: Google Identity Platform
- Deployment: Google Cloud Run with auto-scaling

2. Design Overview

2.1 Technology Stack Decisions

Frontend (Flutter):

- State Management: BLoC pattern for predictable state

- Navigation: go_router for type-safe routing
- UI Framework: Material Design 3
- Local Storage: SharedPreferences, SQLite
- Real-time: WebSocket integration

Backend (Spring Boot):

- Framework: Spring Boot 3.x with Spring Security
- Database: Spring Data JPA with MySQL
- Authentication: Spring Security OAuth2
- API Documentation: OpenAPI 3.0 (Swagger)
- Monitoring: Spring Actuator with Micrometer

Cloud Infrastructure (GCP):

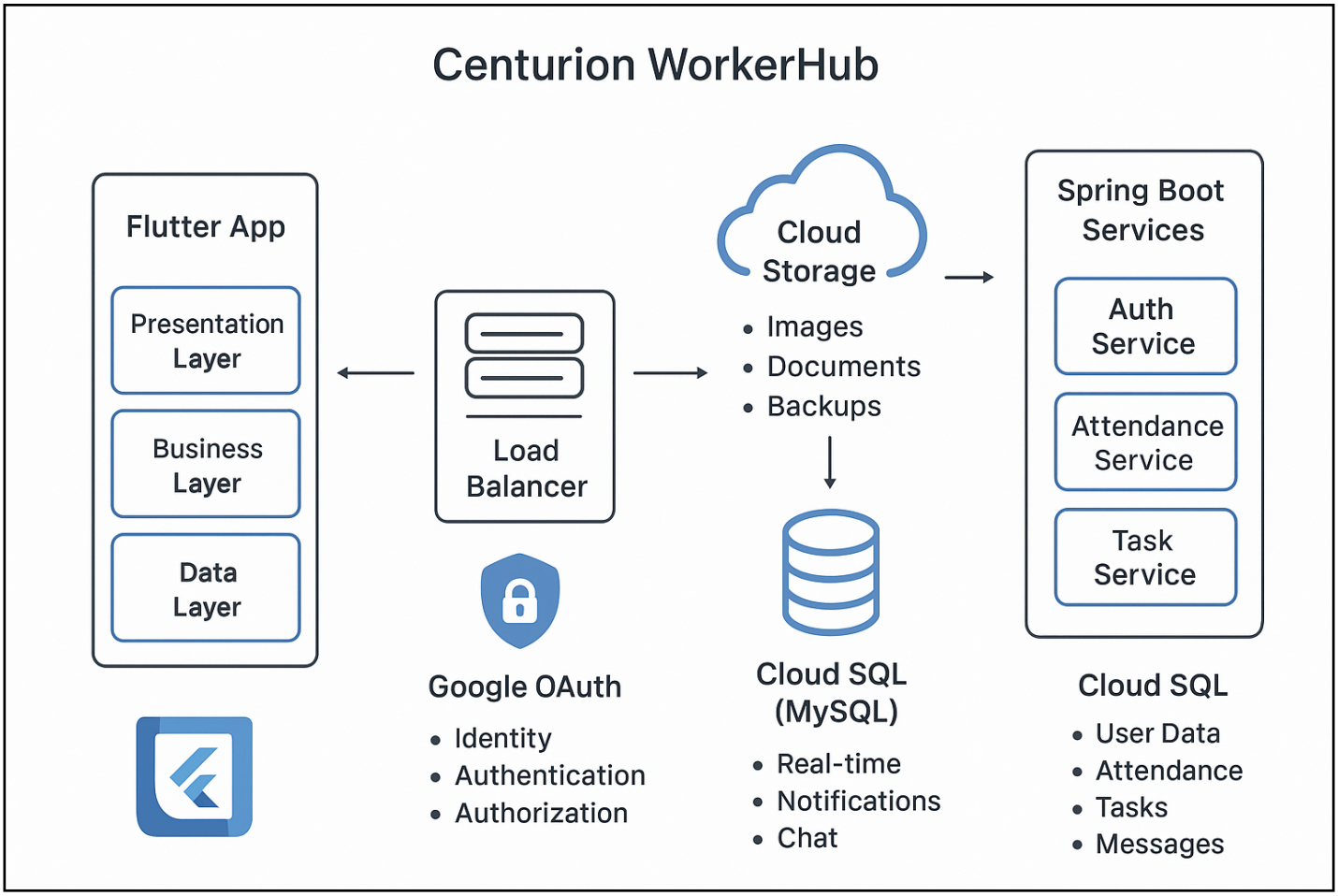
- Compute: Cloud Run for containerised services
- Database: Cloud SQL for MySQL
- Storage: Cloud Storage for files/images
- Authentication: Google Identity Platform
- Monitoring: Cloud Monitoring and Logging

2.2 Architecture Patterns

- Clean Architecture: Separation of concerns with layers
- Micro-services: Domain-driven service boundaries
- Event-Driven: Async communication between services
- CQRS: Command Query Responsibility Segregation for complex operations

4. Architectural Design

4.1 System Architecture



4.2 Component Design

Flutter App Components:

- Authentication Module: Google Sign-In integration
- Attendance Module: GPS + Camera functionality
- Dashboard Module: Role-based UI components
- Chat Module: Real-time messaging
- Settings Module: User preferences and profile

Spring Boot Micro-services:

- Auth Service: JWT token management, role validation
- User Service: Profile management, role assignment
- Attendance Service: Location verification, selfie processing
- Task Service: Work assignment and tracking
- Chat Service: WebSocket message handling
- Notification Service: Push notification delivery

4.3 Data Flow Architecture

Authentication Flow:

1. User initiates Google Sign-In in Flutter app
2. Google validates credentials
3. Backend receives OAuth token, creates/updates user
4. JWT issued for API calls
5. Role-based dashboard loads

Attendance Flow:

1. Worker opens attendance screen
2. GPS location validated (±50m radius)
3. Front camera selfie captured
4. Image uploaded to Cloud Storage
5. Attendance record created with timestamp + location

6. Real-time update sent to manager dashboard

5. Database Design

5.1 Entity Relationship Model

Core tables :

```
CREATE TABLE users (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    google_id VARCHAR(255) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    name VARCHAR(255) NOT NULL,
    role ENUM('WORKER', 'MANAGER', 'PUBLIC') NOT NULL,
    profile_image_url TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);


-- Attendance with GPS and selfie verification
CREATE TABLE attendance_records (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT NOT NULL,
    check_in_time TIMESTAMP NOT NULL,
    check_out_time TIMESTAMP NULL,
    latitude DECIMAL(10, 8) NOT NULL,
```

```
    longitude DECIMAL(11, 8) NOT NULL,
    selfie_url TEXT NOT NULL,
    work_hours DECIMAL(4, 2) DEFAULT 0,
    status ENUM('PRESENT', 'LATE', 'ABSENT') DEFAULT 'PRESENT',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- Tasks and work assignments
CREATE TABLE tasks (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    assigned_to BIGINT NOT NULL,
    assigned_by BIGINT NOT NULL,
    status ENUM('PENDING', 'IN_PROGRESS', 'COMPLETED', 'CANCELLED') DEFAULT
'PENDING',
    priority ENUM('LOW', 'MEDIUM', 'HIGH', 'URGENT') DEFAULT 'MEDIUM',
    due_date TIMESTAMP NULL,
    completed_at TIMESTAMP NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (assigned_to) REFERENCES users(id),
    FOREIGN KEY (assigned_by) REFERENCES users(id)
);

-- Real-time chat messages
CREATE TABLE chat_messages (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
```

```
    sender_id BIGINT NOT NULL,
    receiver_id BIGINT NOT NULL,
    message TEXT NOT NULL,
    message_type ENUM('TEXT', 'IMAGE', 'FILE') DEFAULT 'TEXT',
    file_url TEXT NULL,
    is_read BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (sender_id) REFERENCES users(id),
    FOREIGN KEY (receiver_id) REFERENCES users(id)
);

-- Complaint management
CREATE TABLE complaints (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    submitted_by BIGINT NOT NULL,
    assigned_to BIGINT NULL,
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    status ENUM('SUBMITTED', 'ASSIGNED', 'IN_PROGRESS', 'RESOLVED', 'CLOSED')
DEFAULT 'SUBMITTED',
    priority ENUM('LOW', 'MEDIUM', 'HIGH', 'URGENT') DEFAULT 'MEDIUM',
    images JSON NULL,
    resolution_notes TEXT NULL,
    resolved_at TIMESTAMP NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (submitted_by) REFERENCES users(id),
    FOREIGN KEY (assigned_to) REFERENCES users(id)
);
```

5.2 Cloud Infrastructure Design

- Cloud Run: Auto-scaling Spring Boot services
- Cloud SQL: Managed MySQL
- Cloud Storage: File/image hosting
- Load Balancer: Traffic + SSL termination
- Cloud Monitoring & Logging: Performance + logs
- CI/CD: GitHub Actions → Cloud Build → Cloud Run
- Database Migration: Flyway for schema versioning