# Employee Performance Dashboard — Project Documentation

**Project Name:** Employee Performance Dashboard (Weekly Top & Weak Performers)
**Stack:** Frontend — HTML, CSS, JavaScript, Bootstrap, Angular
Backend — Java, Spring Boot (REST), MySQL
**Purpose:** Provide weekly insights on top-performing and weak-performing employees across teams; enable managers to view, filter, export, and act on performance data.

---

## 1. Project Overview

**Goal:**
Build a web application that ingests weekly performance metrics (automated or manual), calculates rankings, displays top and weak performers, and provides exports, notifications, comments, and action trackers for managers and HR.

**Primary Users & Roles**

- **Admin:** Manage users, teams, metrics, view reports, set thresholds.
- **Manager:** View team performance, comment, assign improvement plans, export reports.
- **Employee:** View own performance, view feedback, acknowledge comments.
- **System/Integrations:** Automated ingestion from time/attendance, task tracking, or manual CSV upload.

**Key Features**

- Weekly Top Performers and Weak Performers list (company / department / team).
- Filters: Date range (week), Department, Team, Location, Role.
- Drill-down: Employee detail (metrics, trend chart, history, feedback).
- Manual or automated data ingestion (CSV upload or API).
- Scorecard calculation engine (weighted metrics).
- Alerts/notifications to managers for weak performers.
- Export to CSV/PDF.
- Role-based access control.
- Audit logs and history.

---

## 2. Functional Requirements

1. **Authentication & Authorization**
    - Login (JWT).
    - Role-based access: ADMIN, MANAGER, EMPLOYEE.
2. **User & Org Management**

- CRUD for users, departments, teams, roles.
- Assign manager-to-team relations.

3. **Performance Data**
   - Ingest weekly metrics: tasks completed, quality score, attendance, client feedback, SLA adherence, etc.
   - Manual entry form and CSV upload endpoint.
   - Data validation.
4. **Scoring & Ranking**
   - Score calculation per employee: weighted sum of metrics.
   - Weekly ranking and designation: Top 5, Weak 5 (configurable counts).
5. **Dashboards**
   - Company-level summary: average score, distribution, top/weak lists.
   - Manager dashboard: team performance, actions pending.
   - Employee dashboard: own weekly score, trends.
6. **Actions & Feedback**
   - Managers can add comments, assign improvement tasks, set review reminders.
   - Employees can acknowledge feedback.
7. **Reporting**
   - Export lists & detailed reports to CSV/PDF.
   - Scheduled weekly report emails (optional).
8. **Audit & Logs**
   - Track data uploads, changes, and user actions.
9. **Non-functional**
   - Responsive UI, secure APIs, scalable infra, backup & restore, monitoring.

---

**3. Non-Functional Requirements**

- **Security:** HTTPS, JWT, secure password storage (bcrypt), RBAC.
- **Performance:** Dashboard loads < 3s for up to 10k employees (design for scaling).
- **Scalability:** Use stateless APIs, RDS for DB, S3 for exports.
- **Availability:** Deploy with health checks and auto-restart (containerization).
- **Maintainability:** Clean code, modular services, API docs (OpenAPI/Swagger).

- **Backup & Recovery:** Daily DB backups, export retention policy.
- **Logging & Monitoring:** Centralized logs (ELK/CloudWatch), alerting.

---

**4. System Architecture (High-level)**

```
[Angular Frontend] <--HTTPS--> [Spring Boot REST API] <---> [MySQL RDS]
       |                          |
       |                          +--> [S3/Blob] (exports, uploads)
       |                          +--> [Message Queue] (optional for heavy ingestion)
       |
 (Static hosting - S3/CloudFront or web server)
```

Optional additions:

- Worker service (Spring Boot) for heavy calculations and scheduled tasks (ranking) via message queue.
- Caching layer (Redis) for frequently-accessed dashboards.
- CI/CD pipeline (GitHub Actions/Jenkins) -> Docker images -> Registry -> Kubernetes/ECS/EC2.

---

**5. Data Model (MySQL) — Core Tables**

**SQL DDL (sample)**

```
CREATE TABLE departments (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE teams (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  department_id BIGINT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (department_id) REFERENCES departments(id)
);
```

```
CREATE TABLE users (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(100) UNIQUE NOT NULL,
  full_name VARCHAR(150),
  email VARCHAR(150) UNIQUE,
  password_hash VARCHAR(255),
  role ENUM('ADMIN','MANAGER','EMPLOYEE') DEFAULT 'EMPLOYEE',
  team_id BIGINT,
  manager_id BIGINT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (team_id) REFERENCES teams(id),
  FOREIGN KEY (manager_id) REFERENCES users(id)
);

CREATE TABLE performance_metrics (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  code VARCHAR(50) NOT NULL, -- e.g., TASKS_COMPLETED, QUALITY_SCORE, ATTENDANCE
  name VARCHAR(150) NOT NULL,
  weight DECIMAL(5,2) DEFAULT 1.0, -- relative weight in scoring
  active BOOLEAN DEFAULT TRUE
);

CREATE TABLE weekly_performance (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT NOT NULL,
  metric_id BIGINT NOT NULL,
  week_start DATE NOT NULL, -- week identifier
  metric_value DECIMAL(10,2) NOT NULL,
  created_by BIGINT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id),
  FOREIGN KEY (metric_id) REFERENCES performance_metrics(id)
);

CREATE TABLE weekly_scores (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT NOT NULL,
  week_start DATE NOT NULL,
  score DECIMAL(10,2) NOT NULL,
  rank INT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE feedbacks (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT NOT NULL, -- subject employee
  created_by BIGINT NOT NULL, -- manager
  week_start DATE NOT NULL,
  comment TEXT,
  action_required BOOLEAN DEFAULT FALSE,
  due_date DATE,
  status ENUM('OPEN','IN_PROGRESS','CLOSED') DEFAULT 'OPEN',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id),
  FOREIGN KEY (created_by) REFERENCES users(id)
);

CREATE TABLE audit_logs (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT,
  action VARCHAR(200),
  details TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

**Notes:**

- performance_metrics defines what metrics exist and their weight.
- weekly_performance stores raw values for each metric per user per week.
- weekly_scores stores computed aggregated score and ranking for querying dashboards quickly.
- Add indexes on user_id, week_start for performance.

---

**6. Scoring Engine**

**Scoring formula (example):**

For each user and week:

score = SUM_over_metrics( normalized_value(metric_value) * metric_weight )

**Normalization approaches:**

- Min-Max scaling per metric per week: norm = (value - min_week_value) / (max_week_value - min_week_value) → 0..1
- Or use predefined thresholds.

**Steps:**

1. For a given week_start, fetch all performance_metrics (active) and their weights.
2. For each metric, compute normalization across employees for that week.
3. Compute weighted sum → store in weekly_scores.
4. Compute ranking (ORDER BY score DESC) and assign rank.

**Execution:**

- Triggered by scheduled job (cron) at end of each week OR invoked after all weekly data ingestion is complete.
- Implement worker (Spring scheduled task) or push to queue for processing.

---

**7. API Specification (REST)**

Base: https://{host}/api/v1

Authentication: Authorization: Bearer <JWT>

**Auth**

- POST /auth/login
  Request: { "username": "x", "password": "y" }
  Response: { "token":"...", "expiresIn":3600, "role":"MANAGER" }

**Users & Org**

- GET /users (ADMIN) — query params: teamId, role
- POST /users — create user
- PUT /users/{id} — update user
- GET /teams, POST /teams, GET /departments

**Metrics & Data**

- GET /metrics — list metrics
- POST /metrics — create metric (ADMIN)
- POST /performance/upload — upload CSV or JSON list for weekly_performance
  - Accepts multipart CSV or JSON body:
  - [
  - {"userId": 101, "metricCode":"TASKS_COMPLETED","weekStart":"2025-09-01","metricValue":15},
  - ...
  - ]
- GET /performance?weekStart=2025-09-01&teamId=5 — raw metrics

**Scoring & Rankings**

- POST /scores/compute?weekStart=2025-09-01 — trigger compute (ADMIN or scheduled)
- GET /scores?weekStart=2025-09-01&teamId=5&sort=desc&limit=10 — top performers
- GET /scores/weak?weekStart=2025-09-01&teamId=5&limit=10 — weak performers (lowest scores)

**Feedback & Actions**

- POST /feedbacks — manager adds feedback/action
- GET /feedbacks?userId=101&weekStart=...
- PUT /feedbacks/{id} — update status

**Reports & Exports**

- GET /reports/weekly?weekStart=...&format=csv — download CSV
- GET /reports/weekly?weekStart=...&format=pdf — download PDF (generated from server or use client-side PDF)

**Audit**

- GET /auditlogs?userId=...&action=...

---

**8. Frontend Details (Angular + Bootstrap)**

**Overall Structure**

- Angular app (CLI) with modules: auth, dashboard, admin, manager, employee, shared.
- Responsive UI with Bootstrap 5.

**Key Routes**

- /login
- /dashboard (role-aware)
- /dashboard/company (ADMIN view)
- /dashboard/manager (Manager view)
- /employee/:id (employee detail)
- /admin/metrics
- /upload (manual CSV upload page)
- /reports

**Components**

- LoginComponent — auth
- NavBarComponent — role-driven menu
- TopWeakListComponent — shows top/weak lists with filters
- EmployeeCardComponent — small card with name, score, change
- EmployeeDetailComponent — metrics chart (chart.js), feedback list
- UploadCsvComponent — upload & validate CSV
- FeedbackModalComponent — add feedback/action
- ReportsComponent — export options
- AdminMetricsComponent — configure metrics/weights

**Services**

- AuthService — login, token
- ApiService — common HTTP calls
- ScoresService — fetch scores, compute calls
- UploadService — CSV upload

- NotificationService — toast & email triggers

**UI/UX**

- Use Bootstrap cards, table for lists.
- Charting: Chart.js or ngx-charts for weekly trends.
- Highlight top performers (green) and weak performers (red).
- Accessibility: aria labels, keyboard navigation.

**Sample UI Wireframe Concepts**

- Dashboard top: Week selector + summary tiles (Avg Score, Top Dept, Weak Dept).
- Middle: Two columns — Top Performers (left) and Weak Performers (right).
- Bottom: Filters & table with pagination + export button.

---

## 9. Backend (Spring Boot) Design

**Project Structure (maven)**

```
src/main/java/com/company/perf
  /controller
  /service
  /repository
  /model (entities)
  /dto
  /config (security, swagger)
  /util
```

**Entities**

- Map to tables in section 5.

**Repository**

- Spring Data JPA JpaRepository interfaces for UserRepository, WeeklyPerformanceRepository, WeeklyScoresRepository, FeedbackRepository, etc.

**Services**

- PerformanceIngestionService — validates and stores raw metrics.
- ScoringService — normalization, weighting, score storage.
- ReportService — export CSV/PDF.
- NotificationService — email/sms triggers to managers.

- UserService — user CRUD and RBAC.

**Controllers**

- AuthController — login/register
- PerformanceController — upload & retrieve
- ScoreController — compute & query
- AdminController — manage metrics, users
- ReportController — export

**Security**

- Spring Security with JWT filters.
- Passwords stored hashed (BCrypt).
- Roles checked on endpoints via @PreAuthorize("hasRole('ADMIN')").

**Exception Handling**

- Centralized @ControllerAdvice for API errors.

**Testing**

- Unit tests with JUnit + Mockito.
- Integration tests with Spring Boot Test and an in-memory DB (H2).
- E2E tests for APIs (Postman/Newman or RestAssured).

---

**10. CSV Upload Format (example)**

CSV columns for ingestion:

```
user_email,metric_code,week_start,metric_value

alice@example.com,TASKS_COMPLETED,2025-09-01,23

bob@example.com,QUALITY_SCORE,2025-09-01,4.6
```

Validation rules:

- user_email or user_id must exist.
- metric_code is known.
- week_start is a Monday (system rule).
- metric_value is numeric and within bounds.

On errors: return structured error (line number, issue).

---

**11. Deployment & DevOps (recommended)**

**Containerization**

- Dockerize Spring Boot app and Angular app.
- Dockerfiles:
    - Backend: multi-stage build (maven build → jre image).
    - Frontend: build assets → serve via nginx.

**Registry**

- Push images to Docker Hub / ECR.

**Orchestration**

- Kubernetes (EKS) or ECS.
- Deploy using Helm charts (recommended) or Kubernetes manifests.

**Database**

- Managed MySQL (Amazon RDS) or self-hosted in Kubernetes.
- Enable automated backups and multi-AZ for RDS.

**CI/CD**

- GitHub Actions or Jenkins Pipeline:
    - on: push to main → run tests → build docker images → push to registry → deploy (kubectl/helm).
    - Use infra IaC: Terraform scripts for RDS, VPC, EKS cluster.

**Secrets**

- Use AWS Secrets Manager / Kubernetes Secrets for DB credentials and JWT secret.

**Monitoring & Logging**

- Application logs to CloudWatch / ELK.
- Metrics to Prometheus + Grafana (K8s).
- Alerts to Slack/Email for job failures or critical alerts.

---

## 12. Testing Plan

**Test Types**

- Unit Tests: Service & repo layers (JUnit + Mockito).
- Integration Tests: Controller endpoints with in-memory DB.
- API Contract Tests: OpenAPI validation.
- Functional Tests: Manual and automated (Selenium or Cypress for frontend).

- Load Testing: JMeter for ranking jobs and dashboard under concurrency.
- Security Tests: Vulnerability scan, dependency-check (SCA).
- UAT: With HR/Managers on sample data.

**Test Cases (examples)**

- Verify CSV ingestion with valid & invalid rows.
- Verify score computation correctness given sample dataset.
- Verify RBAC: Manager cannot access admin endpoints.
- Dashboard loads top/weak lists correctly and filters work.
- Export CSV yields correct values.

---

## 13. Notifications & Email

- Send automated emails weekly to managers with top/weak lists or alerts if team avg drops below threshold.
- Use templated emails via SMTP or AWS SES.
- Optionally integrate Slack/webhook for immediate alerts.

---

## 14. Audit, Compliance & Data Retention

- Log ingestion & manual edits in audit_logs.
- Retain raw weekly performance for X months (policy).
- Secure PII: encrypt sensitive fields; limit access.

---

## 15. Roles & Responsibilities (Team)

- **Product Owner / HR Lead:** Define metrics, thresholds, business logic.
- **Project Manager:** Coordinate, track deliverables.
- **Frontend Developer(s):** Angular UI, charts, responsive design.
- **Backend Developer(s):** Spring Boot APIs, scoring engine, integrations.
- **DBA:** Schema design, backup, indexing, optimization.
- **DevOps Engineer:** CI/CD, containerization, deployment, monitoring.
- **QA Engineer:** Test plans, automation tests, UAT coordination.
- **UX Designer (optional):** Wireframes, usability.

---

**16. Deliverables**

- Requirements Document & Data Dictionary.
- ER Diagram and SQL scripts.
- REST API Documentation (Swagger/OpenAPI).
- Angular frontend codebase.
- Spring Boot backend codebase.
- Docker images and Helm charts.
- CI/CD pipelines (GitHub Actions/Jenkins files).
- Test cases & test reports.
- Deployment & runbook docs.
- User manuals (Admin/Manager/Employee).
- Demo data & production seed script.

---

**17. Sample SQL Seed (example metrics + sample data)**

```
INSERT INTO performance_metrics (code,name,weight) VALUES
('TASKS_COMPLETED','Tasks Completed', 0.30),
('QUALITY_SCORE','Quality Score', 0.30),
('ATTENDANCE','Attendance %', 0.20),
('CLIENT_FEEDBACK','Client Feedback', 0.20);

-- Sample users
INSERT INTO departments (name) VALUES ('Engineering');
INSERT INTO teams (name, department_id) VALUES ('Backend',1);

INSERT INTO users (username,full_name,email,password_hash,role,team_id) VALUES
('alice','Alice Kumar','alice@example.com','$2a$10$...', 'EMPLOYEE', 1),
('bob','Bob Rao','bob@example.com','$2a$10$...', 'MANAGER',1);
```

---

**18. Example: Compute Scores Pseudocode (Spring Service)**

```
public void computeWeeklyScores(LocalDate weekStart) {
  List<Metric> metrics = metricRepo.findAllActive();
```

```
    List<User> employees = userRepo.findAllActive();
    Map<Long, Map<Long, Double>> rawValues = perfRepo.getValuesForWeek(weekStart); // userId -
> (metricId -> value)

    for (Metric m : metrics) {
        // find min & max across users for normalization
        double min = findMinForMetric(rawValues, m.getId());
        double max = findMaxForMetric(rawValues, m.getId());
        for (User u : employees) {
            double val = rawValues.getOrDefault(u.getId(), Map.of()).getOrDefault(m.getId(), 0.0);
            double norm = (max == min) ? 1.0 : (val - min) / (max - min); // handle division
            double weighted = norm * m.getWeight();
            scoresAccum.get(u.getId()).add(weighted);
        }
    }
    // compute final score = sum of weighted metrics
    // persist to weekly_scores and compute rank
}
```

---

**19. Security Considerations**

- Use HTTPS everywhere.
- JWT expiry short; refresh tokens secured.
- Passwords hashed (BCrypt).
- Input validation and sanitization (prevent SQL injection/XSS).
- Limit upload sizes and validate CSV content.
- RBAC at API level with method security annotations.

---

**20. Implementation Checklist (milestone-style but no time estimates)**

- Finalize metric definitions & weights with HR.
- DB schema & initial seed data.
- Authentication module (JWT).

- CRUD for metrics and org entities.
- CSV upload and validation endpoint.
- Scoring engine & scheduled job.
- APIs for dashboards (top/weak lists).
- Angular frontend pages & charts.
- Feedback and action module.
- Export/Reporting feature.
- CI/CD pipelines and Docker images.
- Deploy to staging and setup monitoring.
- QA, load testing, and UAT.
- Production deployment and runbook.

---

**21. Documentation & Handover**

- Maintain an internal Confluence or README with:
    - Architecture diagrams
    - API docs (Swagger)
    - Developer setup steps (local dev, running tests)
    - DB migration steps (Liquibase/Flyway recommended)
    - Production runbook (backup, restore, health-checks)
    - On-call escalation matrix

---

**22. Extras & Enhancements (future)**

- Add ML-based anomaly detection for sudden drops in performance.
- Slack integration for immediate manager alerts.
- Gamification badges for top performers.
- Mobile app or PWA for employee quick view.