Design Specification

Role-Based Internal Platform for Targeted Content Delivery

Group 20

December 5, 2025

Contents

1 Introduction

This Design Specification outlines the structural breakdown of the Platform, including:

- All frontend pages.
- All backend modules.
- Tools and technologies used.
- A complete list of packages, dependencies, and Docker resources required.

It supports both the SRS and SDD by providing a concrete implementation plan.

2 Frontend Design

The frontend is a React-based web interface.

2.1 Login Page

- Fields: Email, Password.
- Actions: Log In.
- Behavior:
  - Sends POST /api/auth/login.
  - Redirects based on user role.

2.2 End User Dashboard

- Displays personalized targeted content (announcements, policies, training).
- Filters: Category, Status (All, Mandatory, Completed).
- Search bar for titles/keywords.
- Each content card shows title, summary, mandatory badge.

2.3 Content Detail Page

- Full content body, metadata, start/end dates.
- Acknowledge button for mandatory content.
- Sends:
  - POST /api/content/{id}/view
  - POST /api/content/{id}/acknowledge

2.4 Admin Content List

- Table of all content items.
- Actions: Create, Edit, Archive.
- Columns: Title, Mandatory, Dates, Status.

2.5 Content Editor & Targeting Rules Page

- Fields: Title, Body, Category, Start/End Dates, Mandatory Flag.
- Targeting Rules:
  - Field (role, department, location)
  - Operator
  - Value

2.6 Reporting Dashboard

- Displays views, acknowledgements, completion rates.
- Export CSV option.

2.7 User Management Page

- Create/Update users.
- Assign roles.
- Activate/Deactivate accounts.

3 Backend Design

The backend is one Python app that the website talks to. It handles things like logging in, checking rules, and getting data from the database.

3.1 Auth Module

- Login, logout, password validation.
- JWT-based session management.
- Role-based access checks.

3.2 User & Role Management

- CRUD operations for users.
- Role assignments.

3.3 Content Management

- Create, edit, schedule, archive content.
- Store metadata for targeting.

3.4 Targeting Engine

- Evaluates targeting rules against user attributes.
- Supports AND/OR logic groups.

3.5 Delivery Service

- Generates personalized content feed.
- Logs view and acknowledgement events.

3.6 Reporting Engine

- Aggregates engagement data.
- Returns compliance summaries.

4 Database Design

Core tables used:

- **User**: id, email, password hash, department, location, active flag.
- **Role**: id, name.
- **UserRole**: user_id, role_id.
- **ContentItem**: id, title, body, mandatory, dates.
- **TargetRule**: id, content_id, field, operator, value.
- **DeliveryEvent**: id, content_id, user_id, type, timestamp.

5 Tools Used

5.1 Languages & Frameworks

- Python (backend)
- Flask (API framework)
- React (frontend)
- JavaScript/TypeScript
- PostgreSQL (database)

5.2 Development Tools

- Docker
- Jira
- GitHub
- LaTeX

6 Dependencies & Packages

6.1 Backend Dependencies (Python)

- Flask
- Flask-JWT-Extended (authentication)
- SQLAlchemy (ORM)
- psycopg2-binary (PostgreSQL connector)
- passlib[bcrypt] (password hashing)
- python-dotenv

6.2 Frontend Dependencies (React)

- react
- react-dom
- react-router-dom
- axios (API calls)
- UI library (Material UI or similar)

6.3 Dev Dependencies

- eslint, prettier
- typescript (if used)

6.4 Docker Components

- Backend Container:
  - Base: python:3.12-slim
  - Installs backend dependencies
- Frontend Container:
  - Base: node:20-alpine
  - Builds React app
- Database Container:
  - Base: postgres:16-alpine
  - Stores persistent data