Leave Management System (LMS)

PREPARED BY SHIVANI S (30447)

MICROSERVICES-BASED PROJECT WITH DOCKER COMPOSE DEPLOYMENT

Architecture Used - Microservices

- - Microservices based system
- - Independent services: Employee Service & Manager Service
- - Each service has its own SQLite database
- - API Gateway connects services
- - Frontend communicates only with Gateway
- - Deployed using Docker Compose

Tools and Technologies:

- Frontend(Angular-Port 8090)
  - Framework: Angular
  - Components:
    - **Login Component** – Employee & Manager login handling
    - **Employee Component** – Apply leave, view leave status, leave history
    - **Manager Component** – Review leave requests, approve/reject
    - **Services (Angular Services)** – HTTP communication with backend via API Gateway

Tools and Framework:

- Backend (Microservices Architecture with Flask):
  - Employee Service (Port 8001):
    - Handles employee records and leave requests
    - Database: SQLite (Employee DB, Leave DB)
    - APIs: Apply Leave, View Leave History

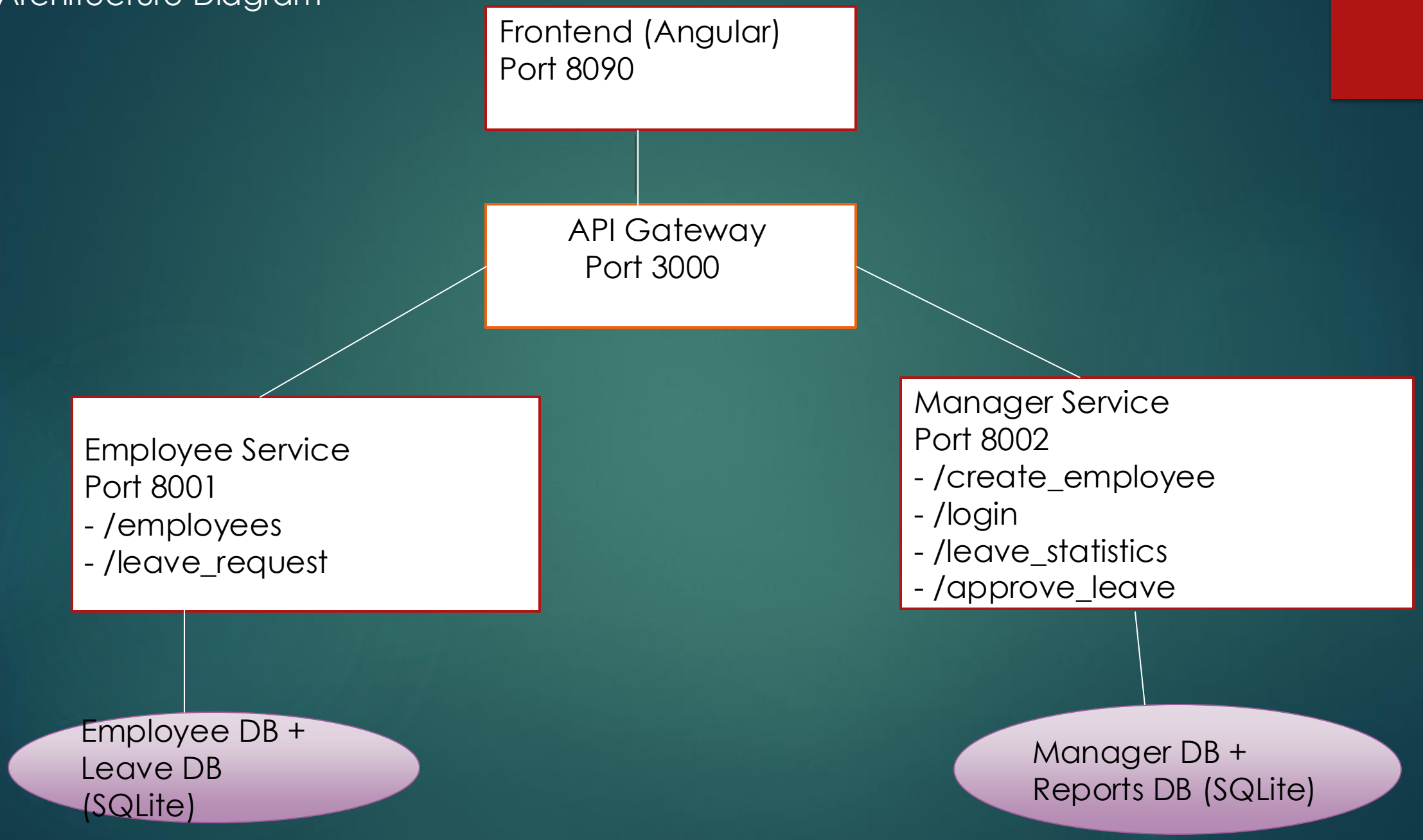
Tools and Framework:

- Manager Service (Port 8002)
  - Handles approval/rejection of leave requests
  - Generates reports for managers
  - Database: SQLite (Manager DB, Reports DB)
  - APIs: Approve/Reject Leaves, View Reports

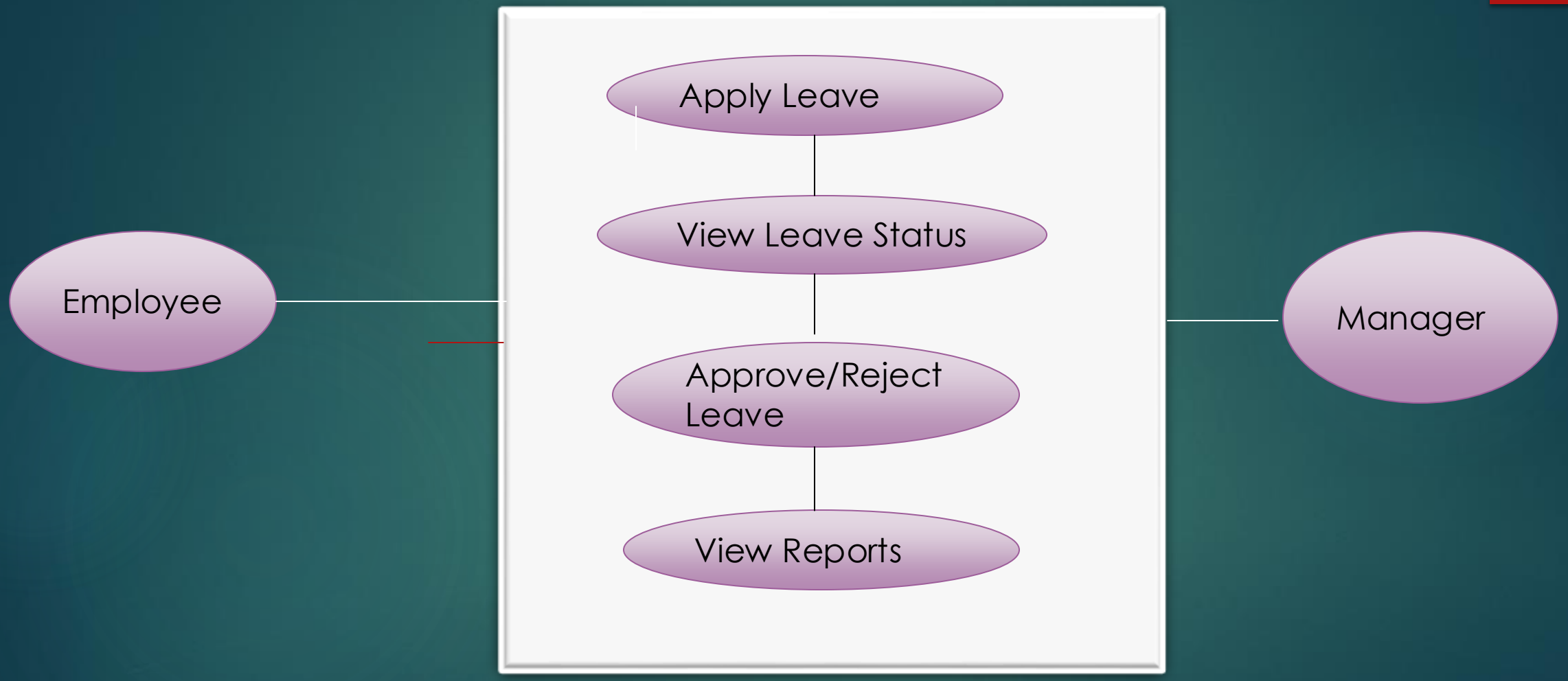
Tools and Framework:

- API Gateway (Port 3000):
  - Routes all frontend requests to respective microservices
  - Provides single entry point for Angular frontend
  - Handles authentication and request forwarding

Architecture Diagram



Use Case Diagram



Employee Service

- - Manages employees and leave requests
- - APIs:
- • GET /employees → list employees
- • POST /leave_request → apply for leave
- - Databases: Employee DB, Leave DB (SQLite)
- - Runs on Port 8001
- - Has its own Dockerfile & requirements.txt

Manager Service

- - Handles approvals and statistics
- - APIs:
- • POST /create_employee → Add new employee
- • POST /login → Manager login
- • GET /leave_statistics → View reports
- • POST /approve_leave → Approve/Reject leave
- - Databases: Manager DB, Reports DB (SQLite)
- - Runs on Port 8002
- - Has its own Dockerfile & requirements.txt

Gateway Service

- - Acts as API Gateway
- - Routes requests to Employee/Manager services
- - Provides single entry point
- - Runs on Port 8090
- - Has its own Dockerfile & requirements.txt

Frontend

- - User Interface for employees and managers
- - Employees apply leave, Managers approve/reject
- - Communicates only with Gateway
- - Runs on Port 3000
- - Has its own Dockerfile & requirements.txt

Containerization

- - Each service packaged in its own Docker container
- - Containers:
- • leave-employee
- • leave-manager
- • leave-gateway
- • leave-frontend
- - SQLite DBs stored in container volumes

Deployment - Docker Compose

- - docker-compose.yaml defines all services & databases
- - Exposed Ports:
- • Frontend: 3000
- • Gateway: 8090
- • Employee Service: 8001
- • Manager Service: 8002
- - One command starts everything:
- docker-compose up

Advantages

- - Independent services → easier updates
- - Fault isolation → one crash doesn't stop system
- - Scalable → run more instances if needed
- - Portable → runs anywhere with Docker
- - Simple deployment → one command with Docker Compose

Future Scope

- - Add role-based access control
- - Add email/SMS notifications
- - Logging & monitoring
- - Improve frontend analytics

Thank You!
Prepared by Shivani S (30447)