EduLearn - Runbook Document

1. App Introduction

EduLearn is a scalable and user-friendly learning management system designed to help educational institutions manage courses, instructors, students, and learning content. The application provides a comprehensive platform for online learning, including course creation, content management, student enrollment, progress tracking, and assessments. EduLearn is built using the MEAN stack (MongoDB, Express.js, Angular, Node.js) and is deployed on Azure Kubernetes Service (AKS).

2. Architecture of the App

The architecture of EduLearn follows a microservices-based approach, with each service handling a specific business function. The core components are:

- **Frontend**: Angular application that provides a responsive and interactive user interface.

Backend API: Node.js with Express.js serving as the REST API layer.

- **Database**: MongoDB for storing application data.
- **Authentication Service**: Uses JSON Web Tokens (JWT) for secure user authentication.
- **Containerization**: All services are containerized using Docker.
- **Orchestration**: Deployed on Azure Kubernetes Service (AKS) for scaling and management.
- **CI/CD Pipeline**: Azure DevOps for automated testing, integration, and deployment.

3. Microservices List and Its Purpose

Microservice	Purpose
User Management	Handles user registration, authentication, and profile management.
Course Management	Manages course creation, editing, deletion, and publishing.
Content Delivery	Provides content management for courses, including videos, documents, etc.
Enrollment Service	Manages student enrollment and tracks their progress in courses.
Assessment Service	Handles the creation and evaluation of quizzes and exams.
Notification Service	Sends email and in-app notifications to users about course updates.

4. Environment URLs

- **Development**: https://dev-edulearn.example.com
- **Test**: https://test-edulearn.example.com
- **Staging**: https://stage-edulearn.example.com
- **Production**: https://www.edulearn.example.com

5. Contacts

Project Contacts

Role	Name	Contact Information
Project Owner	John Doe	john.doe@example.com
Technical Lead	Jane Smith	jane.smith@example.com
Service Manager	Richard Roe	richard.roe@example.com
Delivery Manager	Amanda Johnson	amanda.johnson@example.com
Developers	Team Dev	devteam@example.com

Support Contacts

Role	Name	Contact Information
Support Lead	Michael Green	michael.green@example.com
Support Engineer	Sarah Lee	sarah.lee@example.com
Support Engineer	David Brown	david.brown@example.com

6. Deployment Plan Using Azure DevOps

- **Source Control**: The code is maintained in a Git repository hosted on Azure Repos.
- **Build Pipeline**:
    - Triggers on every commit to the main branch.
    - Runs unit tests and code analysis.

    - Builds Docker images for each microservice.
- **Release Pipeline**:
    - Stages: Dev -> Test -> Stage -> Prod
    - Deploys Docker images to AKS.
    - Runs integration tests in Test and Stage environments.
    - Manual approval is required for deployment to the Production environment.
- **Rollback Strategy**:
    - In case of a failed deployment, the pipeline can roll back to the previous stable version by redeploying the last successful build.

7. MongoDB Database Backup Plan

- **Automated Backups**:
    - Daily backups are scheduled using Azure Backup.
    - Backups are stored in Azure Blob Storage with a retention period of 30 days.
- **Manual Backups**:
    - Can be initiated through the Azure portal.
    - Snapshots are taken before major releases or migrations.
- **Restoration**:
    - Restorations can be performed using the Azure portal, with an option to restore to the same or a different MongoDB instance.

8. Existing Issues in this App

- **Performance Issues**: The Course Management microservice occasionally experiences slow response times under heavy load.
- **Scalability**: The Notification Service currently does not scale well with the number of concurrent users.
- **UI/UX Improvements**: The frontend has some usability issues on mobile devices, particularly in the enrollment section.

9. Downstream and Upstream Systems

- **Downstream Systems**:
    - **Payment Gateway**: For processing course payments.
    - **Email Service**: For sending notifications and course updates to users.
- **Upstream Systems**:
    - **HR System**: For syncing instructor data.
    - **Content Management System (CMS)**: For importing course materials and multimedia content.

10. References

- **Project Documentation**: EduLearn Project Wiki
- **Angular Documentation**: Angular Framework
- **MongoDB Documentation**: MongoDB Documentation
- **Azure DevOps**: Azure DevOps Documentation