**Deliverables for Resource Upskilling Portal – Capstone Project**

### **1. Project Overview (PPT Slide)**

* **Title**: Resource Upskilling Portal – Monolithic Architecture
* **Team Members**: Durga, Rakesh, Anshu, Jyothi, Rohan, Dileep
* **Stack**:
  + Backend: Spring Boot (Monolith)
  + DB: H2 / MySQL
  + RESTful Controllers
  + Repository: Single GitHub repo: [Mmind007/MCQ](https://github.com/Mmind007/MCQ)
* **Key Modules**:
  + Skill & Knowledgebase Management
  + Learning Journey & Logging
  + MCQ Testing Engine
  + User Registration & Self-Assessment

### **2. System Architecture Diagram (PPT Slide)**

* Single Application Block (Monolith)
* Layered Design:
  + Controller Layer
  + Service Layer
  + Repository Layer
  + DB
* Integrations: Swagger UI
* Tools: Lucidchart / Draw.io

### **3. Sample Test Data**

* Auto-loaded from data.sql or via Flyway/Liquibase
* Tables and Sample Entries:
  + skills, skill\_areas
  + learning\_material
  + users, learning\_logs, bookmarks
  + mcq\_questions, mcq\_options, mcq\_answers
  + test\_templates, tests, test\_results
* Sufficient records to demo search, test creation, test submission, and progress tracking

### **4. Entity Relationship Diagram (ERD)**

* Show main relationships:
  + Skills ↔ Areas ↔ Materials
  + User ↔ Bookmarks, Logs, Self-Assessments
  + Questions ↔ Options ↔ Tests ↔ Results
* Use dbdiagram.io or Draw.io

### **5. Process Flows / Sequence Diagrams**

* Core user journeys:
  + Skill and Area Creation → Material Tagging → User Browsing
  + User Registration → Profile Completion → Self-Assessment
  + Daily Learning Log → Timeline View
  + MCQ Test Generation (by area) → Attempt → Auto-Evaluation → Result
  + SME defines Test Template → System generates test =>User stories **US3.3** and **US3.4** in MCQ Testing epic

### **6. Codebase Organization**

* Monolithic Spring Boot project should include:
  + Controllers for each module (e.g., MCQController, SkillController)
  + Services handling business logic
  + JPA Entities and Repositories
  + application.yml for DB config
  + data.sql for demo
  + Unit and Integration tests
  + Swagger for API docs

### **7. GitHub Repository**

* **Repo**: <https://github.com/Mmind007/MCQ>
* Must include:
  + README.md
    - Overview
    - Setup instructions
    - DB setup & test data
    - REST API summary
    - Sample API request/response
  + Folder structure with src/main/java, resources, etc.
  + Postman Collection

### **8. Test Case Plan / Demo Scenarios**

* Test Cases or Scenarios for demo:
  + Admin: Create Skill, Area, Upload Materials
  + Resource: Register → Self-Assess → Log Progress
  + Take MCQ Test (based on Area) → View Result
  + Bookmark Content, View Timeline
  + SME: Define Test Template → System Generated Test
* Demonstrate testing of API endpoints on Swagger UI

### **9. Feature Summary (Excel or Word Table)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Story ID** | **Module** | **Priority** | **Status** | **Assigned** |
| US1.1 | Skill Mgmt | High | Completed | Durga |
| US2.1 | Learning Logs | High | In Progress | Rohan |
| US3.2 | MCQ Testing | High | Completed | Rakesh |
| US8.1 | Registration | High | Completed | Dileep |

### **10. Retrospective Notes (Optional Slide)**

* What worked well
* Challenges (e.g., test evaluation logic, relationships)
* Suggestions for scalability (e.g., future microservices split)

### **11. Final Submission Format (for Demo Only)**

upskill-capstone-demo/  
├── architecture-diagram.pptx  
├── process-flows/  
│ ├── mcq-test-sequence.png  
│ └── self-assessment-flow.png  
├── erd-diagram.png  
├── data.sql  
├── demo-plan.docx  
├── feature-summary.xlsx  
├── MCQ-repo-link.txt # GitHub link  
├── README.md  
└── Postman-Collection.json

### **12. Live Project Folder Structure (Inside Repo)**

MCQ/  
├── src/  
│ ├── main/java/com/upskillportal/  
│ ├── resources/  
│ │ ├── application.yml  
│ │ └── data.sql  
├── test/  
├── pom.xml  
├── README.md  
└── swagger-config/

\***\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***