

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**WORK INTEGRATED LEARNING PROGRAMMES**

**COURSE HANDOUT**

**Part A: Content Design**

|  |  |
| --- | --- |
| **Course Title** | **Scalable Services** |
| **Course No(s)** | **SE ZG583** |
| **Credit Units** | 5 |
| **Course Author** | AKANKSHA BHARADWAJ |
| **Version No** | 2.0 |
| **Date** | Jan 2021 |

**Course Objectives**

|  |  |
| --- | --- |
| **No** | **Course Objective** |
| **CO1** | Build competence to design, develop, implement and manage scalable information systems |
| **CO2** | Gain understanding of different techniques & tools for building and managing scalable services |
| **CO3** | Gain understanding of challenges and best practices in creating and managing scalable services |

**Text Book(s)**

|  |  |
| --- | --- |
| T1 | Microservices patterns by Chris Richardson, Manning Publications 2018 |
| T2 | Microservices in action by Morgan Bruce & Paulo Pereira, Manning Publications 2018 |
| T3 | Building Microservices by Sam Newman, O'Reilly Media 2015 |

**Reference Book(s) & other resources**

|  |  |
| --- | --- |
| R1 | https://kubernetes.io/docs/concepts/ |
| R2 | The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise, Second Edition by Michael T. Fisher; Martin L. Abbott Published by Addison-Wesley Professional, 2015 |
| R3 | Scalability patterns by Microsoft Azure: <https://docs.microsoft.com/en-us/azure/architecture/patterns/category/performance-scalability> |
| R4 | Scalability Patterns: Best Practices for Designing High Volume Websites by Chander Dhall |

**Modular Content Structure**

1. **Getting to know Scalability:** 
   1. Introduction to Performance, Consistency and availability
   2. What is scalability?
   3. Need for scalable architectures
   4. Principles of Scalability
   5. Guidelines for Building Highly Scalable Systems
   6. Architecturally scalable requirements
   7. Challenges for Scalability
   8. YouTube case Study
2. **Popular scaling approaches** 
   1. Managing & processing high volumes of data
      1. Partitioning and sharding
      2. Distributed data (CAP theorem, NoSQL, HDFS)
      3. Distributed Processing (Map reduce, Spark)
   2. Managing high velocity data streams (Kafka)
      1. Video streaming: Netflix, YouTube, use of CDN
      2. Real time analytics: Credit card fraud detection
      3. Web conferencing: WebEx, Zoom
      4. Edge computing: IoT systems
   3. Managing high volume transactions
      1. Service Replicas & load balancing
      2. Minimizing event processing: Command Query Responsibility Segregation (CQRS)
      3. Asynchronous communication
      4. Caching techniques: Distributed cache, global cache
   4. Scalability features in the Cloud (AWS, Azure, Google)
      1. Auto-scaling
      2. Horizontal and vertical scaling
      3. Use of Load balancers
      4. Virtualization
      5. Serverless computing
   5. Best Practices for Achieving Scalability
3. **Microservices -** Introduction
   1. Challenges with Monolith applications
   2. Microservices architecture
   3. Advantages and disadvantages of Microservices
   4. Process & organization for microservices
4. **Decomposition strategies**
   1. Decomposition by business capability
   2. Decomposition by business domain
   3. Decomposition guidelines
   4. Obstacles to decomposing an application into services
   5. Defining service APIs
5. **Communication between Microservices**
   1. Inter-service communication
      1. Synchronous communication (REST, gRPC)
      2. Asynchronous communication
   2. Application boundary
      1. API gateway
      2. API design
      3. Creating and versioning APIs
      4. API security
      5. Backends for frontends
6. **Transaction management**
   1. Distributed transactions
   2. Implementation
   3. Challenges
   4. Solutions
   5. Sagas
7. **Building with pipelines**
   1. Continuous integration
   2. Tooling
   3. Repository patterns – Multi-repo, mono-repo
8. **Designing reliable microservices** 
   1. Sources of failure, cascading failures
   2. Designing reliable communication: Retires, async. Comm., circuit breakers
   3. Maximizing service reliability: Load balancing, Rate limiting (Queues, Throttling)
   4. Service mesh
9. **Securing and Testing scalable services**
   1. Securing code and repositories
   2. Using Authentication and Authorization
   3. Unit testing
   4. Integration testing
   5. Load testing
10. **Deploying Microservices**
    1. Service startup
    2. Running multiple instances
    3. Adding load balancer
    4. Service to host models
       1. Single Service Instance to Host
       2. Multiple static Service Per Host
       3. Multiple scheduled services per host
    5. Deploying services without downtime: Canaries, Blue-Green, & rolling deploys
    6. Deploying microservices using Serverless deployment
11. **Deployment with Containers**
    1. Introduction to containers
    2. Containerizing a service
    3. Deploying to a cluster
12. **Monitoring**
    1. Golden signals
    2. Types of metrics
    3. Recommended practices
    4. Collecting metrics
    5. Instrumenting
    6. Raising sensible & actionable alerts
    7. Using logs & traces
    8. Useful info in log entries
    9. Tools for logging
    10. Logging the right information
    11. Tracing interaction between services
    12. Visualizing traces
13. **Kubernetes**
    1. Dockers for CaaS
    2. What is Kubernetes
    3. Deployment of Microservices using Kubernetes
    4. Scalability in Kubernetes
    5. Security in Kubernetes
    6. CI/CD using Kubernetes
    7. Kubernetes Dashboard

**Learning Outcomes:**

|  |  |
| --- | --- |
| **No** | **Learning Outcomes** |
| LO1 | Understanding of different scenarios where scaling is needed |
| LO2 | Understanding of different approaches to scaling |
| LO3 | Understanding of microservices technology |
| LO4 | Ability to design, develop and deploy microservices based applications |
| LO5 | Understanding of ways to monitor and manage Microservices |
| LO6 | Confident in using tools used in building scalable services |

**Part B: Contact Session Plan**

|  |  |
| --- | --- |
| **Academic Term** | First Semester 2023-2024 |
| **Course Title** | Scalable Services |
| **Course No** | SE ZG583 |
| **Lead Instructor** | AKANKSHA BHARADWAJ |

**Course Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Contact Session** | **List of Topic Title**  **(from content structure in Part A)** | **Topic #**  **(from content structure in Part A)** | **Text/Ref Book/external resource** |
| 1 | **Getting to know Scalability**   * Introduction to Performance, Consistency and availability * What is scalability? * Need for scalable architectures * Principles of Scalability * Guidelines for Building Highly Scalable Systems * Architecturally scalable requirements * Challenges for Scalability * YouTube case Study | 1 | R2 and R4 |
| 2 | **Popular scaling approaches**   * Managing & processing high volumes of data * Partitioning and sharding * Distributed data (CAP theorem, NoSQL, HDFS) * Distributed Processing (Map reduce, Spark) * Managing high velocity data streams (Kafka) * Video streaming: Netflix, YouTube, use of CDN * Real time analytics: Credit card fraud detection * Web conferencing: WebEx, Zoom * Edge computing: IoT systems | 2.1 and 2.2 | R2 |
| 3 | **Popular scaling approaches**   * Managing high volume transactions * Service Replicas & load balancing * Minimizing event processing: Command Query Responsibility Segregation (CQRS) * Asynchronous communication * Caching techniques: Distributed cache, global cache * Scalability features in the Cloud (AWS, Azure, Google) * Auto-scaling * Horizontal and vertical scaling * Use of Load balancers * Virtualization * Serverless computing * Best Practices for Achieving Scalability | 2.3, 2.4 and 2.5 | R2 |
| 4 | **Microservices -** Introduction   * Challenges with Monolith applications * Microservices architecture * Netflix case study | 3.1 and 3.2 | T1 |
| 5 | **Microservices -** Introduction   * Advantages and disadvantages of Microservices * Process & organization for Microservices   **Decomposition strategies**   * Decomposition by business capability * Decomposition by business domain | 3.3, 3.4, 4.1 and 4.2 | T1 |
| 6 | **Decomposition strategies**   * Decomposition guidelines * Obstacles to decomposing an application into services * Defining service APIs   **Communication between Microservices**   * Inter-service communication   + Synchronous communication (REST, gRPC)   + Asynchronous communication | 4.3, 4.4, 4.5 and 5.1 | T1 |
| 7 | **Communication between Microservices**   * Application boundary   + API gateway   + API design   + Creating and versioning APIs   + API security   + Backends for frontends   **Transaction management**   * Distributed transactions * Implementation * Challenges * Solutions * Sagas | 5.2 and 6 | T1 |
| 8 | Review and discussion |  |  |
| 9 | **Building with pipelines**   * Continuous integration * Tooling * Repository patterns – Multi-repo, mono-repo | 7 | T3 |
| 10 | **Designing reliable Microservices**   * Sources of failure, cascading failures * Designing reliable communication: Retires, async. Comm., circuit breakers * Maximizing service reliability: Load balancing, Rate limiting (Queues, Throttling) * Service mesh | 8 | T2 |
| 11 | **Securing and Testing scalable services**   * Securing code and repositories * Using Authentication and Authorization * Unit testing * Integration testing * Load testing | 9 | T1 and T3 |
| 12 | **Deploying microservices**   * Service startup * Running multiple instances * Adding load balancer * Service to host models   + Single Service Instance to Host   + Multiple static Service Per Host   + Multiple scheduled services per host | 10.1, 10.2, 10.3 and 10.4 | T2 |
| 13 | **Deploying microservices**   * Deploying services without downtime: Canaries, Blue-Green, & rolling deploys * Deploying microservices using Serverless deployment   **Deployment with Containers**   * Introduction to containers * Containerizing a service * Deploying to a cluster | 10.5, 10.6 and 11 | T2 |
| 14 | **Monitoring**   * Golden signals * Types of metrics * Recommended practices * Monitoring * Collecting metrics * Instrumenting * Raising sensible & actionable alerts * Using logs & traces * Useful info in log entries * Tools for logging * Logging the right information * Tracing interaction between services * Visualizing traces | 12 | T2 |
| 15 | **Kubernetes**   * Dockers for CaaS * What is Kubernetes * Deployment of Microservices using Kubernetes * Scalability in Kubernetes * Security in Kubernetes * CI/CD using Kubernetes * Kubernetes Dashboard | 13 | R1 and T3 |
| 16 | Review and demonstration of Microservices based application |  |  |

**Case studies:**

1. YouTube
2. Netflix
3. Amazon Prime
4. Google
5. Facebook

**Labs:**

1. Design and develop a Microservices based application
2. Attach a database to a service and perform basic CRUD operations
3. Exploring the communication between services by using shared database pattern
4. Exploring the communication using RabbitMQ
5. Exploring different deployment options for microservices (to be done in session)
6. Configuring Minikube and running a local cluster (to be done in session)
7. Deploying application on Minikube (to be done in session)
8. Exploring the Minikube dashboard (to be done in session)

**Assignment: 20 marks**

A group of a maximum of 4 students will be created and do they will be doing the following as a part of lab and assignment

1. Create a Microservices based application with at least 3 microservices. Each service should be maintained as a separate code repository so that

it can be developed, deployed, and tested independently.

2. Use a suitable database and database related pattern for these services

3. Use a suitable approach for the communication between these services (avoid high coupling between these services)

Exploring lab tools

4. Deploy all services on a single docker container

5. Deploy each service on separate docker containers

6. Run a minikube cluster on your local machine and explore various options in this. Try deployment of your application on this.

7. Create a Kubernetes cluster on the cloud and deploy your application on this cluster. Analyze your cluster on the Kubernetes dashboard (optional)

8. Explore scaling, recovery, security, and CI/CD in Kubernetes (optional)

Submission details:

a) Submit elaborate documentation with group details, contribution by each group member, brief application description, the architecture of

your application, steps, and screenshot for each of the above-mentioned tasks. Also, provide a link to the GitHub repositories in the document.

b) Create the first demo video to explain your microservices, the database, and communication between the services (for points 1,2, and 3)

c) Create a second demo video to show various deployment options (for points 4 and 5)

d) Create the third video to show a demo about the minikube task and the Kubernetes cluster on the cloud (for points 6,7, and 8)

All of this has to be submitted in a single zip file with the file name as <Group member Bits IDS>\_<application name>. Each member of the team has to upload the document

In case, you copy your assignment from the internet or other people in the other groups then no marks will be awarded

**Evaluation Scheme**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Evaluation Component** | **Name**  (Quiz, Lab, Project, Midterm exam, End semester exam, etc) | **Type**  (Open book, Closed book, Online, etc.) | **Weight** | **Duration** | **Day, Date, Session, Time** |
| **EC – 1** | Quiz 1 |  | 5% |  | September 1-10, 2023 |
|  | Quiz 2 |  | 5% |  | October 1-10, 2023 |
|  | Lab (exploring different tools) |  | 10% |  | To be announced |
|  | Assignment (end to end app development) |  | 10% |  | To be announced |
| **EC – 2** | Mid-term Exam | Closed book | 30% | 2 hours | Saturday, 23/09/2023 (FN) |
| **EC – 3** | End Semester Exam | Open book | 40% | 2 ½ hours | Saturday, 25/11/2023 (FN) |

***Note*** *- Evaluation components can be tailored depending on the proposed model.*

**Important Information**

Syllabus for Mid-Semester Test (Closed Book): SL No. 1 to 8

Syllabus for Comprehensive Exam (Open Book): All topics given in plan of study

Evaluation Guidelines:

1. EC-1 consists of either two Assignments or three Quizzes. Announcements regarding the same will be made in a timely manner.
2. For Closed Book tests: No books or reference material of any kind will be permitted. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
3. For Open Book exams: Use of prescribed and reference text books, in original (not photocopies) is permitted. Class notes/slides as reference material in filed or bound form is permitted. However, loose sheets of paper will not be allowed. Use of calculators is permitted in all exams. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
4. If a student is unable to appear for the Regular Test/Exam due to genuine exigencies, the student should follow the procedure to apply for the Make-Up Test/Exam. The genuineness of the reason for absence in the Regular Exam shall be assessed prior to giving permission to appear for the Make-up Exam. Make-Up Test/Exam will be conducted only at selected exam centers on the dates to be announced later.

It shall be the responsibility of the individual student to be regular in maintaining the self-study schedule as given in the course handout, attend the lectures, and take all the prescribed evaluation components such as Assignment/Quiz, Mid-Semester Test and Comprehensive Exam according to the evaluation scheme provided in the handout.</DIV>