



PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage
Bengaluru 560085

Department of Computer Science and Engineering
B. Tech. CSE - 6th Semester
Jan – May 2023

UE20CS352: OBJECT ORIENTED ANALYSIS AND
DESIGN USING JAVA

Project Report

Course-craft: Online Learning Platform

Submitted by:

Sathvik A PES1UG20CS493
Vrushank G PES1UG20CS516
Srinivas Y PES1UG20CS517
Akash S PES1UG20CS534

Class of *Prof. Bharghavi M*

Table of Contents

Objective of the project:.....	4
Synopsis of the project:.....	4
Features of the Projects:.....	4
User Management:	4
Course Management:	4
Content Management:	5
UML models:	5
Use case Diagram:	5
Use Case description.....	6
1. Register User.....	6
2. Login User	6
3. Create Course.....	6
4. Enroll in Course	6
5. Edit Profile	7
Class Diagram:.....	7
State Diagram:	8
Activity Diagram:	9
ARCHITECTURE PATTERN	10
DESIGN PRINCIPLES AND PATTERNS.....	12
Design Principles	12
1. Dependency Injection:	12
2. Open-Closed Principle:.....	12
3. Separation of Concerns:.....	12
4. Single Responsibility Principle:.....	12
Design Patterns	13
1. Singleton Creational Pattern:	13
2. Proxy Design Pattern:	13
3. Iterator Pattern:	13
Individual Contributions	13
Screenshots:	14
Creating new student.....	14
Student Login.....	14
After Login.....	14
View All Courses	15
Enroll Course	16
After Enrollment	16

Teacher Register.....	17
Teacher Login	17
After Login.....	17
Create new course	18
Add Topic.....	19

Objective of the project:

Our objective is to create an innovative online learning platform that brings teachers and students. With user-friendly course creation, authentication and registration features, and an intuitive interface that promotes engagement and interactivity, our platform aims to revolutionize the way people learn and teach online. Our goal is to empower students to acquire new skills and knowledge, and enable teachers to share their expertise with a global audience, all while providing a seamless and enjoyable online learning experience.

Synopsis of the project:

Our project is an online learning platform that connects teachers and students in a dynamic virtual classroom environment.

- The platform has two types of users: students and teachers.
- Students can browse available courses, enrol in courses, and access course content.
- Teachers can create courses, which consists of list of topics.
- The topics can be videos or self-learning notes.
- Users must register and authenticate before accessing the website.

Features of the Projects:

User Management:

- **User registration:** Users can create accounts on the platform by providing basic personal information, such as their name and email address.
- **User authentication:** Users must enter their login credentials (username/email and password) to access the platform.
- **User profile management:** Users can view and edit their personal information, such as their name, profile picture, and contact information.

Course Management:

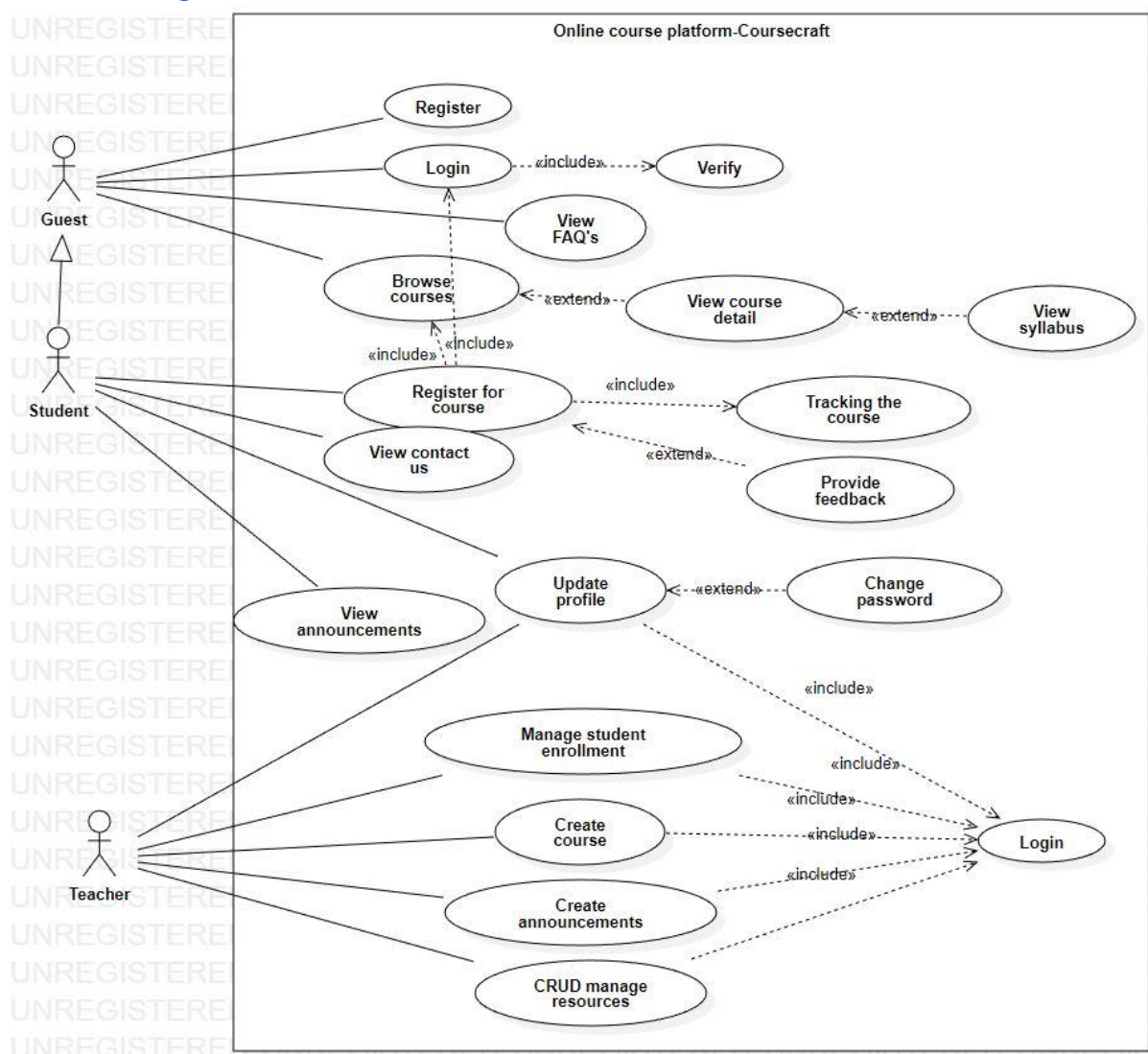
- **Course creation by teachers:** Teachers can create new courses by adding course information (e.g., title, description, price, category) and selecting the topics and course content they want to include.
- **Course enrollment by students:** Students can browse available courses and enroll in the ones they're interested in.
- **Course content management for teachers:** Teachers can edit course information and manage course content, such as adding or removing topics and updating course materials.

Content Management:

- **Topic creation by teachers:** Teachers can create topics within a course and add topic information (e.g., title, description, content type).
- **PDF and video uploading and management:** Teachers can upload PDF or video content to a topic for students to access and download.
- **Topic content preview for students:** Students can preview the content of a topic before enrolling in a course.

UML models:

Use case Diagram:



Use Case description:

1. Register User

- **Actors:** Student, Teacher
- **Pre-conditions:** User is not registered yet
- **Post-conditions:** User account is created
- **Basic Flow:**
 1. User navigates to the registration page
 2. User enters their personal information (name, email, password, etc.)
 3. User submits the registration form
 4. System creates a new user account and redirects the user to the login page

2. Login User

- **Actors:** Student, Teacher
- **Pre-conditions:** User is not logged in
- **Post-conditions:** User is logged in and can access their account
- **Basic Flow:**
 1. User navigates to the login page
 2. User enters their login credentials (email and password)
 3. User submits the login form
 4. System verifies the user's credentials and redirects the user to their account dashboard

3. Create Course

- **Actors:** Teacher
- **Pre-conditions:** Teacher is logged in
- **Post-conditions:** Course is created and published on the platform
- **Basic Flow:**
 1. Teacher navigates to the "Create Course" page
 2. Teacher enters course information (title, description, price, category, etc.)
 3. Teacher adds course topics and content (PDFs, videos, etc.)
 4. Teacher previews and reviews the course content
 5. Teacher publishes the course on the platform

4. Enroll in Course

- **Actors:** Student
- **Pre-conditions:** Student is logged in
- **Post-conditions:** Student is enrolled in the course and can access course content
- **Basic Flow:**
 1. Student navigates to the course page
 2. Student clicks the "Enroll" button
 3. Student is prompted to confirm their enrollment and payment (if applicable)

4. System verifies payment (if applicable) and enrolls the student in the course
5. Student can access the course content from their account dashboard

5. Edit Profile

- **Actors:** Student, Teacher

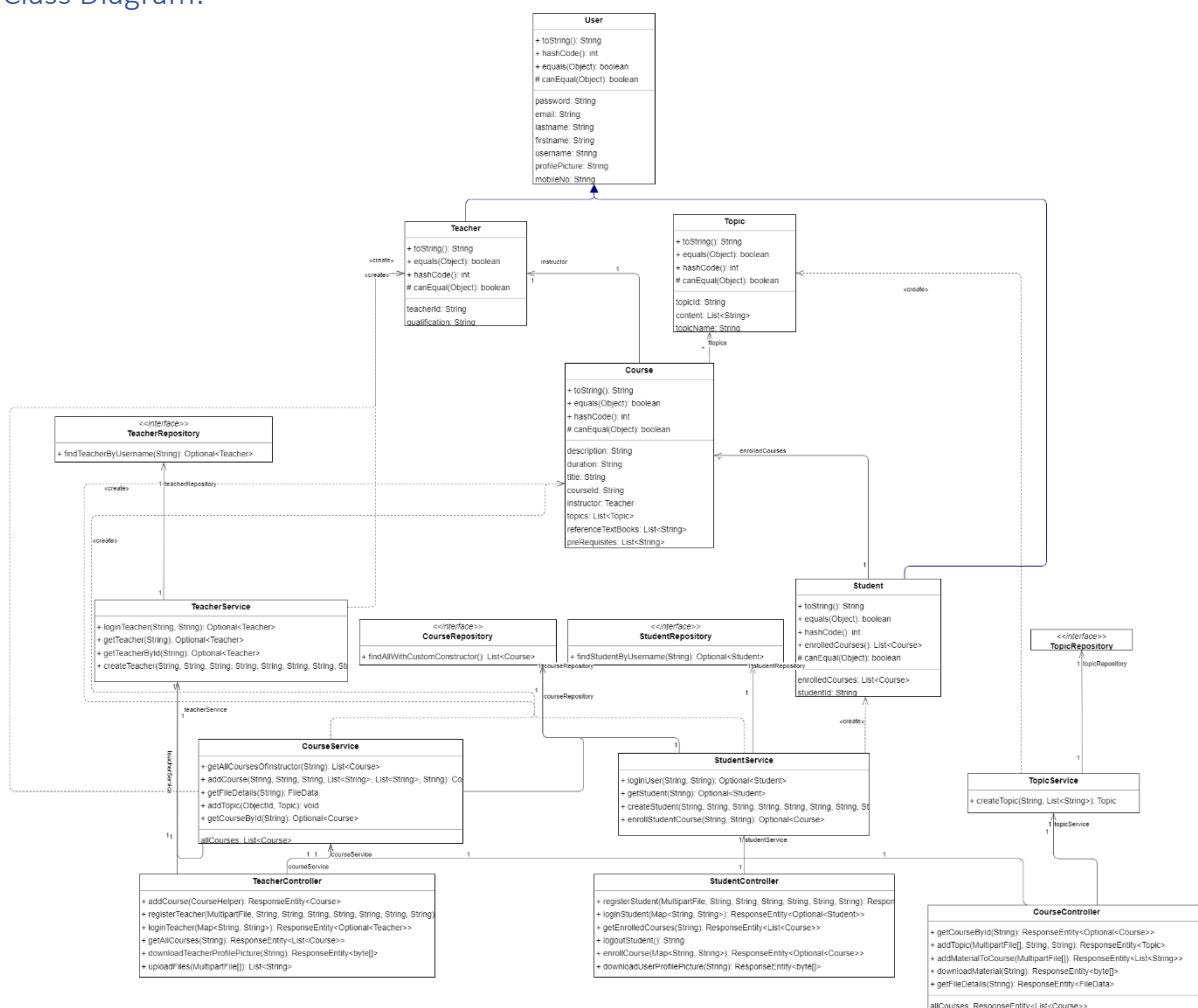
- **Pre-conditions:** User is logged in

- **Post-conditions:** User's profile information is updated

- **Basic Flow:**

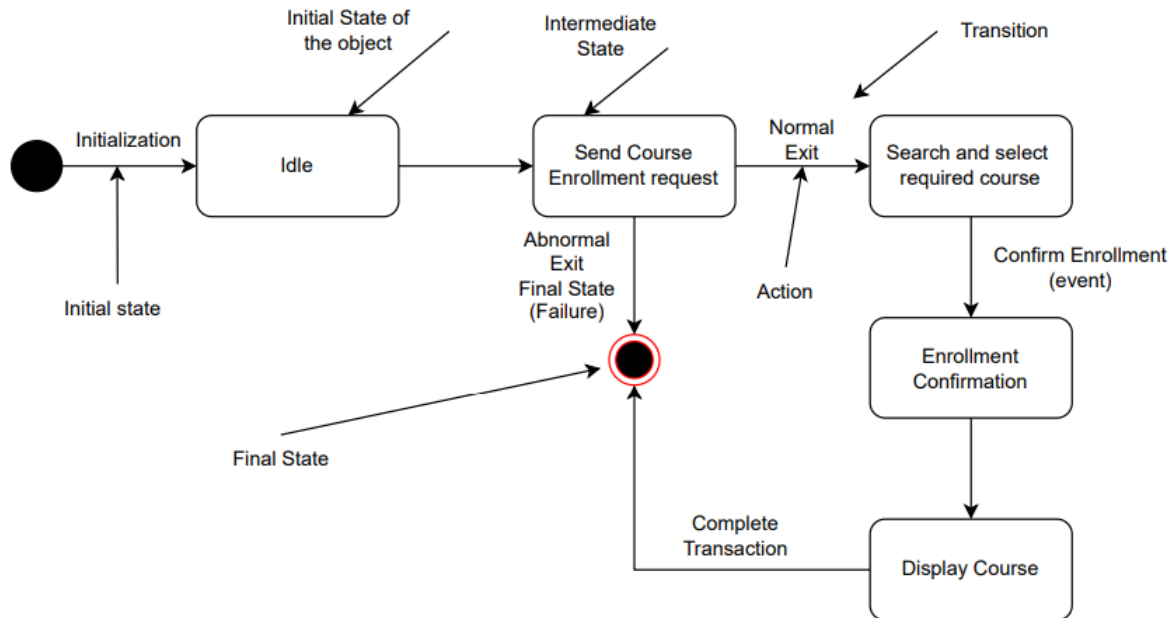
1. User navigates to their profile page
2. User clicks the "Edit Profile" button
3. User updates their personal information (name, profile picture, contact information, etc.)
4. User submits the updated information
5. System saves the updated information and redirects the user to their profile page

Class Diagram:

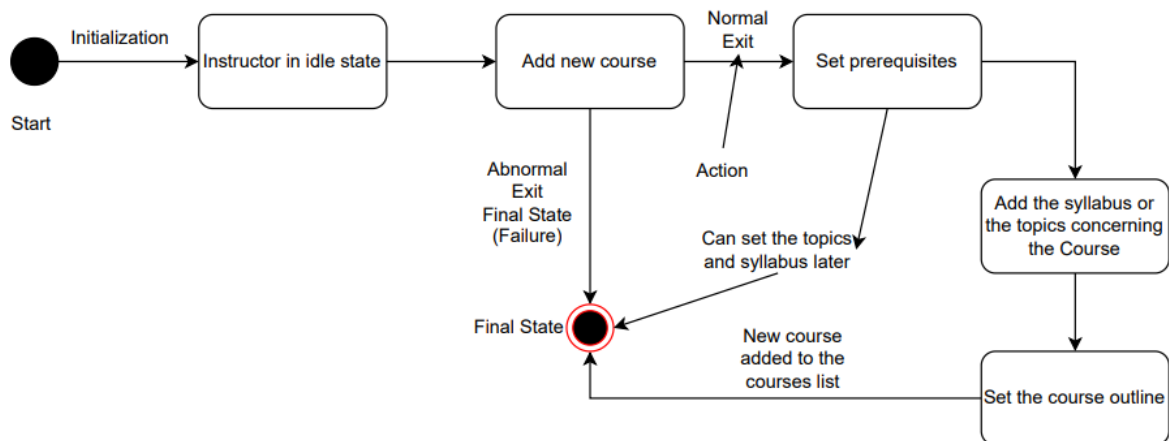


State Diagram:

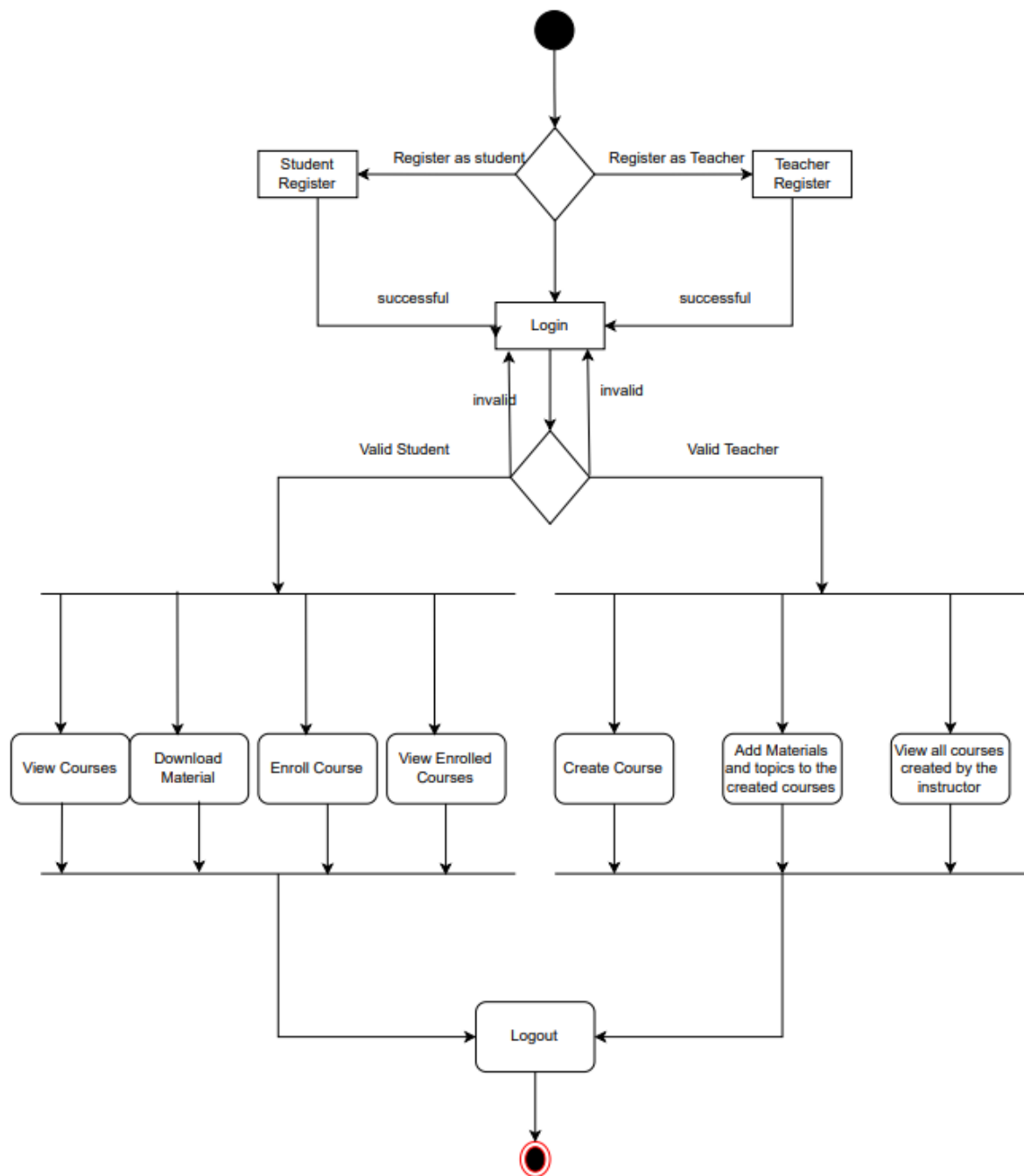
State diagram for Course Enrollment



State diagram for Add Course



Activity Diagram:



ARCHITECTURE PATTERN

In Object-Oriented Programming (OOP), design architecture refers to the overall structure of an application or system. OOP design architecture defines how the various components of an application will interact with each other, and how they will be organized to achieve specific goals.

Model-View-Controller (MVC) Architecture pattern is implemented in our Application.

Model: The Model consists of the classes that represent the data and business logic of the application. In this project, the Model package contains several classes that represent various entities and concepts related to the platform, such as

- Student: represents a student account, including fields for name, email, password, and any other relevant student data
- Teacher: represents a teacher account, including fields for name, email, password, and any other relevant teacher data
- Course: represents a course on the platform, including fields for title, description, price, category, and any other relevant course data
- Topic: represents a topic within a course, including fields for title, description, and content (PDFs, videos, etc.)

View:

The View is responsible for presenting the data to the user and handling user input. In this project, the View package is not explicitly shown, but it could contain classes that Represent the following:

- Registration form: allows students and teachers to create a new account on the platform
- Login form: allows students and teachers to log in to their account
- Student dashboard: displays a student's account information, enrolled courses, and provides navigation to other pages
- Teacher dashboard: displays a teacher's account information, courses created by the teacher, and provides navigation to other pages
- Course page: displays information about a course and allows students to enroll (if applicable) and access course content
- Topic page: displays content for a specific topic within a course

Controller:

Controller: The Controller acts as an intermediary between the Model and the View, handling user input, updating the Model, and notifying the View of any changes. In this project, the Controller package contains several classes that represent the controllers of various functionalities of the platform, such as

- StudentController: handles student-related actions, such as registration, login, and profile editing
- TeacherController: handles teacher-related actions, such as course creation, editing, and deletion
- CourseController: handles course-related actions, such as course viewing, enrollment management, and content access

- TopicController: handles topic-related actions, such as adding, editing, and deleting topics within a course
- EnrollmentController: handles enrollment-related actions, such as enrolling a student in a course, displaying a student's enrolled courses, and unenrolling a student from a course

the MVC pattern provides a clear separation of concerns between the Model, View, and Controller components of the application, enabling better organization, modularity, and maintainability of the code.

DESIGN PRINCIPLES AND PATTERNS

Design Principles

Design principles in OOP are a set of guidelines that help developers create code that is easy to maintain, flexible, and efficient. These principles help to organize and structure code in a way that makes it easier to understand and modify, and they promote code reusability. Following are some of the design principles used in our project.

1. Dependency Injection:

Dependency injection is a design pattern that enables loose coupling between different components in a system, by allowing dependencies to be injected rather than hardcoded. In the context of the online learning platform, this could be achieved by using a dependency injection framework like Spring to manage the creation and injection of objects, such as controllers, services, and repositories. For example, instead of having a controller create a new instance of a service object, the service object can be injected into the controller as a dependency. This promotes modularity and makes it easier to replace or swap out dependencies in the future.

2. Open-Closed Principle:

The Open-Closed Principle (OCP) states that software entities (classes, modules, etc.) should be open for extension but closed for modification. This means that adding new features or behavior to the system should be achieved through the addition of new code rather than modifying existing code. In the context of the online learning platform, this could be achieved by using interfaces and abstract classes to define common behavior and create concrete implementations for each type of user. For example, a User interface could be created and extended by the Student and Teacher classes, allowing for new types of users to be added without modifying existing code.

3. Separation of Concerns:

The Separation of Concerns principle aims to keep different aspects of a system separate, with each component responsible for a single concern. This helps to promote modularity and makes it easier to maintain and extend the system. In the context of the online learning platform, this could be achieved by separating the user interface, business logic, and data storage into separate components. For example, a controller could be responsible for handling requests and responses from the user interface, a service layer could handle the business logic and rules, and a repository layer could handle the data storage and retrieval.

4. Single Responsibility Principle:

The Single Responsibility Principle (SRP) states that a class should have only one reason to change, meaning that it should have only one responsibility or concern. This helps to keep code focused and easier to understand and modify. In the context of the online learning platform, each class and component should have a single responsibility, such as handling user authentication, managing course enrollments, or providing access to course content. For example, a UserController should be responsible for handling user-related actions, such as registration, login, and profile editing, rather than trying to handle course-related actions as well.

Design Patterns

Design patterns are typical solutions to common problems in software design. Each pattern is like a blueprint that you can customize to solve a particular design problem in your code. They define a common language that helps your team communicate more efficiently.

1. Singleton Creational Pattern:

The Singleton pattern is a creational design pattern that restricts the instantiation of a class to a single instance and provides global access to that instance. In the context of the online learning platform, this pattern could be used for creating a single instance of a resource-intensive class, such as a **database connection pool** or a caching mechanism, that needs to be shared across the system. For example, a Singleton pattern could be used to ensure that there is only one instance of the **CourseService class**, which manages all the course-related operations, such as creating new courses, adding topics, and enrolling students.

2. Proxy Design Pattern:

The Proxy pattern is a structural design pattern that provides a surrogate or placeholder object to control access to another object. In the context of the online learning platform, this pattern could be used to provide an additional layer of security or access control over certain operations or resources. For example, a proxy object could be used to restrict access to certain courses or topics based on the user's subscription level or permission level. The proxy object could check the user's credentials and then delegate the request to the actual object only if the user is authorized.

3. Iterator Pattern:

The Iterator pattern is a behavioral design pattern that provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation. In the context of the online learning platform, this pattern could be used to iterate over a collection of course topics, assignments, or quizzes, and provide a uniform way of accessing them without exposing the underlying data structures. For example, an iterator pattern could be used to provide a consistent way of iterating over the topics in a course, regardless of whether they are stored in a list, array, or database. This would make it easier to modify the storage mechanism in the future without affecting the rest of the system.

Individual Contributions

Sathvik: Teacher

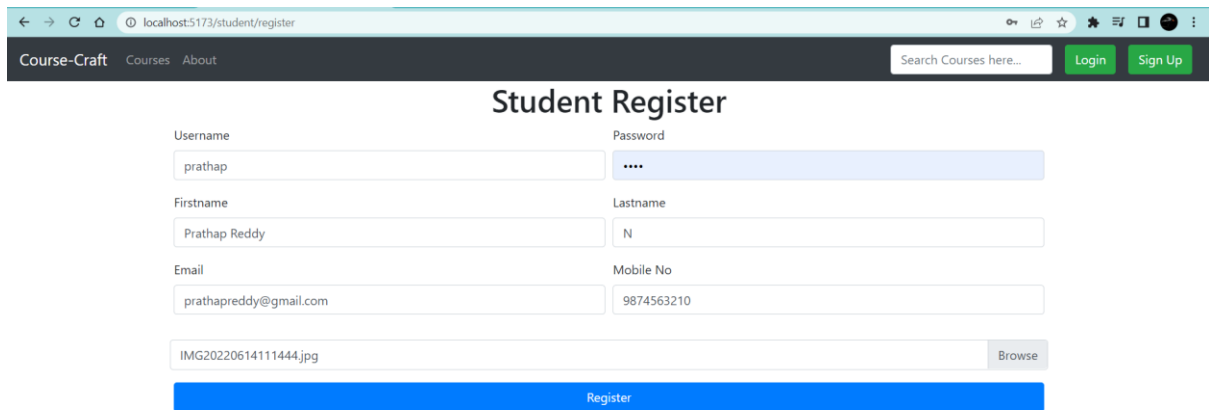
Vrushank: Student

Akash and Srinivas: Course

Frontend was divided equally.

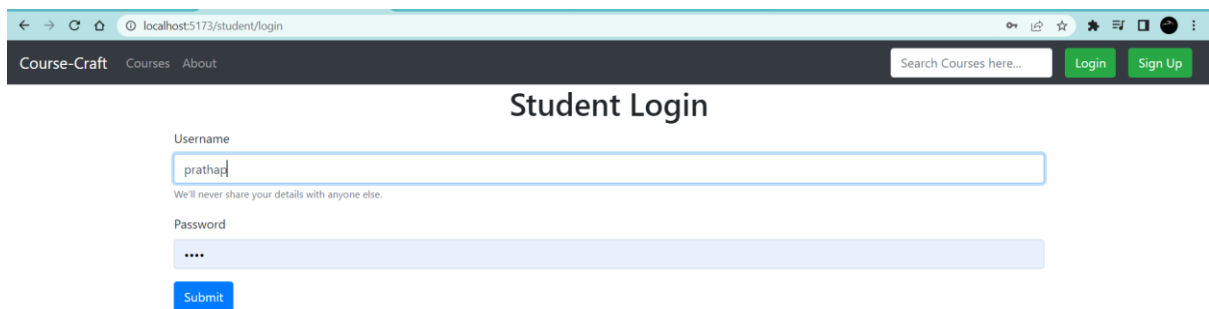
Screenshots:

Creating new student



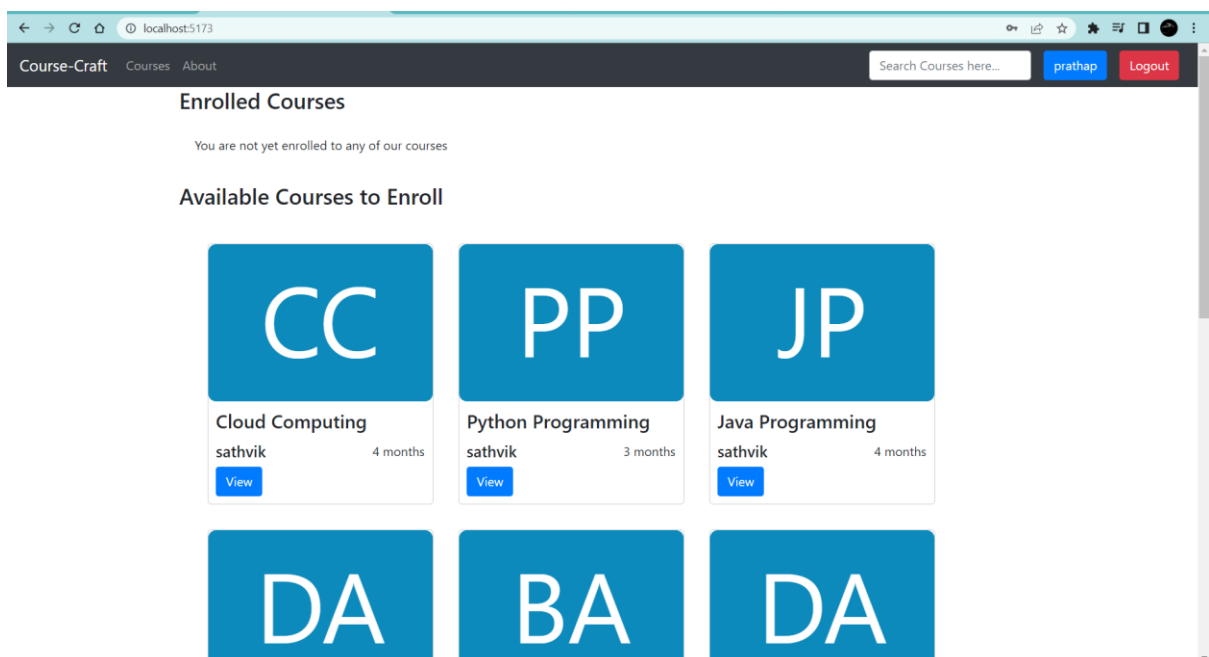
The screenshot shows a web browser at localhost:5173/student/register. The page has a dark header with 'Course-Craft' and navigation links 'Courses' and 'About'. A search bar and 'Login'/'Sign Up' buttons are on the right. The main content is titled 'Student Register' and contains a form with the following fields: Username (prathap), Password (masked with dots), Firstname (Prathap Reddy), Lastname (N), Email (prathapreddy@gmail.com), and Mobile No (9874563210). There is a file upload field for a profile picture with a 'Browse' button. A large blue 'Register' button is at the bottom.

Student Login



The screenshot shows a web browser at localhost:5173/student/login. The page has the same dark header as the register page. The main content is titled 'Student Login' and contains a form with 'Username' (prathap) and 'Password' (masked with dots) fields. A small text note says 'We'll never share your details with anyone else.' Below the password field is a blue 'Submit' button.

After Login



The screenshot shows a web browser at localhost:5173. The page has a dark header with 'Course-Craft', navigation links, and a search bar. The user 'prathap' is logged in, as indicated by the 'prathap' and 'Logout' buttons. The main content is titled 'Enrolled Courses' and shows a message: 'You are not yet enrolled to any of our courses'. Below this is a section titled 'Available Courses to Enroll' with six course cards arranged in two rows of three. Each card has a blue header with initials, a title, the instructor 'sathvik', and the duration. Each card also has a blue 'View' button.

Course Initials	Course Title	Instructor	Duration
CC	Cloud Computing	sathvik	4 months
PP	Python Programming	sathvik	3 months
JP	Java Programming	sathvik	4 months
DA			
BA			
DA			

View All Courses

The screenshot displays a web application interface for viewing all courses. The browser address bar shows the URL `localhost:5173/courses`. The application header includes the text "Course-Craft" and navigation links "Courses" and "About". A search bar with the placeholder "Search Courses here..." is present, along with user controls for "prathap" and "Logout".

The main content area features a grid of six course cards, each with a blue header, a title, the instructor's name "sathvik", the duration, and a "View" button:


- CC** Cloud Computing, sathvik, 4 months
- PP** Python Programming, sathvik, 3 months
- JP** Java Programming, sathvik, 4 months
- DA** Data Structures and Algorithms, sathvik, 5 months
- BA** Big Data Analytics, sathvik, 6 months
- DA** Data Analytics, sathvik, 3 months

Enroll Course

← → ↻ 🏠 📄 ⭐ ⚙️ 🔍 🌙 🌐 🖨️ ⌵

Course-Craft Courses About

Search Courses here... prathap Logout



Cloud Computing

Instructor: Sathvik A

Description:

Advanced Certification in DevOps & Cloud Computing from IIT Roorkee E&ICT & Intellipaat. Learn from IIT Professors & Industry Practitioners. Advance Your Career Now!

Topics

1. Introduction
2. DevOps
3. Containerization
4. Aws

Pre-requisites

Basic computer skills Knowledge of networking

Knowledge of networking Familiarity with Linux

Reference Textbooks

Cloud Computing Cloud Native


Enroll

After Enrollment

← → ↻ 🏠 📄 ⭐ ⚙️ 🔍 🌙 🌐 🖨️ ⌵

Course-Craft Courses About

Search Courses here... prathap Logout



Cloud Computing

Instructor: Sathvik A

Description:

Advanced Certification in DevOps & Cloud Computing from IIT Roorkee E&ICT & Intellipaat. Learn from IIT Professors & Industry Practitioners. Advance Your Career Now!

Topics

1. Introduction
2. DevOps

Pre-requisites

Basic computer skills Knowledge of networking

Knowledge of networking Familiarity with Linux

Reference Textbooks

Cloud Computing Cloud Native

Enrolled

1 Introduction

1 Introduction to Cloud Computing and Parallel Computing.pdf

Size: 0.64 MB Download

2 Grid & Cloud Computing.pdf

Size: 0.72 MB Download

3 Cloud Models.pdf

Size: 0.51 MB Download

2. DevOps

1 Introduction to Cloud Computing and Parallel Computing.pdf

Size: 0.64 MB Download

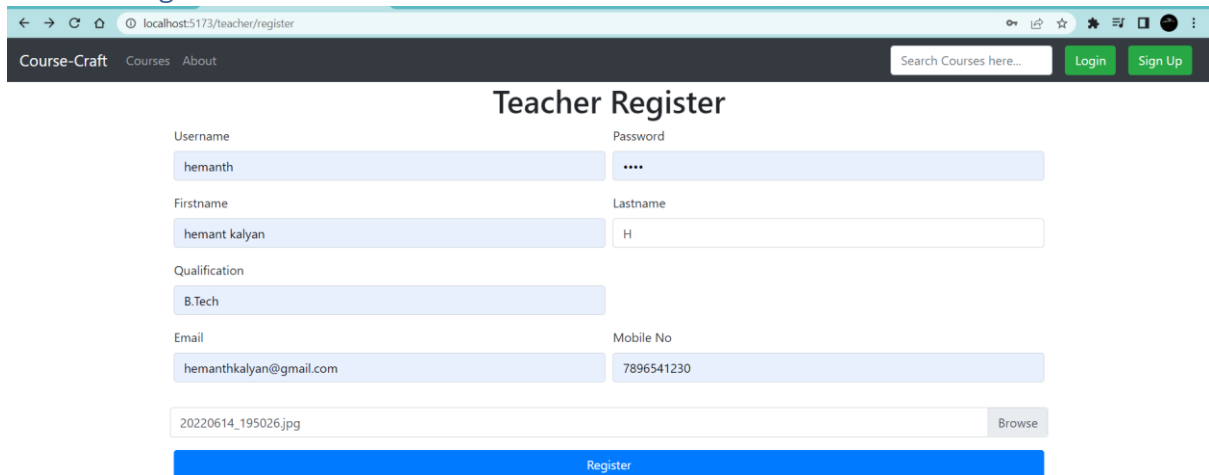
2 Grid & Cloud Computing.pdf

Size: 0.72 MB Download

3 Cloud Models.pdf

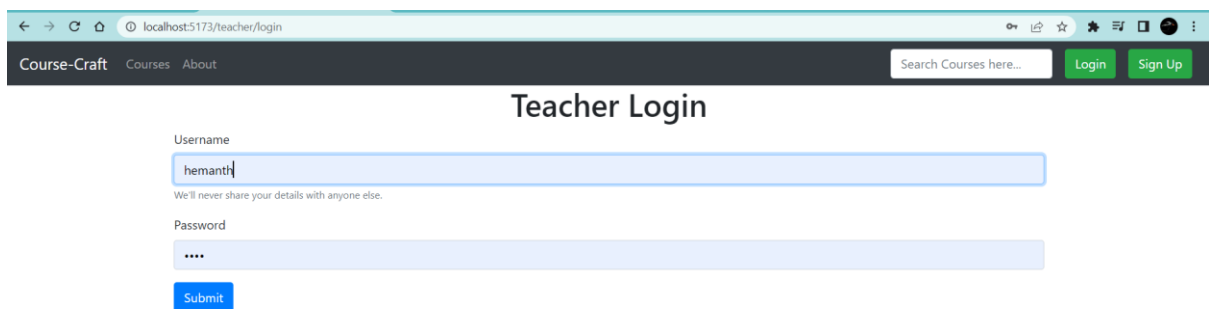
Size: 0.51 MB Download

Teacher Register



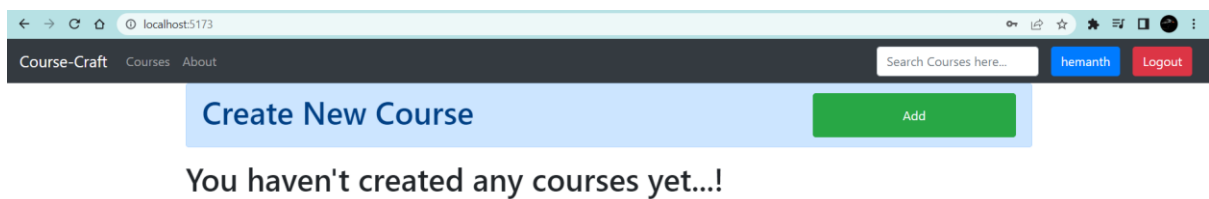
The screenshot shows a web browser at localhost:5173/teacher/register. The page has a dark header with 'Course-Craft' and links to 'Courses' and 'About'. A search bar and 'Login'/'Sign Up' buttons are on the right. The main form is titled 'Teacher Register' and contains the following fields: Username (hemanth), Password (masked with dots), Firstname (hemant kalyan), Lastname (H), Qualification (B.Tech), Email (hemanthkalyan@gmail.com), and Mobile No (7896541230). There is a file upload field for a profile picture (20220614_195026.jpg) with a 'Browse' button. A large blue 'Register' button is at the bottom.

Teacher Login



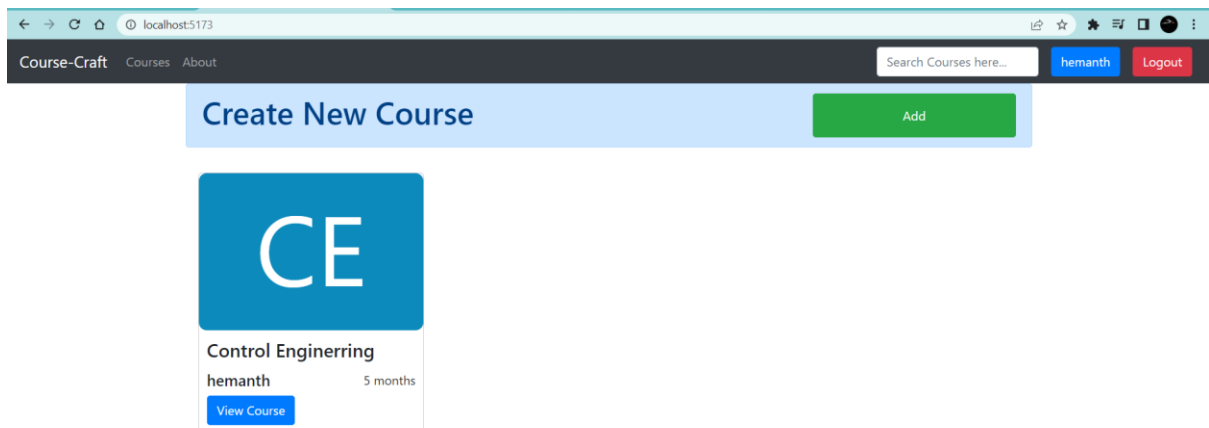
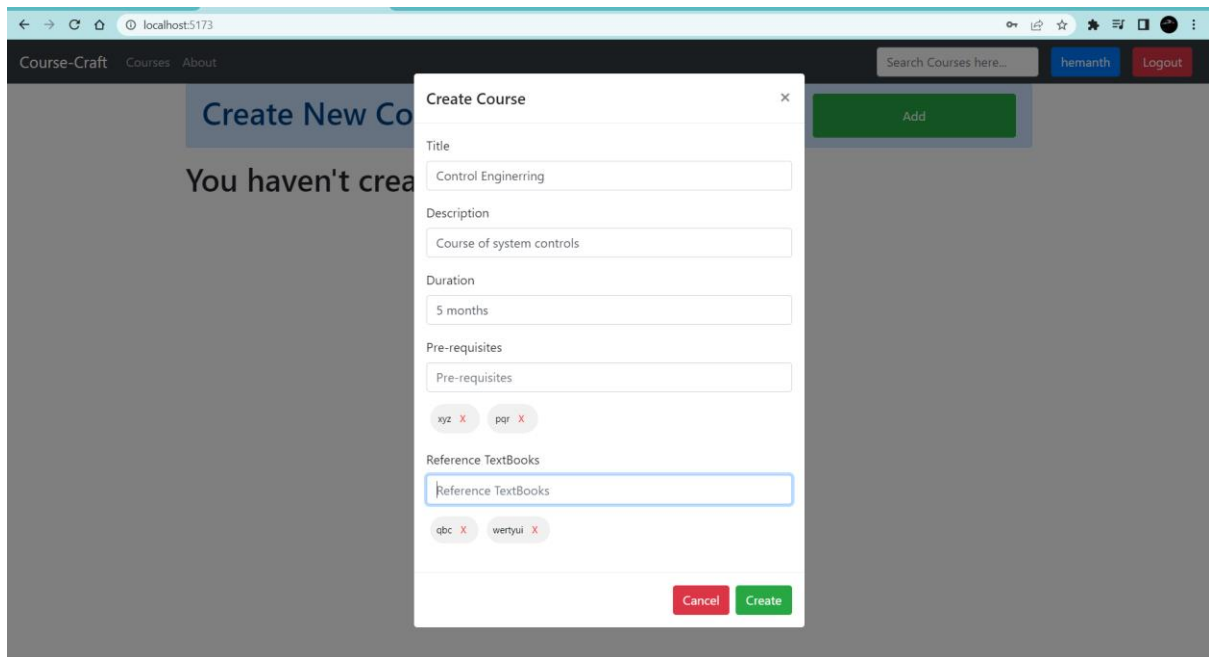
The screenshot shows a web browser at localhost:5173/teacher/login. The header is identical to the register page. The main form is titled 'Teacher Login' and contains: Username (hemanth), Password (masked with dots), and a 'Submit' button. A small text note below the username field reads: 'We'll never share your details with anyone else.'

After Login



The screenshot shows the user interface after login. The header now includes the username 'hemanth' and a 'Logout' button. A large blue banner with the text 'Create New Course' and a green 'Add' button is displayed. Below the banner, the text 'You haven't created any courses yet...!' is shown.

Create new course



Add Topic

