Applications Development

Course Name: Applications Development

Course Code: IST 3108

Course Level: Year 3 Sem 1

Contact Hours: 60

Credit Units: 4 CUs

Lecture Hours:

- Tuesday, 11:00 Am to 13:00 PM LLT 3A
- Friday, 14:00 PM to 16:00 PM
- Wednesday, 17:00 PM to 21:00 PM

Lecturer: Dr. Peter Wakholi (Ph.D)

Pre-requisite Courses:

- Structured Programming
- Data and Information Management
- System Analysis and Design
- Object-Oriented Programming

Overall Course Objectives

1. **Develop Practical Software Development Skills:** Equip students with the ability to design, develop, and deploy full-stack web applications, applying advanced programming concepts and modern development frameworks.
2. **Understand and Apply Agile Methodologies:** Teach students how to effectively use Agile methodologies in software development, enabling them to manage projects iteratively and respond flexibly to changes.
3. **Master the MVC Architecture:** Provide students with in-depth knowledge of the Model-View-Controller (MVC) architecture and how it can be used to structure and organize web applications.
4. **Implement Secure and Scalable Solutions:** Ensure students understand and can apply best practices in application security and performance optimization, focusing on building secure, efficient, and scalable web applications.
5. **Integrate Front-End and Back-End Technologies:** Enable students to seamlessly integrate front-end and back-end components of a web application, ensuring consistent and reliable communication between different layers.

6. **Deploy and Manage Applications:** Train students in deployment techniques using modern DevOps practices, including Continuous Integration/Continuous Deployment (CI/CD) pipelines and containerization technologies.
7. **Collaborate Effectively in Teams:** Foster teamwork and collaboration, encouraging students to work together in diverse groups to deliver a comprehensive web application as their capstone project.

Expected Outcomes

By the end of the course, students will be able to:

1. **Design and Develop a Full-Stack Web Application:** Independently or in teams, students can design and develop a full-stack web application that addresses a real-world problem using the MVC architecture.
2. **Apply Advanced Programming Techniques:** Demonstrate the ability to develop software applications using advanced programming concepts such as design patterns, SOLID principles, and algorithmic problem-solving.
3. **Manage Software Development Projects:** Use Agile methodologies to plan, manage, and deliver software projects, including creating user stories, sprints, and iterative development cycles.
4. **Ensure Application Security and Performance:** Identify and implement security best practices and performance optimization techniques, conducting security audits and performance tests to ensure application robustness.
5. **Deploy Applications Using Modern DevOps Practices:** Successfully deploy a web application to a cloud environment using CI/CD pipelines and containerization technologies, ensuring the application is scalable, reliable, and maintainable.
6. **Collaborate and Communicate Effectively:** Work effectively in a team environment, demonstrating strong collaboration and communication skills throughout the software development lifecycle.
7. **Prepare Comprehensive Documentation:** Produce detailed technical documentation, including system design documents, user manuals, and final project reports, ensuring all aspects of the development process are well-documented and accessible.

Course Description

This course focuses on developing real-life applications, emphasizing web application development. Students will learn advanced programming concepts, full-stack development, database integration, application security, and deployment. The course combines theoretical knowledge with practical, hands-on experience to prepare students for real-world software development challenges.

Course Content:

Week	**Module**	**Topics**	**Reference Materials**	**Self-Study Topics**
1	Introduction to Application Development	Overview of SDLC, Agile methodologies, Setting up the development environment	• "Agile Software Development" by Robert C. Martin • Online tutorials on Git and GitHub	Research on different SDLC models
	Discussion by Group 1	• Discuss the challenges and best practices in setting up a collaborative development environment using GitLab and IDEs. • Explore common issues faced during setup and how to troubleshoot them.		
2	Capstone Project	Project planning and management, Agile methodologies, Development and presentation	• "Scrum: The Art of Doing Twice the Work in Half the Time" by Jeff Sutherland • Online resources on project management	Working on a team project to develop a real-life application
	Discussion by Groups 2	• Analyze your project plan, focusing on risk management and timeline feasibility. • Discuss how to create effective user stories that align with your project's goals and ensure that all critical functionalities are captured. • Demonstrate using Agile project management tools like JIRA, Asana, and Azure Boards in managing your project.		
3	Advanced Programming Concepts	Advanced OOP, Design patterns, SOLID principles, Advanced data structures, Algorithmic problem-solving	• "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma et al. • "Introduction to Algorithms" by Cormen et al.	Implementing design patterns in a chosen language
	Discussion by Group 3	• Identify and justify the design patterns that will be most beneficial for your capstone project. • Discuss how these patterns can help manage complexity and improve code reusability in your application.		

4	Web Development Fundamentals	HTML5, CSS3, JavaScript ES6+, Responsive design, Server-side programming, RESTful APIs	• "Eloquent JavaScript" by Marijn Haverbeke • "Learning Web Design" by Jennifer Robbins	Building a simple responsive website
	Discussion by Group 4	• Present the UI/UX design principles you've applied to your capstone project. • Discuss how you plan to implement responsive design and ensure that the user interface meets the needs of your target audience.		
5	Full-Stack Development with MVC Architecture	Understanding MVC, Implementing MVC in a web framework, Integrating front-end and back-end, User authentication and authorization	• "Pro MERN Stack" by Vasan Subramanian • Documentation of chosen web framework (e.g., Express.js, Django)	Creating a basic full-stack application
	Discussion by Group 5	• Discuss the MVC framework you've chosen for your capstone project. • Explore how you plan to structure the project's front-end and back-end components to ensure seamless integration and maintainability.		
	6. Database Management and Integration	Advanced database concepts, SQL and NoSQL databases, ORM tools, Database migrations	• "Database System Concepts" by Silberschatz, Korth, and Sudarshan • Online tutorials on ORM tools (e.g., Sequelize, Mongoose)	Designing and implementing a database schema
	Discussion by Group 6	• Present your capstone project's database schema and justify your choice of SQL or NoSQL databases. • Discuss how ORM tools will be used to manage data migrations and ensure data integrity in your application.		
	7. Application Security and Performance	Common vulnerabilities, Secure coding practices, Caching strategies, Code optimization	• "Web Application Security" by Andrew Hoffman • OWASP Top Ten documentation	Conducting a security audit of a web application
	Discussion by Group 7	• Discuss the security measures you've implemented in your capstone project to protect against common vulnerabilities. • Additionally, explore strategies for optimizing your code and improving the performance of your application.		

	8. Deployment and DevOps	CI/CD pipelines, Cloud deployment, Containerization with Docker and Kubernetes	• "The DevOps Handbook" by Gene Kim et al. • Online tutorials on Docker and Kubernetes	Setting up a CI/CD pipeline for a project
		• Explain your plan for deploying your capstone project, including the use of CI/CD pipelines. • Discuss how containerization (e.g., using Docker) could enhance the scalability and reliability of your deployed application.		

Mode of Assessment:

- Quizzes (20%)
  - Regular assessments to test understanding of core concepts
- Capstone Project (30%)
  - Team-based project to develop a real-life application.
  - Final presentation and code review
- Discussions (10%)
  - Active participation in lectures and workshops
- Exams (40%)
  - End-of-semester exam

Reading List:

- "Agile Software Development" by Robert C. Martin
- "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma et al.
- "Introduction to Algorithms" by Cormen et al.
- "Eloquent JavaScript" by Marijn Haverbeke
- "Learning Web Design" by Jennifer Robbins
- "Pro MERN Stack" by Vasan Subramanian
- "Database System Concepts" by Silberschatz, Korth, and Sudarshan
- "Web Application Security" by Andrew Hoffman
- "The DevOps Handbook" by Gene Kim et al.
- "Scrum: The Art of Doing Twice the Work in Half the Time" by Jeff Sutherland

Semester-Long Application Development Project: Appointment Booking App for a Health Clinic

Case Study Overview

The capstone project now involves designing and implementing a simple appointment booking app for a health clinic. The clinic provides various medical services, and patients need an easy way to book, reschedule, and cancel appointments. The app should also help clinic staff manage appointments efficiently, ensuring that doctors' schedules are organized and patients receive timely reminders.

Requirements for the Appointment Booking App

1. **User Management**
    a. Patient Accounts: Patients should be able to create accounts with basic personal information, including name, contact details, and medical history.
    b. Clinic Staff Accounts: Implement accounts for clinic staff with roles such as Admin and Receptionist, each with different access permissions.
    c. User Authentication: Secure login system with options for password recovery and account management.
2. **Appointment Scheduling**
    a. Booking Appointments: Patients can view available time slots for different doctors and book appointments based on their preferred time and service.
    b. Rescheduling/Cancellation: Provide functionality for patients to reschedule or cancel appointments. Update the clinic's schedule accordingly.
    c. Doctor Availability: Manage and display doctors' availability, ensuring patients can only book appointments during available slots.
3. **Notifications and Reminders**
    a. Appointment Confirmations: Send confirmation messages to patients once an appointment is booked.
    b. Reminders: Automatically send reminders to patients before their appointment (e.g., 24 hours in advance).
    c. Cancellation Alerts: Notify clinic staff and doctors of any cancellations or changes in the schedule.
4. **Clinic Management**
    a. Doctor Schedules: Allow doctors to manage their schedules, including setting availability and blocking off times when they are unavailable.
    b. Patient History: Provide a way for doctors to view a patient's appointment history and notes from previous visits.

    c. Daily Appointment Summary: Generate a daily summary of all scheduled appointments for clinic staff to prepare accordingly.
5. **Reporting and Analytics**
    a. Appointment Reports: Generate reports on appointment statistics, such as the number of appointments per week, the most popular services, and patient no-show rates.
    b. Patient Statistics: Track patient visits over time, including frequent visitors and appointment patterns.
6. **User Interface and Experience**
    a. Simple and Intuitive UI: Design a user-friendly interface that allows patients to book and manage their appointments easily and with minimal effort.
    b. Mobile Compatibility: Ensure the app is responsive and works well on mobile devices, allowing patients to book appointments.
7. Data Backup and Security
    a. Data Protection: Implement basic security measures to protect patient information and appointment data.

Expected Outcomes

By the end of the project, students should deliver a functional appointment booking app deployed on a web server that meets the clinic's needs. The app should be thoroughly tested, and students should provide documentation explaining the design choices, the technologies used, and how the app can be extended or scaled.

In this semester-long project, students will design and develop a comprehensive web application that addresses a real-world problem. The project will focus on implementing the Model-View-Controller (MVC) architecture using software frameworks, clearly differentiating the front and back end.

Project Tasks and Deliverables

Part 1: Project Proposal and Design (Due 22nd Sept 2024)

- **Task 1: Project Proposal**
    - Form teams of 4-5 students.
    - Submit a project proposal outlining the project's scope, objectives, and technologies/frameworks to be used.

- **Deliverable:** Project Proposal Document (including team members, project description, and chosen frameworks). (No more than 500 words)

- **Task 2: System Design and Architecture**
  - Design the system architecture, including the database schema and MVC structure.
  - Define the core features and user interface.
  - **Deliverable:** System Design Documentation and MVC Architecture Diagram.

Part 2: Development (Due 28th October 2024)

- **Task 3: Front-end Development**
  - Develop the application's front end using a chosen software framework (e.g., React, Angular, Vue.js).
  - Implement user interfaces for core functionalities.
  - **Deliverable:** Front-end codebase and a functional front-end interface. Write a one-page description of your core functions and implementation. There is no need to provide a code.
- **Task 4: Back-end Development**
  - Develop the system's back end using a framework or technology stack (e.g., Node.js, Django, Ruby on Rails).
  - Implement the server, handle user requests, and interact with the database.
  - Secure and optimize the back end.
  - **Deliverable:** Back-end codebase and a functional back-end system. Provide a link to your GitLab repository for the developed app and a URL link to your system.
- **Task 5: Integration**
  - Integrate the front-end and back-end to establish communication using RESTful APIs or other methods.
  - Test the entire system for functionality and performance.
  - Address any integration issues.

- **Deliverable:** Integrated Web Application. Provide a brief description of the system with screenshots. Provide a report on integration tests. Describe your RESTful API. (No more than two pages)

Part 3: Testing and Presentation (Due 24th Nov 2024)

- **Task 6: User Testing and Documentation**
  - Conduct user testing to gather feedback and make necessary improvements.
  - Create user and technical documentation, including setup instructions and user guides.
  - Prepare the final project report.
  - **Deliverable:** User feedback reports, documentation, and a final project report.
- **Task 7: Project Presentation**
  - Present the web application to the class, demonstrating its features and functionality.
  - Showcase the use of MVC architecture and chosen frameworks.
  - Receive feedback from peers and the instructor.
  - **Deliverable:** Project presentation and live demonstration.
- **Task 8: Final Submission**
  - Submit the complete project, including all documentation, code, and presentation materials.
  - Ensure that the application is hosted and accessible for evaluation.
  - **Deliverable:** Final Project Submission.

Report Format

1. **Title Page**
   a. Project Title: Clearly state the name of your project (e.g., "Appointment Booking App for Health Clinic").
   b. Team Members: List the names of all team members involved in the project.
   c. Date: Include the submission date.

2. **Abstract**
    a. Brief Summary: Provide a concise summary of your project, including the main goals, the problem it addresses, and the key features of the app. Mention the technologies used and the overall outcome (150-200 words).
3. **Introduction**
    a. Background and Context: Explain the problem your app is solving. Describe the challenges clinics face in managing appointments manually and how your app addresses these challenges.
    b. Objectives and Scope: Clearly outline the project's objectives.
4. **System Design**
    a. Architecture Diagram: Present a high-level architecture diagram of your app, showing how different components (e.g., front-end, back-end, database) interact.
    b. Database Schema: Include the database schema that defines the structure of your database, detailing tables such as users, appointments, and doctors.
    c. MVC Structure: Describe how you've implemented the Model-View-Controller (MVC) architecture. Explain how each part (Model, View, Controller) handles data, the user interface, and the application logic.
5. **Implementation**
    a. Front-End Development: Discuss the technologies and frameworks used for the front-end (e.g., React, Bootstrap). Explain how you designed the user interface, focusing on user experience, responsiveness, and accessibility.
    b. Back-End Development: Describe the back-end setup, including the server, database integration, and API development. Mention the programming language, frameworks (e.g., Node.js, Django), and any third-party libraries used.
    c. Integration: Explain how you integrated the front-end with the back-end, ensuring seamless data flow and communication. Discuss any challenges faced during integration and how they were resolved.
6. **Testing**
    a. Testing Methodologies: Detail the testing strategies you used, such as unit testing, integration testing, and user acceptance testing. Mention any automated testing tools used.
    b. Test Cases and Results: Provide examples of test cases, particularly those critical to the app's functionality, like booking, rescheduling, and cancelling appointments. Summarize the test results and any bugs found and fixed.
7. **Conclusion**
    a. Summary of Achievements: Reflect on what was accomplished during the project. Highlight key successes, such as the app's deployment, its effectiveness in solving the problem, and any positive feedback.
    b. Future Work: Suggest areas for future improvement or expansion, such as adding new features (e.g., integration with electronic health records), enhancing security, or scaling the app for larger clinics.
8. References

    a. List of References and Resources Used: Cite all the resources, including books, articles, online tutorials, and software libraries, that you consulted or used during the project.

9. Appendices
    a. Additional Materials: Include any supplementary materials that support your report. This could be code snippets, additional diagrams, screenshots of the app in action, or detailed user feedback forms.

Assessment and Grading

Table

Component	Weight
Project Proposal	5%
System Design Documentation and MVC Architecture Diagram	10%
Front-end Development	15%
Back-end Development	15%
Integration	20%
User Testing and Documentation	10%
Project Presentation	15%
Final Project Submission	10%