Lecture 9 –Testing and Deployment

IST 3108 Application Development

Dr. Peter Khisa Wakholi

Dept of Information Systems, Makerere University

Agenda

- Testing
- Progressive Web Apps
- Deployment

Resources

- "JavaScript Testing Recipes" by James C. Shore and Russel B. Nemeroff.
- "Don't Make Me Think" by Steve Krug for Usability Testing.
- Online resources such as MDN Web Docs for JavaScript debugging techniques.

Testing

Part 1

Importance of Testing

Quality Assurance: Ensures that the application meets specified requirements and functions correctly.

Bug Identification: Helps identify and rectify bugs and errors before deployment, enhancing overall system stability.

User Experience: Testing contributes to a positive user experience by identifying and resolving usability issues.

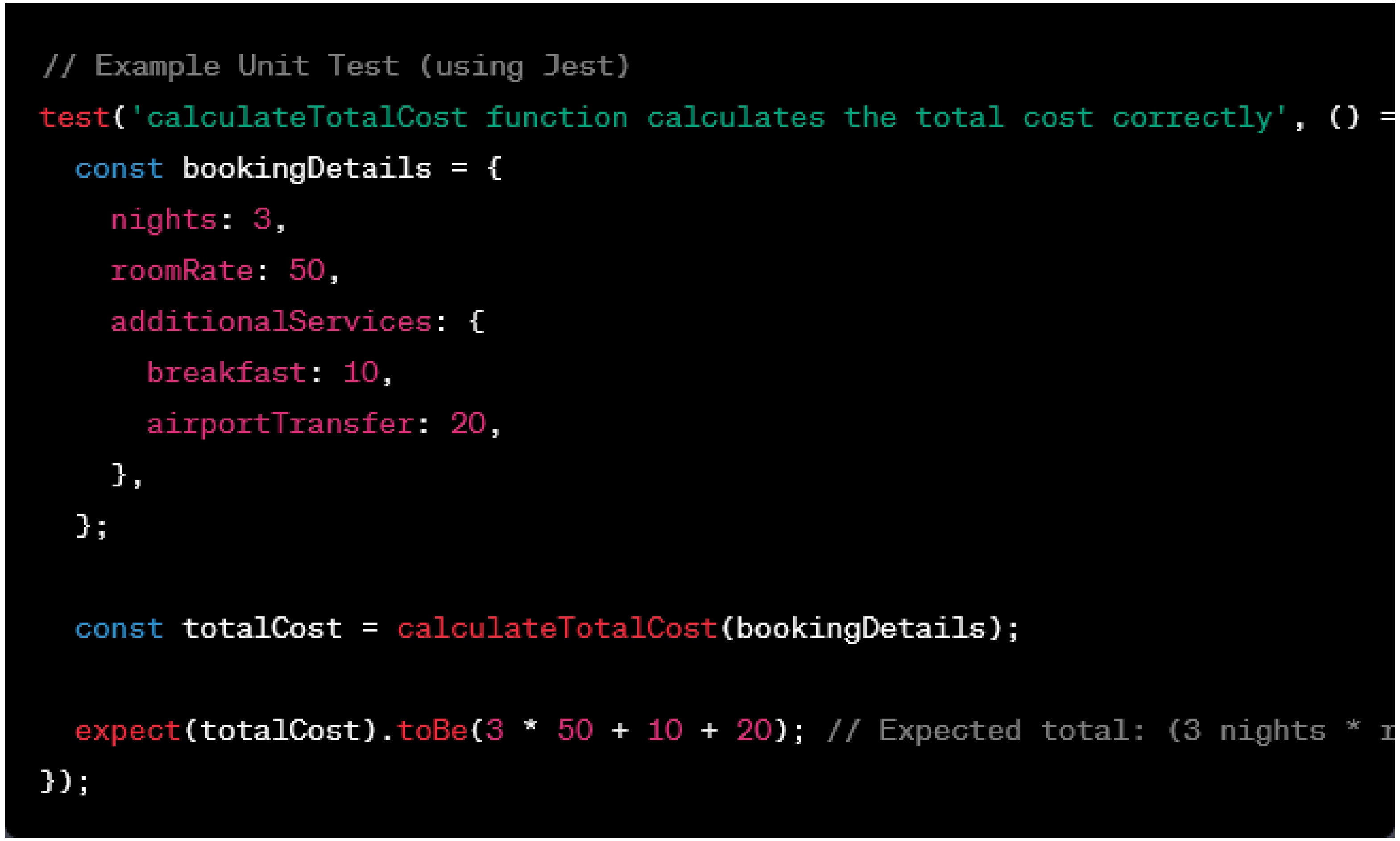
Security: Testing helps uncover and address security vulnerabilities, safeguarding user data.

Overview of Testing

- Manual Testing: Testing carried out by human testers manually without the use of automated tools.
  - **Unit Testing:** Verifies the correctness of a specific function.
  - **Integration Testing:** Ensures components work together as expected.
  - **System Testing**: Validates the end-to-end functionality of the entire system.
  - **User Acceptance Testing (UAT):** Validates that the app meets user expectations.
- Automated Testing: Testing conducted with the assistance of automated tools to validate software functionality.
  - **Regression Testing:** Ensures existing functionalities work after introducing changes.
  - **Performance Testing:** Evaluates the system's responsiveness and resource usage under load.
  - **Security Testing:** Identifies and addresses security vulnerabilities in the application.

Unit Testing

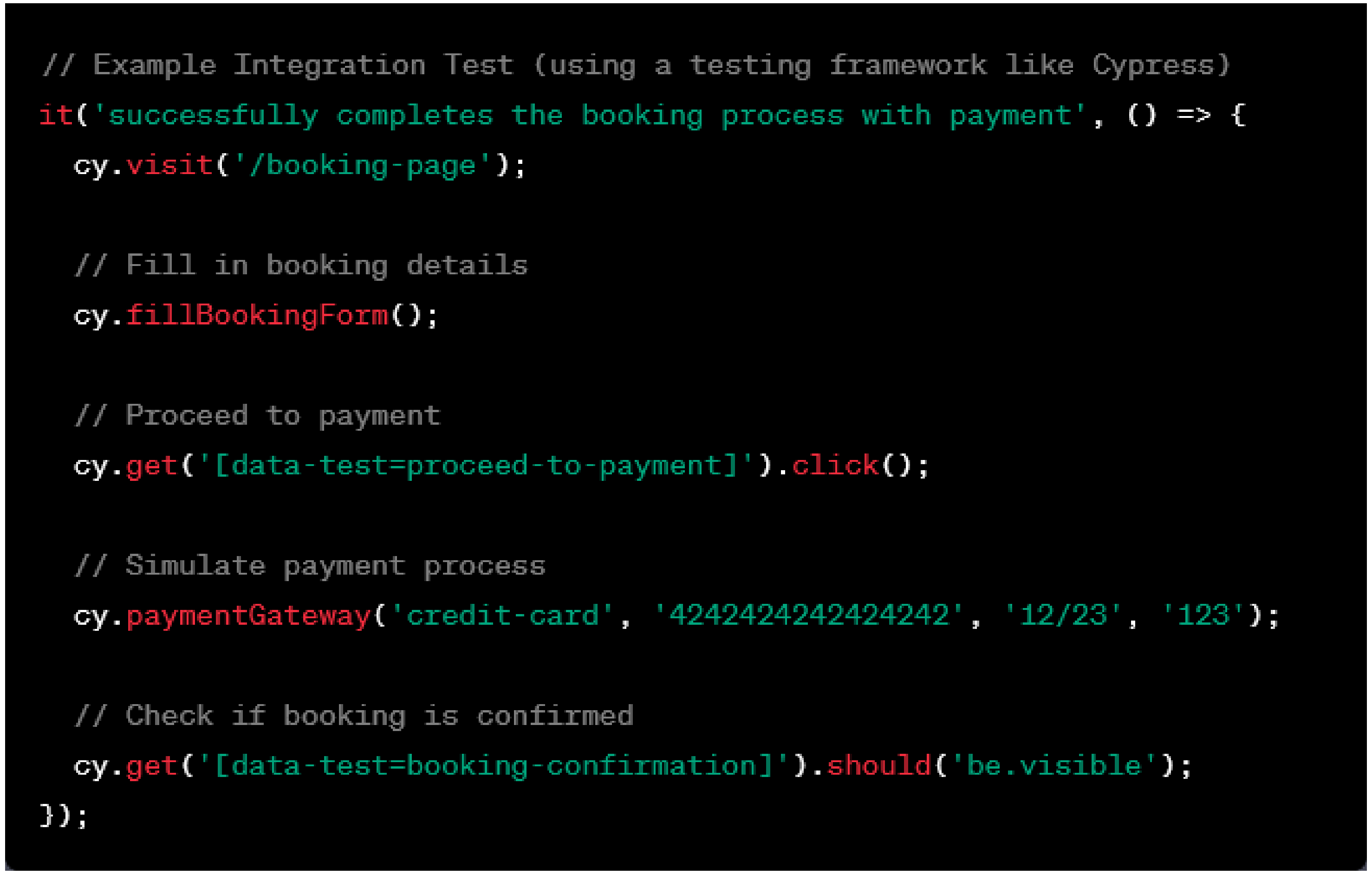
Scenario: Testing the function responsible for calculating the total cost of a hostel booking.



```
// Example Unit Test (using Jest)
test('calculateTotalCost function calculates the total cost correctly', () =
  const bookingDetails = {
    nights: 3,
    roomRate: 50,
    additionalServices: {
      breakfast: 10,
      airportTransfer: 20,
    },
  };

  const totalCost = calculateTotalCost(bookingDetails);

  expect(totalCost).toBe(3 * 50 + 10 + 20); // Expected total: (3 nights * r
});
```

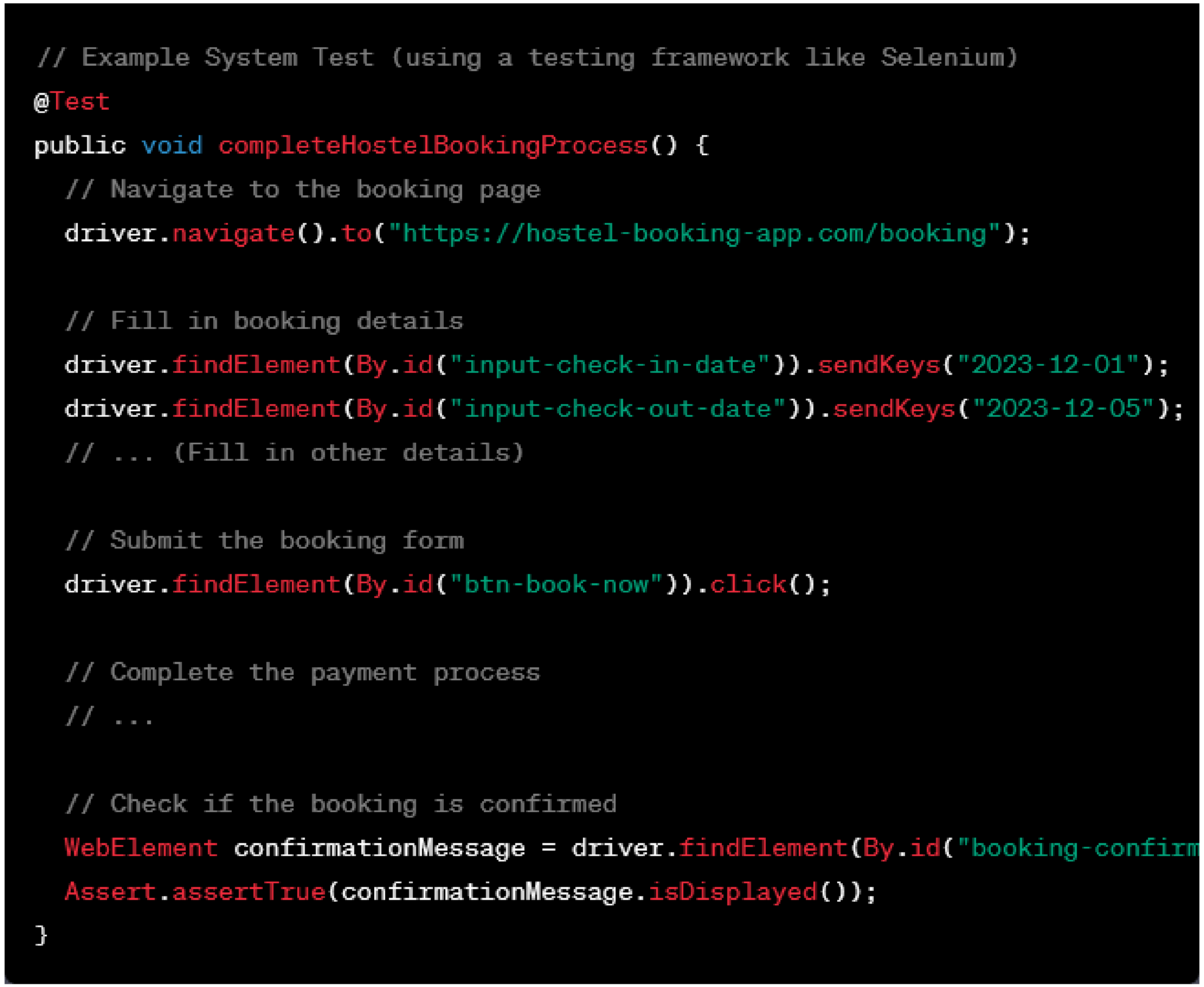
Integration Testing

Scenario: Testing the interaction between the booking system and the payment gateway.


```
// Example Integration Test (using a testing framework like Cypress)
it('successfully completes the booking process with payment', () => {
  cy.visit('/booking-page');

  // Fill in booking details
  cy.fillBookingForm();

  // Proceed to payment
  cy.get('[data-test=proceed-to-payment]').click();

  // Simulate payment process
  cy.paymentGateway('credit-card', '4242424242424242', '12/23', '123');

  // Check if booking is confirmed
  cy.get('[data-test=booking-confirmation]').should('be.visible');
});
```

System Testing:

- Scenario: Testing the end-to-end flow of booking a hostel, including interactions with the database.



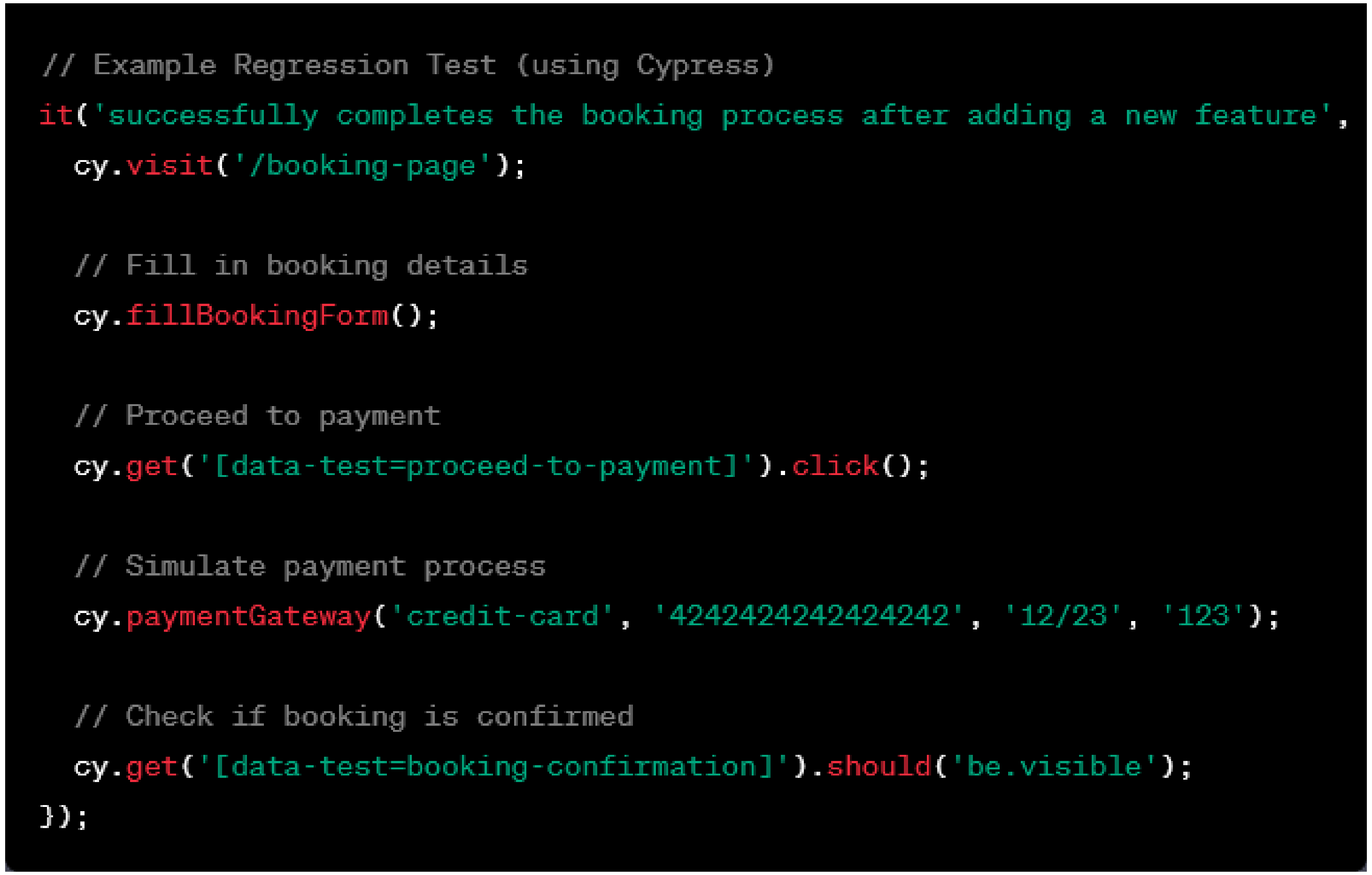
```java
// Example System Test (using a testing framework like Selenium)
@Test
public void completeHostelBookingProcess() {
  // Navigate to the booking page
  driver.navigate().to("https://hostel-booking-app.com/booking");

  // Fill in booking details
  driver.findElement(By.id("input-check-in-date")).sendKeys("2023-12-01");
  driver.findElement(By.id("input-check-out-date")).sendKeys("2023-12-05");
  // ... (Fill in other details)

  // Submit the booking form
  driver.findElement(By.id("btn-book-now")).click();

  // Complete the payment process
  // ...

  // Check if the booking is confirmed
  WebElement confirmationMessage = driver.findElement(By.id("booking-confirm
  Assert.assertTrue(confirmationMessage.isDisplayed());
}
```

User Acceptance Testing (UAT)

- Scenario: A user, without technical knowledge, tests the app to ensure it meets their expectations
  - As a user,
  - I navigate to the hostel booking app,
  - I fill in my details and select a room,
  - I proceed to payment and complete the booking process,
  - I check if the confirmation message is displayed.

Regression Testing

- Scenario: After adding a new feature, ensure existing functionalities, like the booking process, still work as expected.



```
// Example Regression Test (using Cypress)
it('successfully completes the booking process after adding a new feature',
  cy.visit('/booking-page');

  // Fill in booking details
  cy.fillBookingForm();

  // Proceed to payment
  cy.get('[data-test=proceed-to-payment]').click();

  // Simulate payment process
  cy.paymentGateway('credit-card', '4242424242424242', '12/23', '123');

  // Check if booking is confirmed
  cy.get('[data-test=booking-confirmation]').should('be.visible');
});
```

Performance Testing

- Scenario: Simulating multiple users accessing the booking page simultaneously to evaluate system responsiveness.
- Performance Testing Tool: Apache Jmeter
- Test Plan:
  - 1. Simulate 100 users concurrently accessing the booking page.
  - 2. Measure response times for loading the page and submitting the booking form.
  - 3. Analyze the throughput and response time metrics to ensure satisfactory performance under load.

Security Testing

- Scenario: Checking for vulnerabilities in the hostel booking app's payment processing.
- Security Testing Tool: OWASP ZAP
- Test Steps:
  - 1. Perform a security scan on the payment processing module.
  - 2. Identify and fix security vulnerabilities such as SQL injection or cross-site scripting.
  - 3. Ensure secure communication between the app and the payment gateway.

Debugging Techniques

- **Understanding Errors:** A crucial aspect of debugging involves understanding error messages and identifying their root causes.
- **Console Logging**: Utilizing browser developer tools to log messages and inspect variables during runtime.
- Common Debugging Techniques
  - **Breakpoints:** Setting breakpoints in the code to pause execution and inspect variables at specific points.
  - **Step-through Debugging:** Stepping through code line by line to trace the flow and identify issues.
  - **Browser Developer Tools:** Leveraging browser tools like Chrome DevTools to inspect and debug web applications.

Progressive Web Apps

Part 2

Progressive Web Apps (PWAs)

- **Definition:** PWAs use modern web capabilities to deliver an app-like experience to users. They are reliable, fast, and engaging.
- **Key Features:**
    - Offline Functionality
    - App-Like Experience
    - Push Notifications
    - Responsive Design

Examples

- Twitter Lite: https://twitter.com/
- Why it's Progressive:
  - Offline Functionality: Twitter Lite allows users to browse tweets even when offline.
  - Fast Loading: It loads quickly, even on slower network connections.
  - Add to Home Screen: Users can add it to their home screen for a more app-like experience.
- Uber: https://m.uber.com/
- Why it's Progressive:
  - Fast Loading: Uber's PWA loads quickly, even on slower network connections.
  - Add to Home Screen: Users can add the Uber PWA icon to their home screen.
  - Offline Functionality: It supports basic functionality like ride booking offline.

Responsive Design Principles

Responsive design ensures that a web application adapts to different devices and screen sizes, providing an optimal viewing experience.

Media Queries: CSS techniques to apply styles based on device characteristics.

Flexible Grids and Layouts: Designing layouts that can adapt to different screen sizes.

Mobile-First Design: Prioritizing design for mobile devices, then scaling up.

Deployment and DevOps

Part 3

References

- Newman, P. (2017). "Building Microservices." O'Reilly Media.
- Wiggins, A., & Smith, A. (2016). "DevOps: A Software Architect's Perspective." Pearson.
- AWS Documentation. https://docs.aws.amazon.com/
- Microsoft Azure Documentation. https://docs.microsoft.com/
- Google Cloud Platform Documentation. https://cloud.google.com/docs

Deploying Web Applications to Cloud Platforms

Deployment is the process of making an application accessible to users.

Proper deployment ensures the application is live, reliable, and scalable.

Cloud Platforms

Benefits of Cloud Deployment

- Scalability, flexibility, and cost-effectiveness.
- Automation of deployment processes.
- High availability and redundancy.

Popular Cloud Platforms

- **Amazon Web Services (AWS):** Offers a wide range of services for deployment.
- **Microsoft Azure:** Provides integrated tools for deployment and management.
- **Google Cloud Platform (GCP):** Emphasizes scalability and machine learning.

Deployment Strategies

- **Blue-Green Deployment:** Reduces downtime by switching between two identical environments.
- **Canary Deployment:** Gradual release to a subset of users for testing.

Hosting Options for Web Applications

Shared Hosting

- Multiple websites share resources on a single server.
- Cost-effective for smaller projects.

Virtual Private Server (VPS)

- Dedicated resources within a virtual environment.
- More control and customization compared to shared hosting.

Dedicated Hosting

- Entire server dedicated to a single website.
- High performance and control.

Cloud Hosting

- Resources provided by a cloud service provider.
- Scalable and pay-as-you-go pricing.

Serverless Hosting

- No need to manage servers; functions run in response to events.
- Scales automatically based on demand.

DevOps in Deployment

- **What is DevOps**
  - **DevOps Culture:** Collaboration between development and operations teams.
  - **Automation:** Streamlining processes for efficiency.
- **Key DevOps Practices**
  - **Continuous Integration (CI):** Regularly merging code changes to a shared repository.
  - **Continuous Deployment (CD):** Automatically deploying code changes to production.
- **Tools for DevOps**
  - **Jenkins:** Automation server for building, testing, and deploying.
  - **Docker:** Containerization for consistent deployment.
  - **Kubernetes:** Container orchestration for managing containerized applications.

Task 6 In Detail

Part 5

Conducting User Testing:

- **Objective:** Gather feedback from potential users to identify usability issues and areas for improvement.
- **Process:**
    - Select a diverse group of users, including potential hostel residents and administrators.
    - Develop test scenarios and tasks to cover various features of the app.
    - Observe and record user interactions, paying attention to user preferences and pain points.
    - Collect feedback through surveys, interviews, or usability testing tools.
    - Analyze the gathered feedback to prioritize and address identified issues.

Implementing Necessary Improvements:

- **Action Steps:**
  - Address critical issues that impact user experience.
  - Implement user-suggested improvements.
  - Iteratively test and refine the application based on user feedback.
  - Ensure that the application aligns with user expectations and requirements.

Task 7&8 in Detail

Part 5

Slides and Visuals

- **Introduction:** Briefly introduce the project, its purpose, and the problem it addresses.
- **Features and Functionality:** Highlight key features of the Hostel Booking System.
- **Architecture Overview:** Showcase the chosen MVC architecture and frameworks.
- **User Interface:** Present screenshots or live views of the user interface.
- **User Feedback:** Share insights gained from user testing and any adjustments made.
- **Challenges and Solutions:** Discuss challenges faced during development and how they were overcome.
- **Future Enhancements:** Mention potential future improvements or features.

Final Submission Report:

- **Front Page (0.5 Page)**
  - Overview of the final project submission.
  - Confirmation of hosting and accessibility.
  - Group Names, RegNo. StudentNo. GITlab username,
  - Admin account for access code on GITlab
- **Introduction (0.5 Page)**
  - Brief overview of the Hostel Booking System project.
  - Objectives and goals.
- **System Design and Architecture (1 Page)**
  - Overview of the chosen system architecture.
  - Description of the database schema.
  - MVC structure implementation.
- **Front-end Development (2 Pages)**
  - Summary of front-end technologies and frameworks used.
  - Design principles and considerations.
  - Highlights of user interface design.

Final Submission Report

- **Back-end Development (2 Pages)**
  - Overview of back-end technologies and frameworks used.
  - Discussion of server implementation.
  - Database schema and interactions.
- **Integration and Testing (1.5 Pages)**
  - Description of the integration process between front-end and back-end.
  - Overview of testing strategies employed.
  - Test results and identified issues.
- **User Testing and Documentation (1 Page)**
  - Overview of user testing procedures.
  - User feedback and improvements made.
  - Creation of user and technical documentation.

Final Submission Report

- **Conclusion (0.5 Page)**
  - Reflection on the entire project lifecycle.
  - Lessons learned and areas for improvement.
- **Appendix (2 Page)**
  - Any supplementary materials, charts, or diagrams.