

DevOpps

by Paul Newman

Submission date: 09-Jan-2026 12:47PM (UTC+0000)

Submission ID: 272669479

File name: 11910_Paul_Newman_DevOpps_694650_868529166.docx (718.12K)

Word count: 5344

Character count: 35161



Agile Software Development and DevOps

Paul Newman 1805494

Module ACCA7023

Nathan Morgan

Contents

Table of Contents	3
Task One.....	4
Examples of User story's	4
Task Three Part One	12
Introduction to Dev Operations	12
Conclusion to IDO:	17
Task Three Part Two	17
Web application and its purpose	18
Architecture Diagram:.....	18
Software DevOps Principles Analysis	19
Continuous Integration (CI):.....	19
Continuous Delivery (CD):.....	19
Automation:	20
Agile Support:.....	20
Identify potential challenges.....	20
Challenges and improve quality of the web application.....	21
Postman	22
Selenium	23
Bibliography	24

Table of Contents

Figure 1 Room Booking Requirement – Part 1.....	5
Figure 2 Room Booking Requirement – Part 2.....	5
Figure 3 Room Booking Requirement – Part 3.....	5
Figure 4 Room Booking Requirement – Part 4.....	6
Figure 5 Room Booking Requirement – Part 5.....	6
Figure 6 Alien Capability Definition Table.....	7
Figure 7 Player Capability Definition Table	7
Figure 8 Level Environment and Gameplay Area.....	8
Figure 9 Player Buffs and Power-Up Effects.....	8
Figure 10 User Interaction Design.....	9
Figure 11 GitHub screen shot.....	10
Figure 12 Architecture of a Library System	19
Figure 13 Postman with API with GET request	23

Task One

Stakeholder Requirements and Use Scenarios

User requirements are expressed as brief, user-centred descriptions that outline expected system behaviour from an end-user perspective. that define required features or functionality from the perspective of the end user within a DevOps environment. Their purpose is to help development and operations teams understand user requirements and support the incremental delivery of value. A typical user story follows the format: Each requirement is structured to identify the user role, the intended system action, and the resulting benefit or outcome. For example, Automated deployment pipelines are required to support system administrators by reducing the likelihood of human error and increasing deployment efficiency. By promoting user-centred design this approach supports alignment between user expectations, technical implementation, and project By clearly defining functional requirements, this approach aligns project objectives between development and operations teams involved in system implementation and maintenance. And business stakeholders throughout the DevOps lifecycle. They support task prioritisation, define acceptance criteria, and ensure continuous delivery meets user expectations. User stories are commonly managed using agile project management tools.

Examples of User story's

Structuring user groups to guide the community centre room booking process improves usability and addresses the varying needs of different user audiences. By categorising users into groups such as local businesses, event organisers, and community members, the reservation system can be designed to better support their specific requirements. This approach helps ensure the booking platform remains inclusive and responsive to user expectations.

For example, community users may prioritise a simple and efficient booking process, while event organisers often require more advanced features such as recurring reservations or requests for additional facilities. Incorporating feedback from these groups allows the system to provide an intuitive interface with clear guidance and features such as availability calendars, integrated payments, and automated booking confirmations.

Involving user groups during development and testing promotes inclusivity and ensures the system meets real-world needs. This collaborative approach improves the reservation process while strengthening community engagement and trust, encouraging broader use of the centre's facilities.

Agile Dev and Software Development Ops

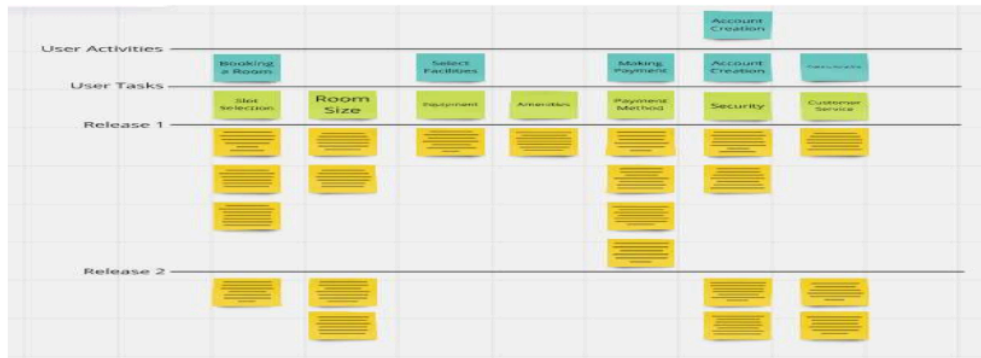


Figure 1 Room Booking Requirement – Part 1



Figure 2 Room Booking Requirement – Part 2



Figure 3 Room Booking Requirement – Part 3

Agile Dev and Software Development Ops

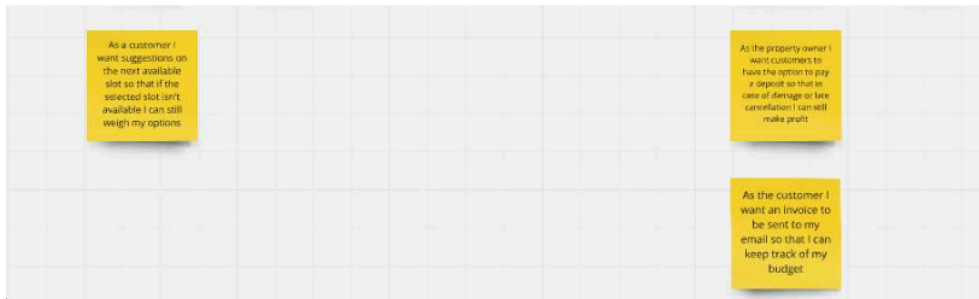


Figure 4 Room Booking Requirement – Part 4



Figure 5 Room Booking Requirement – Part 5

Implementation of a Space Invaders-Inspired Game

Developing game classes for a *Space Invaders* style application requires organising code into modular, reusable components to support maintainability and scalability. that represent distinct game features. Each class encapsulates specific attributes and behaviours, supporting clarity, maintainability, and future development.

Alien Class: This class represents the alien enemies within the game and defines key attributes such as position, category, and behavioural logic. It also implements methods responsible for movement, attack behaviour, and collision handling.

Class Name: Alien	
Responsibilities	Collaborators
Shooting Dying Types of Aliens Movement Movement Speed	User Interface Level Layout

Figure 6 Alien Capability Definition Table

Player Object Definition: This class models the player-controlled spacecraft and includes attributes such as position, movement speed, and remaining lives. It also defines the functionality for player **Movement and firing projectiles.**

Class Name: Player	
Responsibilities	Collaborators
Keeping Score Using Buffs Dying Shooting Movement Movement Speed	User Interface Powerups

Figure 7 Player Capability Definition Table

Level Layout Class: This class oversees the core game logic, managing elements such as enemy generation, level progression, and score tracking. It functions as the central controller, coordinating interactions between the various components of the game.

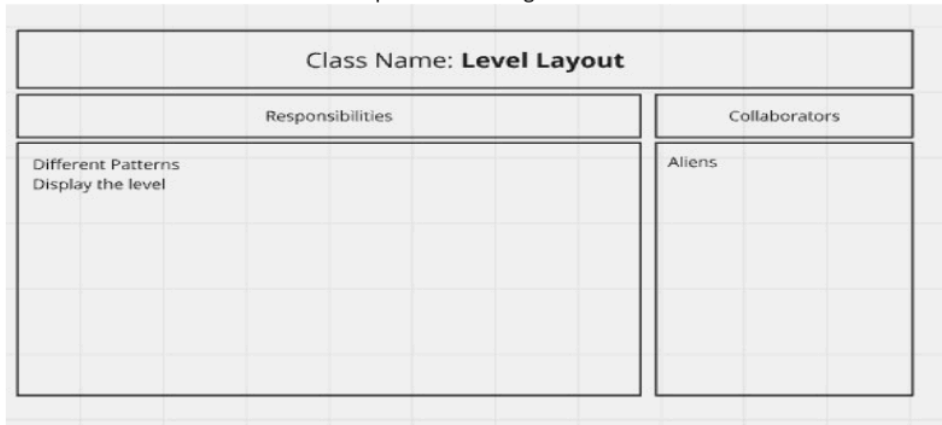


Figure 8 Level Environment and Gameplay Area

Powerup Class: This class is responsible for handling projectiles fired by both the player and enemy entities. It maintains attributes such as position and velocity, while providing methods to update Movement and perform collision detection.

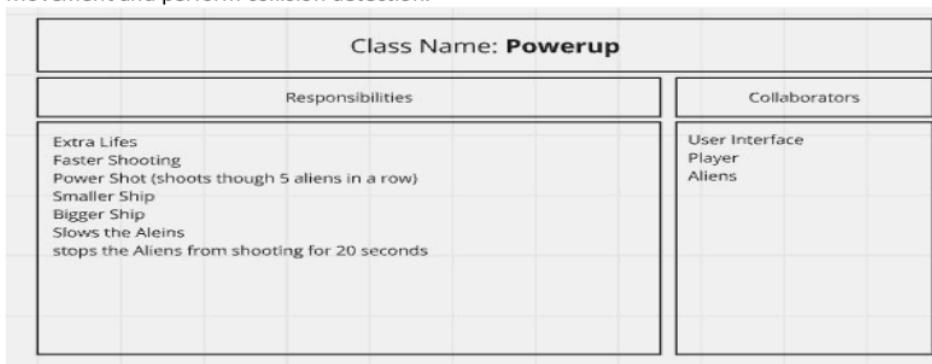


Figure 9 Player Buffs and Power-Up Effects

UI Class: This class manages the presentation of user interface elements, including the score display, remaining lives, and game-over screens. By separating visual components from core game logic, it ensures a clear distinction between presentation and functionality.

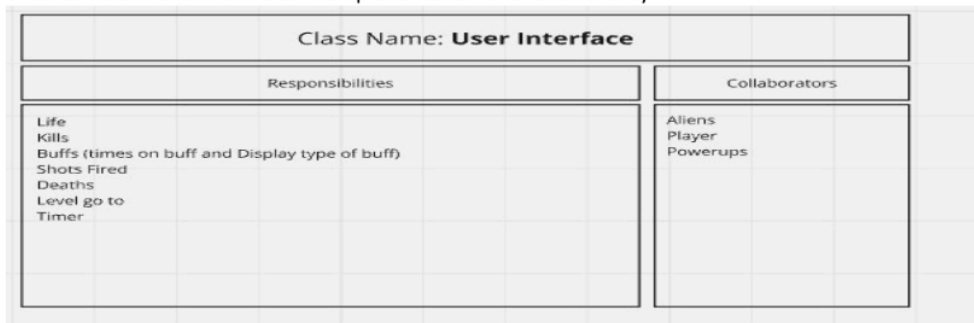


Figure 10 User Interaction Design

This structured approach improves scalability and simplifies debugging, while allowing new features such as power-ups or additional enemy types to be added with minimal impact on existing code. The use of object-oriented principles, including inheritance and polymorphism, also supports the reuse of shared behaviours, such as implementing specialised enemy subclasses with distinct attack patterns.

Introduction to GitHub

GitHub is an online platform designed to support software version control and collaborative development. It enables developers to manage source code through cloud-hosted repositories, track changes over time, and coordinate contributions efficiently. The platform is built on Git, a distributed version control system originally developed by Linus Torvalds, and provides tools that support structured collaboration and code management. The key features of GitHub are outlined below [1].

- **Repositories (Repos):** Repositories function similarly to project directories by storing source code alongside a complete history of changes. They can be configured as private, limiting access to authorised users, or public, allowing the contents to be viewed by anyone. [2].
- **Version Control:** GitHub uses Git to track and manage changes to source code, allowing developers to see when modifications were made, who made them, and to revert to earlier versions when necessary [3].
- **Branching and Merging:** GitHub allows developers to create separate branches, enabling new features or changes to be developed independently and later merged into the main codebase without disrupting the original version. [2] [3].
- **Pull Requests:** A pull request is created when a contributor proposes merging changes into the main project, allowing team members to review updates, provide feedback, and approve or reject the merge.
- **Issues and Project Management:** GitHub enables developers worldwide to collaborate on open-source projects, fostering a strong and active community. The platform also supports networking, shared learning, and the demonstration of technical skills to potential employers. [3] [1].
- **Community and Collaboration:** GitHub enables developers worldwide to collaborate on open-source projects, fostering a strong and active community. It also supports networking, shared learning, and the demonstration of technical skills to potential employers.
- **Integrations:** GitHub integrates with a wide range of widely used tools, including continuous integration and continuous deployment (CI/CD) services, which play a vital role in supporting automated DevOps pipelines. [2].

GitHub supports collaborative software development and documentation using version control mechanisms, and collaboration for both individuals and organisations. It promotes open-source contribution by enabling developers to share code, collaborate on projects, and resolve issues collectively. Through core features such as repositories, version control, branching, and pull requests,

developers can work independently and integrate changes through structured review processes that help maintain code quality and project stability [3].

GitHub includes integrated tools that support task tracking and workflow organisation, and is also widely used by developers to present and share their work [2] [4].

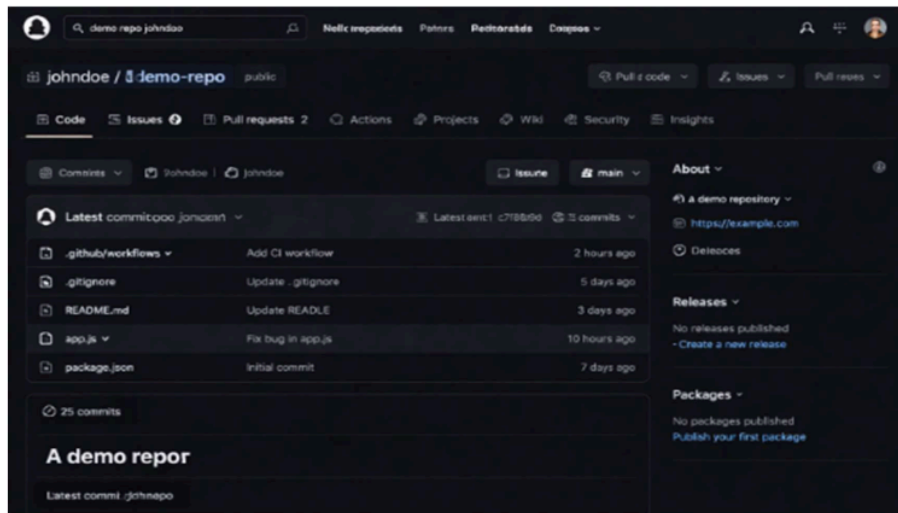


Figure 11 GitHub screen shot

GitHub's integration with CI/CD pipelines and automation tools streamlines DevOps workflows by unifying code development, testing, and deployment. As a result, GitHub supports a wide range of use cases, from large-scale and to contribute to collaborative open-source projects. [1] [4].

Overview of the Scrum Framework

Development Toolchain Complexity: Managing multiple DevOps tools for collaboration, version control, monitoring, and automated deployment can present significant challenges for organisations. Poor coordination of complex toolchains may result in miscommunication, inefficiencies, and reduced productivity. Addressing these issues requires a clear strategy to ensure consistent tool adoption across teams, supported by well-defined guidelines for tool selection, integration, and use. Organisations should also invest in targeted training to help teams understand the capabilities and benefits of selected tools. Regular review and refinement of the toolchain can further improve efficiency by streamlining workflows and reducing redundancy. Prioritising strategic toolchain management enables organisations to minimise complexity, enhance collaboration, and maximise the effectiveness of DevOps practices.

Managing Software Supply Chain Risks

Managing risks within the software supply chain is essential for secure system delivery management that focuses on identifying, assessing, and mitigating risks that may disrupt supply chain operations.

Core Components of the Software Supply Chain

Supply Chain Risk Identification:

- **Risk Categories:** These risks may originate from various sources, including natural disasters, geopolitical uncertainty, supplier disruption, economic fluctuations, cyber security threats, and regulatory changes.
- **Origins of Risk:** Supply chain risks may originate internally, such as from supplier-related issues, or externally, including factors like market instability.

Risk Assessment:

- **Risk Analysis:** Both likelihood and impact are evaluated during this process. of identified risks, often using a combination of qualitative and quantitative methods.
- **Risk Ranking:** Risks are prioritised based on their impact on supply chain performance and broader organisational objectives.
- **Risk Reduction Strategies:**
- **Supplier Diversification:** This approach reduces reliance on a single supplier or geographic location by diversifying the supplier network.
- **Inventory Management:** This strategy includes holding safety stock or buffer inventory to help absorb disruptions and reduce the impact of supply chain shocks.
- **Supplier Relationships:** Establishing strong supplier partnerships improves communication and collaboration, particularly during periods of disruption. Contingency planning involves developing structured response procedures that enable organisations to react quickly and effectively to different risk scenarios.
- **Monitoring and Review:**
- **Continuous Monitoring:** This process requires continuous and regular monitoring to identify emerging supply chain risks. and changes in existing risk conditions.
- **Performance Indicators:** Performance Measurement Indicators can be implemented to measure and assess the effectiveness of supply chain risk management strategies.

Integration of Technologies

- **Analytical Data Processing:** Analytical and predictive modelling techniques can be employed to anticipate potential risks and assess their likely impacts.
- **Visibility Across the Supply Chain:** This involves deploying tools that offer real-time visibility across the supply chain, allowing organisations to respond more rapidly to disruptions.
- **Collaboration Technologies:** Technology can be leveraged to enhance stakeholder communication and collaboration across the supply chain.

Value of Software Supply Chain Risk Management

- **Disruption Reduction:** Effective supply chain risk management enables organisations to anticipate potential disruptions, reduce downtime, and sustain operational continuity.
- **Cost Efficiency:** By reducing exposure to risk, organisations can prevent costly disruptions and financial losses, thereby improving overall operational efficiency.
- **Enhanced Resilience:** Organisations that implement robust supply chain risk management practices are better equipped to adapt to changes within the evolving supply chain environment.
- **Enhanced Decision-Making:** A thorough understanding of potential risks supports informed decision-making and more effective strategic planning.

Recommended Conclusion



In a highly interconnected environment, effective supply chain risk management is essential for maintaining competitive advantage and operational resilience. Proactive risk management enables organisations to navigate uncertainty and strengthen overall supply chain performance.

Task Three Part One

Introduction to Dev Operations

Several key factors must be considered when evaluating the application of DevOps within the software development process. The following discussion examines how DevOps principles influence application development practices.

Continuous Integration and Continuous Deployment (CI/CD)

Efficiency:

Continuous integration and continuous deployment (CI/CD) pipelines are a core DevOps practice that enable frequent integration of code changes and rapid delivery of updates. By automating testing and deployment, CI/CD pipelines accelerate feature releases and bug fixes, reduce human error, and shorten development cycles. Automating repetitive tasks such as code builds, quality checks, and deployments improves productivity and allows developers to focus on higher-value work. Continuous feedback from automated testing also helps maintain code quality, enabling teams to respond quickly to user requirements, resolve issues, and remain aligned with evolving project and market demands.

Rapid Feedback:

CI/CD provides developers with timely feedback on code quality after each commit, allowing issues to be identified and resolved early. This feedback loop improves code quality, reduces downtime, and lowers the cost of reversing changes by detecting problems before they escalate. Automated testing ensures updates meet quality standards before deployment, while rapid feedback supports collaboration and enables teams to enhance features with confidence, resulting in a more stable and reliable product. [5] [6] [7].

Collaboration and Information Sharing

Interdisciplinary Collaboration:

DevOps supports collaboration between development, operations, and stakeholders., fostering a culture of shared goals and responsibility. By breaking down traditional silos, teams gain a clearer understanding of each other's challenges, improving productivity and supporting a more unified approach to delivery. Cross-functional collaboration enables issues to be addressed from multiple perspectives, leading to faster problem resolution, increased innovation, and the efficient delivery of high-quality software. Overall, this collaboration enhances project efficiency and supports the development of resilient and adaptable systems. [7].

Shared Responsibility:

In a DevOps environment, shared responsibility for application performance and availability between development and operations teams fosters ownership and accountability. Involving both groups in monitoring and maintenance encourages a broader commitment to product reliability beyond individual roles. Developers are motivated to produce high-quality code, while operations teams collaborate closely to identify issues and improve system resilience, reinforcing a collective focus on uptime. This shared accountability strengthens collaboration, reduces blame-driven practices, and results in more reliable, well-maintained systems, while also promoting trust and proactive problem-solving across teams. [5] [6] [8].

1.) Automation Practices

Infrastructure as Code (IaC):

DevOps supports the automation of infrastructure provisioning, configuration, and management through Infrastructure as Code (IaC). By treating infrastructure definitions like application source code, teams can store, version, and track changes, making audits and rollbacks easier. IaC reduces configuration drift and improves reliability by enabling consistent environments across development, testing, and production, supporting scalable resource use to be provisioned or adjusted efficiently in response to changing demands. By making infrastructure configurations transparent and shareable, IaC enhances collaboration and promotes standardisation, faster deployments, fewer manual errors, and more agile DevOps workflows [5] [7].

Automated Testing:

Automated unit, integration, and functional testing throughout the development lifecycle enables early issue detection and consistent maintenance of code quality. When integrated into CI/CD pipelines, these tests validate code changes automatically, identifying defects before deployment. Automated testing reduces the time and cost of fault detection compared to manual methods and provides rapid feedback, allowing developers to implement changes confidently without introducing regressions. This approach also supports scalability, as tests can be repeatedly executed as the application grows, resulting in more stable and reliable software releases. [5] [7] [8].

1. Scalability and Reliability

Scalable Architecture:

DevOps supports the adoption of microservices architectures, enhancing system scalability, flexibility, and maintainability. By decomposing applications into smaller, independently deployable services, teams can modify individual components without affecting the entire system. This modular design enables rapid development, testing, and deployment while minimising downtime. Microservices also improve performance and resource efficiency by allowing services to scale independently in response to demand. As systems grow in complexity, this scalability enables teams to respond more effectively to evolving user requirements and organisational goals, strengthening long-term resilience and adaptability [6] [8].

Monitoring and Incident Response :

Core DevOps practices such as continuous monitoring, logging, and alerting are essential for maintaining application performance and reliability. Proactive monitoring enables teams to track system health, application metrics, and user behaviour in real time, allowing potential issues to be identified before they escalate. Automated alerts support rapid incident response, minimising downtime, improving user satisfaction, and supporting troubleshooting through logging. From incidents and continuously improve system performance. Together, these practices enhance reliability, promote accountability, and support high service availability and a consistent user experience. [7].

2. Agility and Flexibility

Adaptability to Change:

DevOps supports Agile development by enabling frequent adaptation and rapid responses to changing requirements. By fostering a collaborative culture, teams are encouraged to treat change as an integral part of development rather than an obstacle. This adaptability allows organisations to respond efficiently to customer feedback and evolving market demands while remaining aligned with user and business objectives. Continuous integration and deployment enable rapid, incremental updates with minimal disruption, while cross-functional collaboration ensures diverse perspectives inform decision-making. Together, these practices improve customer satisfaction, support competitiveness, and enable teams to deliver high-quality solutions that meet changing user needs [9].

Reduced Delivery Time:

The integration of automation, collaboration, and CI/CD practices significantly accelerates application development and delivery, reducing time-to-market. Automated testing and deployment allow teams to prioritise software quality rather than manual processes. CI/CD pipelines enable frequent, incremental releases and rapid feedback cycles, supporting continuous product improvement. In addition, close collaboration between cross-functional teams improves decision-making, streamlines workflows, and ensures alignment with shared objectives, providing organisations with a competitive advantage in responding to changing market demands [6].

3. Cost Optimisation

Reduced System Downtime:

Frequent software releases supported by automated testing and continuous monitoring reduce production issues, lowering maintenance costs and system downtime. A continuous delivery approach enables updates to be deployed in small, controlled increments, making issues easier to identify and resolve early. Automated testing ensures changes meet quality standards and do not disrupt existing functionality, while proactive monitoring allows performance anomalies to be detected and addressed promptly. Together with rapid incident response, these practices minimise user impact, maintain system stability, and allow resources to be redirected towards innovation and feature development, contributing to a more resilient operational environment [7].

Optimal Resource Utilisation:

Organisations can improve resource efficiency and reduce operational costs by minimising waste through Infrastructure as Code (IaC) and optimised CI/CD pipelines. IaC enables infrastructure to be defined and managed programmatically, allowing resources to be provisioned and scaled dynamically in response to demand, reducing over-provisioning and unused capacity. CI/CD pipelines further streamline development and deployment by automating repetitive tasks and removing bottlenecks, enabling teams to focus on higher-value work. Integrated monitoring tools provide insight into resource usage, allowing inefficiencies to be identified proactively. Together, these practices lower operational costs, enhance system performance and agility, and support a more sustainable operational model [8].

4. Security (DevSecOps)

Shift-Left Security:

DevOps has reshaped how security is approached in software development through a shift-left strategy that embeds security early in the development lifecycle. By integrating security tools into CI/CD pipelines, teams can identify and address vulnerabilities early rather than treating security as a post-development activity. Automated security testing techniques, such as static application security testing (SAST) and dynamic application security testing (DAST), provide rapid feedback and reduce the cost and complexity of fixing issues after deployment. In addition, promoting security awareness and targeted training helps establish a security-focused culture that encourages secure coding practices. Overall, the shift-left model embeds security as a core component of development, resulting in more trustworthy systems and greater stakeholder confidence [7].

Compliance Automation:

Compliance automation plays an important role in DevOps by ensuring applications meet security and regulatory requirements without slowing development. By embedding automated compliance checks into CI/CD pipelines, teams can continuously verify that code and system configurations conform to required standards throughout the development lifecycle. This reduces the risk of non-compliance by enabling real-time assessment against predefined regulatory criteria. Automating compliance activities also minimises reliance on manual audits, allowing developers to focus on delivering high-quality software while using organisational resources more efficiently. Automated reporting further streamlines audits by generating compliance documentation and providing clear visibility of compliance status. Overall, compliance automation improves operational efficiency, reduces regulatory risk, and reinforces shared responsibility for compliance across the organisation. [6].

5. Challenges and Considerations

Culture Shift:

The cultural transformation required to successfully adopt a DevOps strategy can present significant challenges for teams accustomed to traditional development models. This transition demands a fundamental change in mindset, placing greater emphasis on collaboration, flexibility, and continuous improvement across all levels of the organisation. To support this shift, organisations must invest in training programmes that enhance not only technical skills but also a deeper understanding of DevOps principles and practices. Such initiatives should aim to break down silos between development, operations, and other stakeholders by promoting shared responsibility and collective ownership. Establishing cross-functional teams that combine diverse skill sets further supports the development of a collaborative culture, improving communication and coordination. Leadership also plays a pivotal role in driving cultural change, as visible commitment to DevOps values can motivate and guide teams in adopting new ways of working. By fostering an environment that values transparency, innovation, and resilience, organisations can

overcome resistance to change and achieve more effective DevOps adoption. Although challenging, this cultural shift is essential for maximising the benefits of DevOps and ensuring long-term success in a highly competitive environment [6] [7]

Toolchain Complexity:

Managing multiple DevOps tools across the delivery pipeline can introduce complexity for organisations a complex challenge. When these tools are not effectively coordinated, the resulting complexity can lead to miscommunication, operational inefficiencies, and reduced productivity. Addressing this challenge requires a well-defined strategy to ensure consistent adoption and effective use of tools across all teams. This involves establishing clear guidelines and best practices for tool selection, integration, and usage, as well as promoting collaboration to facilitate the sharing of knowledge and experience. Organisations should also invest in comprehensive training programmes to help team members develop proficiency with selected tools and fully understand their capabilities and benefits. Regular review and refinement of the DevOps toolchain can further enhance efficiency by streamlining workflows and identifying unnecessary overlap. By prioritising strategic toolchain management, organisations can reduce complexity, strengthen collaboration, and maximise the value of their DevOps initiatives, ultimately leading to more efficient and successful development processes [5] [10].

Conclusion to IDO:

DevOps plays a central role in modern software development by enabling faster, more reliable, and more efficient application delivery. It is particularly well suited to Agile environments, supporting scalability, security, enhanced collaboration, and the automation of essential processes. However, to fully realise the advantages of DevOps adoption, organisations must also address associated cultural shifts and the complexities of managing diverse toolchains [11].

Task Three Part Two

Assessing DevOps in the Development of a Library Web Application

The application functions as a digital library, enabling users to access and interact with the library catalogue:

- **Borrow and Reserve Books:** The system allows catalogue searches, availability checks, and item borrowing.
- **Automated Notifications:** Users receive automated alerts for book availability, overdue status, and reservation changes.
- **Administrator Panel:** Library staff are responsible for managing book inventory, updating availability status, and handling reservations and returns.
- **Book Information Database:** The database stores key information such as titles, authors, genres, availability status, and ISBNs [12].

Continuous Integration and Continuous Deployment (CI/CD)

- **Regular Updates:** The library application benefits from DevOps practices by enabling regular updates to the database, administrative tools, and user interface through automated testing and deployment pipelines [5].
- **Fast Feature Rollout:** DevOps allows these updates to be delivered rapidly in small, incremental releases, reducing downtime and minimising disruption for users. Such enhancements may include adding new book genres, improving search functionality, or refining the administrative interface to support more effective system management [13].
- **Purpose:** The purpose of this online application is to simplify the management of library resources for both users and administrators. It improves operational efficiency by providing an intuitive interface for discovering and reserving books, alongside backend functionality that supports effective inventory management [8].

Web application and its purpose

Architecture Diagram:

Components:

- The user interface (UI) serves as the front-end component of the system, allowing users to search for and interact with book listings.
- *API Layer:* It handles requests from the user interface by managing reservations and retrieving book-related data from the backend.
- *Database:* A database is used to store book-related information, including titles, authors, and availability status.
- *Admin Panel:* A backend administration platform enables library staff to add, update, or remove book records, as well as manage reservations.
- *Notification Service:* The system automatically sends notifications to users regarding overdue items, reservations, and book availability via email or SMS.
The authentication service manages user and administrator logins, ensuring secure access to the application

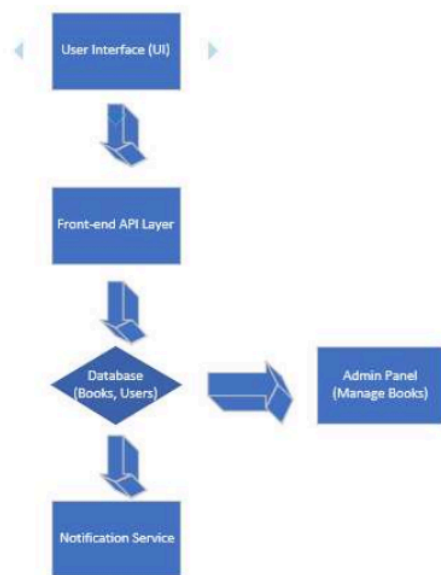


Figure 12 Architecture of a Library System

Software DevOps Principles Analysis

Continuous Integration (CI):

- **Regular Integration Practices:** Continuous Integration allows frequent code integration across application components, reducing conflicts and integration issues.
- **Test Automation:** Continuous Integration (CI) ensures that all code changes are subjected to automated testing before being merged into the main codebase. This early detection of issues improves overall code quality, particularly when modifying critical components such as the database or user authentication services. [5] [8].

Continuous Delivery (CD):

- **Accelerated Update Cycles:** Continuous Deployment (CD) enables application updates to be released automatically, including enhancements to the notification system and the introduction of new administrative features. This ensures that both users and administrators receive the latest functionality without manual intervention.
- **Rollback Capabilities:** Continuous delivery pipelines support rapid rollback to a stable release if a newly introduced feature or bug fix causes issues, such as faults within the borrowing system. [8].

Automation:

- **Automated Infrastructure Setup:** Tools such as Infrastructure as Code (IaC) enable automated provisioning of databases, servers, and environments, ensuring consistency across development, testing, and production stages. This consistency helps prevent deployment issues caused by differences between environments.
- **Monitoring and Logging:** DevOps strongly emphasises real-time monitoring of system performance to detect and address bottlenecks such as slow database query response times before they negatively affect users. [8].

Agile Support:

- **Iterative Development:** By integrating Agile methodologies with DevOps practices, the library application can be developed iteratively. Ongoing feedback can then be used to continuously refine features such as book management and user notification services. [5].
- **Ongoing Feature Releases:** CI/CD pipelines minimise delays between feature updates and enhancements to existing system functionality. [9].

Identify potential challenges

Managing the Complexity of Continuous Integration and Continuous Deployment Pipelines

Integrating all system components including the user interface, administrative panel, database, and notification services often requires a highly sophisticated CI/CD pipeline configuration. This complexity demands careful planning of build processes, deployment workflows, and comprehensive test cases. [8].

1.) Database Management:

Maintaining a consistent shared database across development, testing, and production environments can be challenging. Changes to the database schema such as introducing additional fields for book records must be implemented with care to avoid service disruption or data loss.

2.) Security:

Ensuring the protection of user data throughout integration and deployment processes, as well as maintaining secure access to the administrative panel so that only authorised personnel can use it, can be challenging. Addressing these concerns requires a DevOps pipeline that places strong emphasis on security controls and safeguards.

3.) Legacy Systems:

Adopting DevOps practices can be particularly difficult when the library relies on legacy technologies. Migrating existing systems and data into a modern DevOps environment may require significant effort and careful planning.

4.) Cultural Shift:

Implementing DevOps can be difficult when development and operations teams are unaccustomed to working collaboratively, as it requires a significant cultural shift towards shared responsibility and teamwork.

Challenges and improve quality of the web application

1.) CI/CD Pipeline Complexity:

Solution: DevOps tools such as Jenkins, CircleCI, and GitLab CI enable build, testing, and deployment processes to be automated in a structured, step-by-step manner. With careful design, these pipelines can independently manage multiple system components by automating tasks such as deployments, database migrations, and testing activities, thereby reducing the need for manual intervention.

Benefit: Automation helps ensure that routine updates do not introduce new defects, thereby improving the overall quality and stability of the application.

2.) Database Management:

Solution: When database migration tools are employed, schema changes can be version-controlled, tested, and deployed consistently across all environments. This approach reduces the risk of data corruption and simplifies rollback procedures if issues arise during deployment [5].

Benefit: Consistent database management enhances system reliability by ensuring data integrity and minimising downtime.

3.) Security:

Solution: DevSecOps practices can be embedded within the CI/CD pipeline to automate security testing across the development process. This includes securing user data, identifying vulnerabilities in the administrative interface, and applying encryption where appropriate.

Benefit: Incorporating security measures early in the development lifecycle enables vulnerabilities to be identified at an early stage, thereby improving the overall security of the application.

4.) Legacy Systems:

Solution: DevOps can support a smooth transition from legacy systems by gradually introducing automation, beginning with areas such as testing and infrastructure management. As teams become more familiar with DevOps practices, the system can be fully migrated in a controlled and phased manner.

Benefit: This approach reduces the risk of disruption by modernising the application's infrastructure [5].

5.) Cultural Shift:

Solution: One effective way to address this gap is by providing training and actively promoting collaboration between development and operations teams. Further alignment of objectives can be achieved through the use of collaborative tools such as shared documentation platforms, project tracking systems, and team communication channels.

Benefit: A strong DevOps culture enhances team productivity, improves communication, and leads to more reliable software alongside faster issue resolution.

Postman

Postman is an API development platform that enables the design, testing, automation, and documentation of application programming interfaces, supporting efficient and reliable software development practices [14].

What Postman is used for:

1. API testing

Postman lets users send HTTP requests (GET, POST, PUT, DELETE, etc.) to an API and immediately view responses such as status codes, headers, and data payloads. This helps verify that APIs behave as expected [14].

2. Automated testing

Tests can be written within Postman to automatically check responses (for example, confirming correct status codes or returned data). These tests can be reused and automated as part of development workflows.

3. API development & debugging

Developers use Postman to experiment with API endpoints, debug issues, and validate changes before integrating them into applications [14].

4. Collections and environments

- a.) Collections group related API requests together (e.g. all endpoints for a library system).
- b.) Environments store variables such as URLs, tokens, or credentials, allowing the same tests to run in development, testing, and production environments.

5. Documentation

Postman can automatically generate clear, human-readable API documentation directly from collections, helping teams and stakeholders understand how an API works [14].

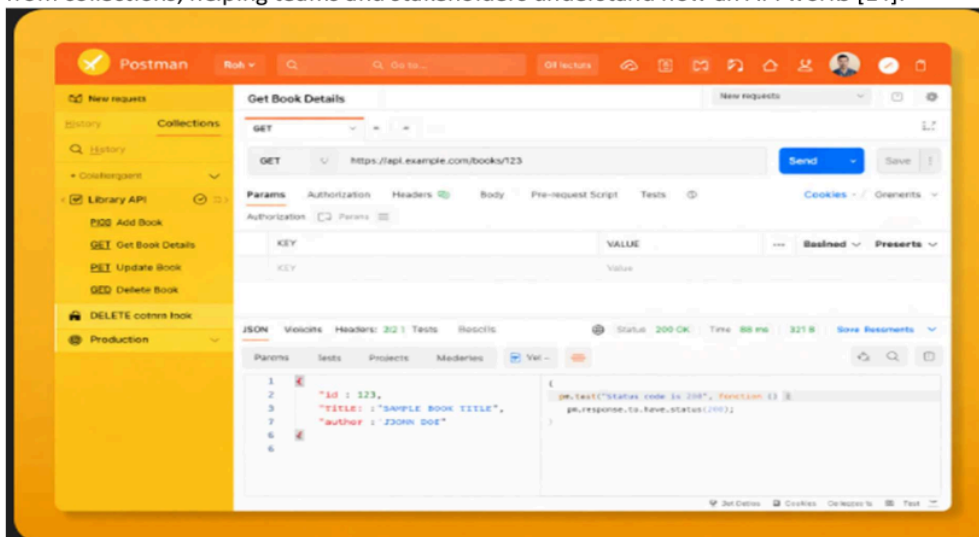


Figure 13 Postman with API with GET request

Selenium

Selenium is an open-source tool used for automating web browsers. It is mainly used for testing web applications, particularly the user interface (UI). Selenium is commonly used to validate user interactions, ensure cross-browser compatibility, and support automated regression testing within continuous integration workflows [15].

- Automatically test web pages (buttons, forms, links, logins)
- Check that UI changes don't break existing functionality
- Run regression tests after updates
- Test applications across different browsers (Chrome, Firefox, Edge, etc.)

Selenium in DevOps & CI/CD

- Selenium tests can be triggered in a CI/CD pipeline
- Tests run automatically after each code change
- If a test fails, deployment can be stopped

This helps ensure:

- UI features still work after updates
- Bugs are caught before users see them

Bibliography

- [1] N. Friedman, "time.com," TIME USA, LLC, 2024 . [Online]. Available: <https://time.com/>. [Accessed 2024 10 31].
- [2] T. Preston-Werner, "forbes.com," Forbes Media LLC., 2024. [Online]. Available: <https://www.forbes.com/>. [Accessed 2024 10 31].
- [3] C. Wanstrath, "forbes.com," Forbes Media LLC, 2024 . [Online]. Available: <https://www.forbes.com/>. [Accessed 2024 10 31].
- [4] S. Chacon, "scottchacon.com," [Online]. Available: <https://scottchacon.com/>. [Accessed 2024 10 31].
- [5] G. Kim, The Phoenix Project : A Novel about IT, DevOps, and Helping Your Business Win, George Spafford, 2015.
- [6] J. H. a. D. Farley, Continuous Delivery, Addison-Wesly, 2010.
- [7] P. Debois, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology, IT Revoulution, 2016.
- [8] N. Forsgren, Accelerate: The Science of DevOps, portland: IT Revolution.
- [9] L. Fong-Jones, "honeycomb.io," Hound Technology, Inc, 2024 . [Online]. Available: <https://www.honeycomb.io/>. [Accessed 29 10 2024].
- [10] D. Edwards, "devopscon.io," S&S Media, 2024 . [Online]. Available: <https://devopscon.io/>. [Accessed 29 10 2024].
- [11] K. Hightower, "theregister.com," <https://situationpublishing.com/>, 1998–2024. [Online]. Available: <https://www.theregister.com/>. [Accessed 29 10 2024].
- [12] C. Majors, "agilemanchester," Software Acumen LTD, 2013-2024 . [Online]. Available: <https://agilemanchester.net/>. [Accessed 29 10 2024].
- [13] A. Glover, "kb.founderculture.net," founderculture, [Online]. Available: <https://kb.founderculture.net/>. [Accessed 29 10 2024].
- [14] D. Westerveld, API Testing and Development with Postman, Birmingham, UK: Packt Publishing, 2021.
- [15] S. Project, "SeleniumHQ," 2024. [Online]. Available: <https://www.selenium.dev/documentation/>. . [Accessed 8 1 2026].

FINAL GRADE

33 / 100

GENERAL COMMENTS

This submission demonstrates understanding of DevOps principles, CI/CD practices, and collaborative development tools, with reasonable coverage of concepts such as automation, monitoring, and continuous delivery.

However, the work is significantly undermined by several critical omissions and organisational issues. Most notably, Task Two (Estimation and Burndown Charts) is entirely absent, representing a loss of 20 potential marks.

The Agile and Scrum content required in Task One is inadequately addressed; the four Agile concepts (multiple short iterations, face-to-face communication, adaptability, and quality ownership) are not systematically explained, and the Scrum methodology section contains misplaced content about supply chain risk management rather than the required material on user story flow and story mapping.

The SwDevOps section provides reasonable conceptual coverage of the library application and testing tools, but fails to include the required GitHub Actions workflow file examples and practical code demonstrations for Postman and Selenium.

The bibliography lacks academic journal articles and contains inconsistent referencing with several incomplete citations and somewhat irrelevant references.

Access calls from 2024?

The report would benefit substantially from a clear introduction, proper figure numbering with in-text references, and removal or relocation of irrelevant content such as the Space Invaders game classes and supply chain material.

Marks by Section

Agile Concepts and Scrum Methodology

Section 1 Engagement with Literature 5 out of 16

Section 1 Knowledge and Understanding 6 out of 16

Section 1 Cognitive and Intellectual Skills 5 out of 8

Section 1 Total 16 out of 40

Estimation and Burndown Charts

Section 2 Engagement with Literature 0 out of 8

Section 2 Knowledge and Understanding 0 out of 8

Section 2 Cognitive and Intellectual Skills 0 out of 4

Section 2 Total 0 out of 20 -Completely missing

SwDevOps Evaluation

Section 3 Engagement with Literature 4 out of 16

Section 3 Knowledge and Understanding 8 out of 16

Section 3 Cognitive and Intellectual Skills 5 out of 8

Section 3 Total 17 out of 40

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12



Comment 1

no comment

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24
