

UNIVERSITY OF WALES TRINITY SAINT DAVID(UWTSD)

Assignment

Md.Nabil Rahman

ID: 2504501

Course Code: ACCA7023

Module: Agile Development and Software Dev Ops

Lecturer: Nathan Morgan

MSc Software Engineering and Artificial Intelligence

January 9th 2026

Abstract

Agile software development has become a dominating mechanism of dealing with complexity as well as change in the current software development. Agile concepts are implemented through such frameworks as Scrum which offer well-organized practices of iterative delivery, constant feedback, and tight cooperation among stakeholders. Simultaneously, Software Development and Operations (SwDevOps) methods have enabled companies to deliver reliable software at decreasing timelines with shorter periods because of automation, unceasing testing, and refined deployment pipelines.

The present report is a critical analysis of the implementation of four basic concepts of Agile i.e. multiple short iterations, focus on face-to-face communication, responsiveness to changing requirements and quality ownership across the product lifecycle as embodied in the Scrum methodology. The flow of user stories is particularly given special consideration, as it offers a user-friendly approach to the prioritization of features and traceability. A web application example is used to illustrate the SwDevOps practices, that is, the continuous integration and deployments with the help of GitHub Actions, API testing with the help of Postman, and user interface testing with Selenium. The paper also highlights how user story mapping works in characterizing iterations and sprint planning, as well as estimating methods with the help of burndown charts to monitor progress. The report critically analyzes the benefits of integrating Scrum with SwDevOps such as enhancing flexibility, accelerated delivery process, enhanced quality of software and increased collaboration among the team members. Simultaneously, it also recognizes the constraints of its nature, including the requirement to have an organization prepared to undertake the process, cultural fit, technical savvy, and a strict adherence to Agile principles. Finally, the research shows that Scrum and SwDevOps represent an effective tool to handle complexity and uncertainty but their effectiveness relies on a balanced approach between the rigor and flexibility of the process, effective communication, and learning in the course of development teams. Moreover, the combination of visual aids, automated processes, and continuous planning will strengthen transparency and responsibility so that team productivity and product deliverables become maximized.

Contents

Abstract	i
List of Figures	iii
1 Introduction	1
2 Overview of Agile and Scrum Methodology	1
3 Critical Evaluation of Core Agile Concepts within Scrum	2
3.1 Multiple Short Iterations	2
3.2 Face-to-Face Communication and Collaboration	2
3.3 Adaptability to Changing Requirements	4
3.4 Quality of Ownership Throughout the Product Lifecycle	4
4 Flow of User Stories in Scrum	5
4.1 User Story Mapping and Iteration Planning	6
5 Estimating Story Implementation Time and Difficulty	7
5.1 Purpose of the Burndown Chart	7
6 Web Application Case Study Overview	8
6.1 Software Components Requiring Testing	8
6.2 GitHub Actions for CI/CD in SwDevOps	9
7 API Testing Using Postman	9
7.1 User Interface Testing Using Selenium	10
8 Benefits and Limitations of SwDevOps Adoption	11
8.1 Benefits of Adopting SwDevOps	11
8.2 Limitations of SwDevOps Adoption	12
9 Conclusion	13
10 REFERENCES	14

List of Figures

3.1	Scrum Sprint Cycle Diagram	3
3.2	Adaptability to Changing Requirements	5
4.3	Flow of User Stories	6
5.4	Sprint Burndown Chart	7
6.5	CI/CD Pipeline Using GitHub Actions	10
7.6	Example Postman Test	10

1 Introduction

The Agile software development paradigm has become the leading approach of providing complex software systems in settings that are typical of uncertainty, rapid evolution, and continually changing stakeholder expectations [1]. Conventional software creation models like waterfall model fail to work well in such circumstances because they have a strict, step-by-step format and also because of the upfront design choices that are only confirmed as true at the latter phases of the development procedure. Such inflexibility usually leads to a slow feedback cycle, higher risk and a decreased fit between what the user needs.

Agile practices, and Scrum specifically, aim to overcome these shortcomings through encouraging iterative and incremental development, high-frequency feedback and close team and stakeholder interactions and shared ownership of the quality of the end product. Scrum offers an organized but flexible process that can help teams effectively respond to change without being deprived of transparency and accountability across the development lifecycle.

This report is a critical analysis of four Agile principles in implementation in the Scrum framework. These values consist of short, time-boxed iterations to provide incremental value; the belief in face-to-face communication as a means to get more people to collaborate and understand each other better; the capacity to adopt and embrace change even at the end of the development process; and shared ownership of quality through continuous testing and improvement. Special focus is put on user stories and user story mapping as fundamental methods of requirements capture, functionality prioritization, and user centric definition of sprint iterations.

2 Overview of Agile and Scrum Methodology

Agile development is based on the manifesto of Agile which contains concepts that highlight the people and relationships over the processes and tools, working software over full documentations, collaboration with customers over contract negotiation and responding to change instead of sticking to a predetermined plan [1]. Agile does not dictate certain procedures, but the philosophy is one that embraces flexibility, lifelong learning, and customer value delivery. This style is also very suitable when the requirements are dynamic and when they are changeable.

Scrum is one of the most popular Agile frameworks; it is structured and flexible, yet it is applied to practice a lot in this method of applying Agile principles [2]. Scrum offers a well defined structure which allows teams to be agile and cope with hustle and bustle. It does that by establishing a set of roles, events, and artifacts that would inform the development process but not limit creativity and innovation. Sprint Planning helps to create a common vision of the goals and priorities and

the Daily Scrum is aimed at encouraging continuous communication and coordination of the team. Sprint Review allows the stakeholders to check the increment and give feedback, and the Sprint Retrospective is aimed at the reflection and the ability to find the ways to be improved.

The framework also establishes three major roles as follows: Product Owner: who maximizes product value and manages the Product Backlog; Scrum Master: who facilitates the Scrum process and eliminates impediments; and Development Team: who delivers a potentially shippable product increment every sprint. In line with these functions are Scrum artifacts like Product Backlog, Sprint Backlog and Increment that give visibility on the progress of work and the evolution of the product.

3 Critical Evaluation of Core Agile Concepts within Scrum

Critical Assessment of the fundamental principles of Agile in Scrum. Various diagrams have been referred to in this section including sprint life cycle models, user story maps, burndown charts generated throughout in-class group work. Such artefacts are vital to visualising the Scrum practices and critical evaluation (see Fig. 1–Fig. 3).

3.1 Multiple Short Iterations

Scrum works with short iterations operationalised by time-boxed sprints, which last one to four weeks [2]. The goals of each sprint are a potentially shippable piece of software. This will be less risky because it will enable the inspection of progress to be done frequently and possible problems will be detected at the initial stages.

A short iteration is more predictable and more likely to make the stakeholders confident, as the progress is evident at regular steps. When examined critically, this routine inspection identifies inefficiencies early on, but it necessitates strict backlog refinement to avoid piecemeal delivery. Nevertheless, too short sprints can lead to high overhead costs as there is a large number of planning and reviewing the meetings. As such, sprint length should be well balanced in order to achieve maximum value delivery.

3.2 Face-to-Face Communication and Collaboration

Face-to-face communication is a significant element of Scrum that the framework focuses on as an important tool of collaboration and effective information exchange. The daily stand-up meetings, sprint planning, sprint review, and retrospectives (see Fig. 3.1) are all aimed at promoting the face-to-face communication between team members and stakeholders. Face-to-face communication enhances understanding, minimizes the use of a lot of documentation, enables quick decision making

SCRUM PROCESS

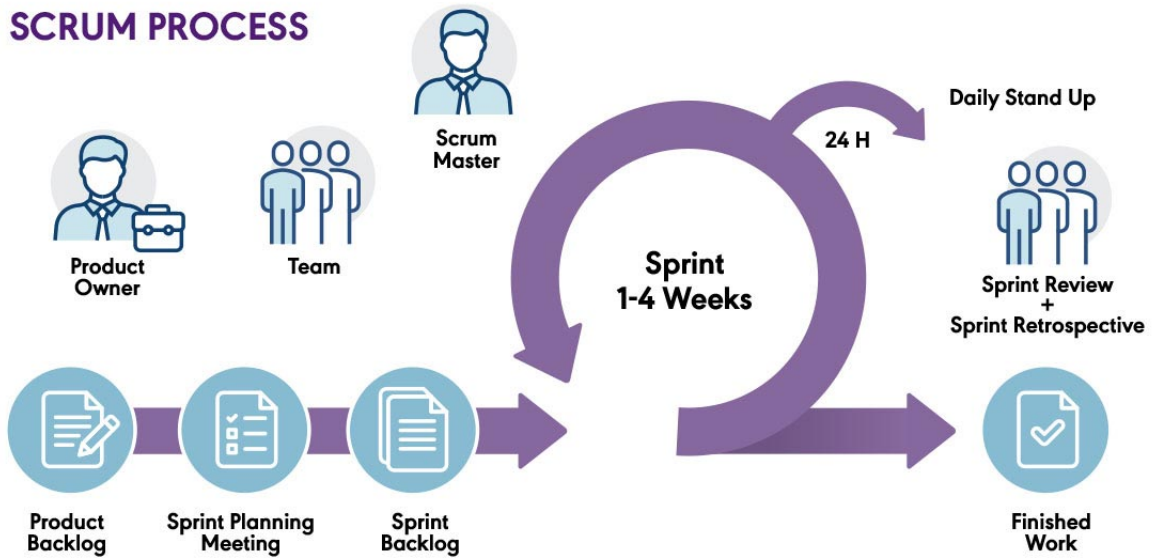


Figure 3.1: Scrum Sprint Cycle Diagram

and also allows issues to be resolved fast. Ensuring that feedback is immediate, these interactions will assist the teams in keeping track of the goals, clarifying the requirements, and responding to the challenges before they get out of control.

In remote or distributed teams, however, this principle is frequently opposed because co-location cannot be done. Video conferencing, messaging tools and collaborative tools should be applied in such instances to recreate the power of face-to-face communication. Although the tools can facilitate a synchronous mode of communication and an exchange of information, they might be ineffective in capturing the subtle non-verbal behaviors like body language, gestures, and tone which are significant in the development of trust, team spirit, and understanding. This weakness may occasionally result in confusion, lack of involvement or poor team dynamics.

Most importantly, effective communication and collaboration in Scrum depends upon the tools applied, as much as discipline within the team, active participation, and culture of openness and transparency. In distributed environments, teams can also have effective teamwork through using video calls alongside collaborative documentation, shared dashboards, and effective communication protocols. Finally, the focus on direct communication in Scrum helps to develop closer relationships among team members, quicker problem-solving, and supportiveness to change, still, it needs a thorough adjustment in the situation when teams are not co-located.

3.3 Adaptability to Changing Requirements

Scrum is also created to adapt to change hence it is specifically suited in dynamic environments where requirements are constantly changing. In contrast to the traditional development models, which fix the requirements at the beginning of the process, Scrum enables the refinement and re-prioritization of backlog requirements at any stage in the product life process. This flexibility is usually manifested in the changing user story maps which are created during backlog planning and refinement steps (see Fig. 3.2) [2]. User story mapping allows the teams to imagine functionality using the perspective of the user and re-prioritize in accordance with new knowledge or based on the feedback of stakeholders or other changes in the market.

Any new or changed requirements may be added to the Product Backlog at any given time and planned to be included in future sprints. This makes sure that the development processes do not go out of track that is in line with present business requirements and customer expectations, although these evolve as time goes by. Scrum facilitates incorporation of feedback through both sprint review and real-life application to allow the teams to keep on refining the direction of the product and delivering the most valuable features first.

This freedom of action gives it a great competitive edge since the organizations can react quickly to the new opportunities, new regulations or demands of the users. But there is risk introduced with such receptiveness to change. Frequent change is likely to cause scope creep, lack of focus and instability in a development team unless there is a high level of Product Owner discipline and clear priorities criteria. Coherent product vision thus demands proper management of the backlog, alignment of the stakeholders and communication.

3.4 Quality of Ownership Throughout the Product Lifecycle

In the Scrum method, quality is a collective effort of the whole development team as opposed to one testing or quality assurance phase. This principle of collective quality ownership ensures that the quality is designed into the product at all of the lifecycle stages of the development process rather than being tested or fixed at the end. Through its characterization of quality practices in every sprint, Scrum goes hand in hand with the Agile focus on providing potentially shippable increments with a certain frequency.

The major practices that assist in shared quality ownership are the Definition of Done (DoD), automated testing, and peer reviews. The Definition of Done creates a standardized and clear benchmark that has to be fulfilled before a product increment is qualified as being done. This usually covers criteria of success of unit and integration testing, code review, compliance to code standards, and updated documentation. Quality is further achieved through automated testing,

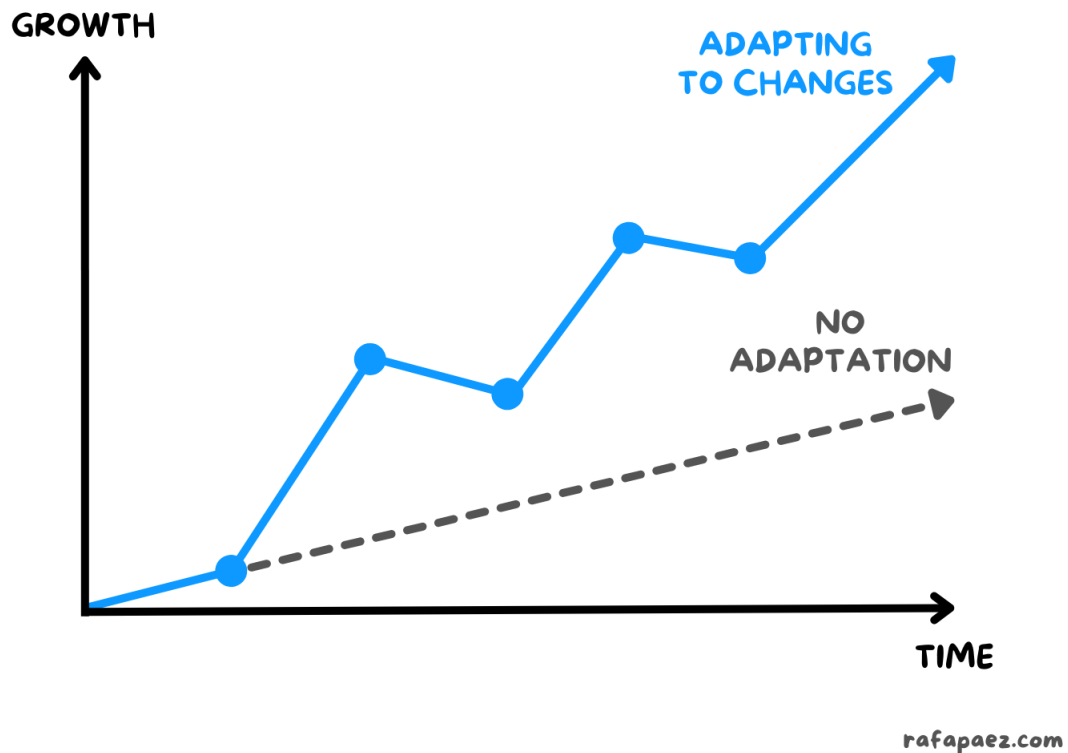


Figure 3.2: Adaptability to Changing Requirements

which allows the detection of defects at an early stage and provides quick feedback, whereas peer reviews encourages knowledge sharing, technical debt reduction, and code maintainability.

4 Flow of User Stories in Scrum

The user story flow begins with identification of the high level user activities that is better known as the backbone of user story map. Such processes are marked in the upper part of the map (Figure 4.3) as the group work-related activities and include the activities connected to forming group chats or becoming a part of them, sending messages, emojis, controlling the text, and updating the user profiles. User stories are further elaborated under each and every activity. Among the activities that involve messaging, it is possible to single out such personal stories as text chat, GIFs sharing, voice notes, declining messages, muting notifications, and blocking users. The user activity directly adds all the stories such that the requirement traceability to user goals can be offered.

In the story map, the user stories will be organized as functional areas which include:

- User (user status, friends list, presence online)
- Team Features (Adding and managing users)

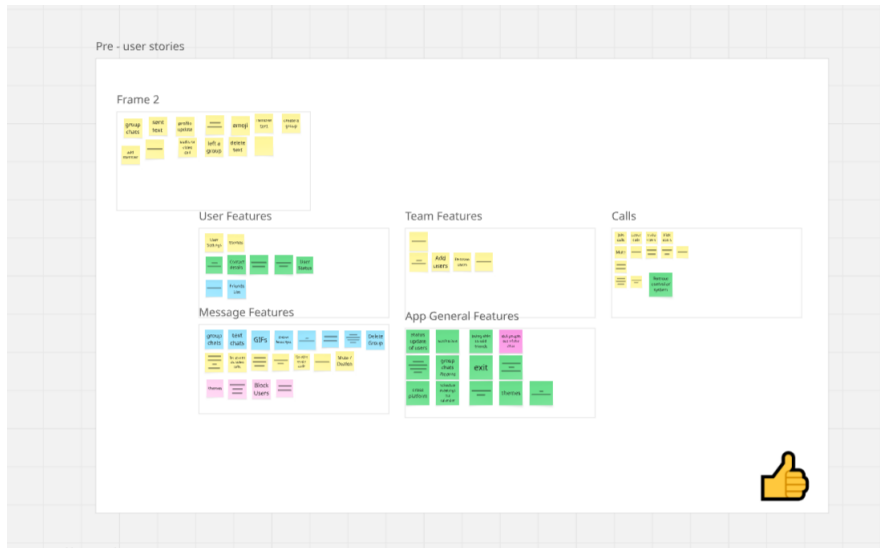


Figure 4.3: Flow of User Stories

- Message Functions (group chats, text, GIFs, voice notes, mute and block)
- General App Features (search, themes, exit, and profile updates)
- Included in the calls: (audio and video calling options)

This organization enhances understanding and promotes teamwork among the teams due to the ability of the teams to specialize in particular areas and yet understand the overall system.

4.1 User Story Mapping and Iteration Planning

The capability to facilitate the idea of iteration planning by tying it to vertical slicing is one of the main strengths of user story mapping. The horizontal axis is the user journey whereas the vertical axis is priority and delivery sequence.

The most important user stories are located at the top of the map in the class work. These narratives constitute Minimum Viable Product (MVP) and are applied during the initial version. As an example, the basic features of the system like the user checking in, formation or joining of groups, and sending simple text messages would be part of the first release.

The next versions will add more values by integrating less important yet superior functions, including emojis, GIFs, profile customization, and notification settings. The innovations, such as voice note, calling facility, theme, and higher user management, will be implemented in subsequent iterations. This methodology makes every release have a fully functional subset of functionality, not half-baked or unrelated functionality.

5 Estimating Story Implementation Time and Difficulty

Estimating the time to implement and difficulty of a story is supposed to forecast the amount of effort that needs to be done so that the work can be scheduled realistically within a sprint. Scrum generally uses story points in estimation, which are the relative effort, complexity, risk and uncertainty required to implement a user story, as opposed to the specific amount of time. Techniques used to support this process include Planning Poker. In Planning Poker, the team members will estimate a story independently and after that discuss the differences and in this way technical risks, dependencies, and assumptions can be uncovered. This is a collaborative method that enhances a mutual understanding and minimizes personal bias, and the estimates become more accurate as the time goes by. In Sprint A (Figure 5.4) the original scope was approximated 31 story points as it was reflected in the burndown chart.

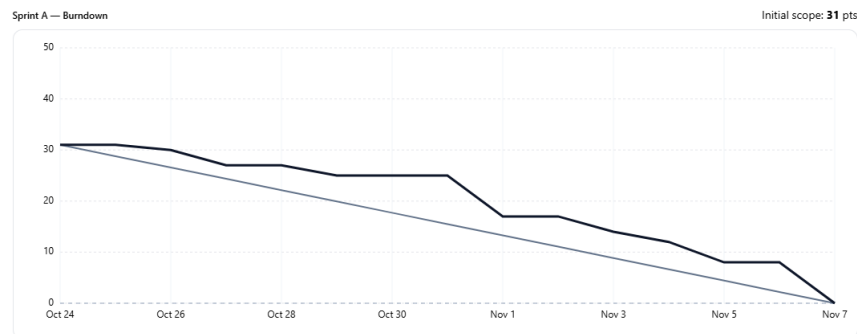


Figure 5.4: Sprint Burndown Chart

From the chart, we can observe that:

- The actual burndown line stays above the ideal burndown line for most of the sprint. This indicates that the team was progressing slower than originally estimated.
- There are flat sections (for example, around Oct 27–30 and Nov 5–6), meaning no story points were completed during those periods. This suggests that some stories were more complex or time-consuming than expected, or that tasks were not broken down into smaller, manageable stories.
- A sharp drop near the end of the sprint shows that several stories were completed at once, which often happens when stories are large or underestimated.

5.1 Purpose of the Burndown Chart

A basic artifact of Scrum is the burndown chart which is a visual representation of the remaining work in the course of a sprint [2]. Its main aim is to offer visibility to the sprint progress enabling

the development team as well as the stakeholders to be in a position to track the effectiveness of planned work in being completed. Replacing the points or tasks left on the plot on a daily basis, the chart will indicate the course of work completion and allow identifying the deviations at the intended course at the earliest stages.

A downward trend in the burndown chart indicates a progression that is always on track toward the sprint goals, a flat trend or positive trend could be an indicator of a possible problem with underestimated user stories, technical blockers, or change of scope. As a case in point, in activities during classes, a typical burndown chart (Figure 5.4) would demonstrate the ideal and the actual progress curve, which would help the teams compare the expected and actual performance. This type of visual feedback helps to implement timely interventions, including task reassignments, re-prioritization, or even more cooperation to ensure that sprint goals are achieved.

Although burndown charts are very useful in the process of monitoring progress and ensuring transparency within the team, they also have significant restrictions. They cannot be used to assess performance by individuals or by teams, since exorbitant use of chart measures can cause individuals to gaming-related behaviors to push work or estimates (or work that has been done-in-lie-of) to be reported as being on track. The chart therefore must be viewed as an inspection and adaptation aid, and not a numerical indicator of productivity. Burndown charts can be used to keep Scrum teams focused, increase their predictability, and make prompt and effective responses to emerging challenges, which serves to reinforce the iterative and collaborative aspect of Agile development.

6 Web Application Case Study Overview

In this paper, I shall examine a case study of Web Application. The case study chosen is an Todo web application which enables users to add, read, update task and delete them. High reliability, security and high performance are needed in such applications.

6.1 Software Components Requiring Testing

The software components that need to be tested include. The most notable major elements that need to be tested are the user interface, backend API, Application logic, database interactions and Component-to-Component Integration. Thorough testing delivers system soundness and functionality.

6.2 GitHub Actions for CI/CD in SwDevOps

GitHub Actions is an effective realization of pipeline that can be utilized to develop end-to-end automation to construct, test, and release softwares to support the traditions of constant integration (CI) and constant delivery (CD) [4]. By integrating GitHub Actions with the source code repository, developers may automate critical software delivery lifecycle activities, reducing human labor and improving product reliability. GitHub Actions can be used to support continuous integration when coupled with Software Development and Operations (SwDevOps) practices, where the changes are automatically validated as they are being added to the code base. A series of predefined workflow procedures are triggered by each code push or pull request, assembling the application, executing pre-established test suites, and achieving quality assurance. This will guarantee that the new code easily fits with the existing functionality and that bugs are also detected at an early stage before the code is taken to production conditions. Continuous deployment has also been facilitated through Github Actions which automates the deployment of code that has been successfully tested to be in the staging or production environment. Deployments are predictable and less likely to include human mistake when workflows are standardized and repeatable. Teams can deploy new features and updates more frequently and confidently thanks to this automation, which also makes environments more uniform.

Namely, GitHub Actions provides CI/CD by the following means:

- To provide developers with immediate feedback, tests are automatically run on every push or pull request.
- Early detection of errors and integration problems in the development process, as well as minimizing rework and the risk of deployment.
- Using version-controlled work flows to create consistent and repeatable build and deployment procedures is crucial.

In general, GitHub Actions used in a SwDevOps-enabled Scrum environment contribute to a more efficient development process, a better quality of the software delivered, and adherence to the concepts of Agility by providing quick feedback, a continuous improvement process, and more timely value delivery to the stakeholders.

7 API Testing Using Postman

By allowing developers to test endpoints, response formats, and error handling during automated quality assurance procedures, Postman facilitates API testing [5]. Regression coverage can be

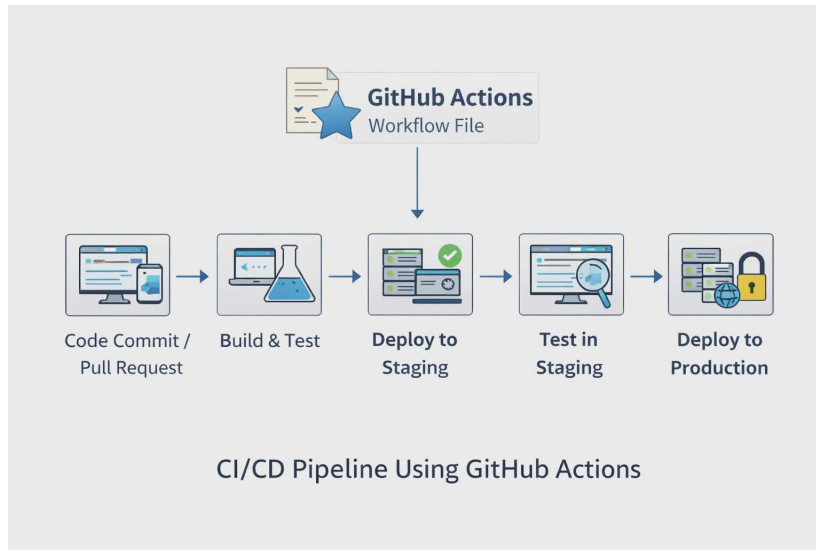


Figure 6.5: CI/CD Pipeline Using GitHub Actions

enhanced with automated collections using CI workflows. The Postman is employed to test the REST API endpoints of the To-Do application.

Postman allows testers to:

- Send requests to API endpoints
- Validate responses and status codes
- Automate regression tests using collections

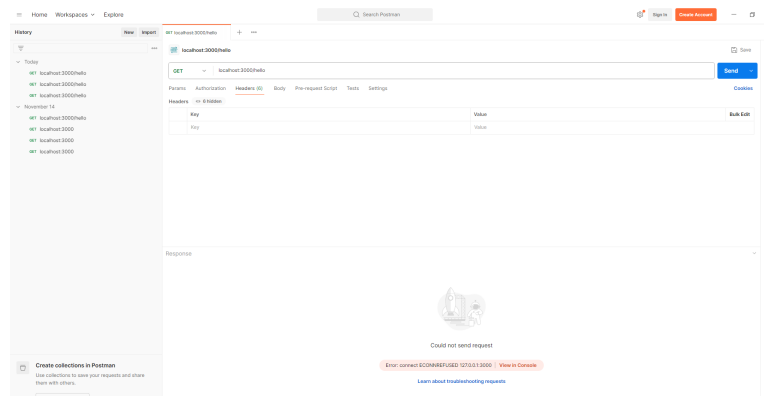


Figure 7.6: Example Postman Test

7.1 User Interface Testing Using Selenium

Selenium is an automation framework that is used to test the user interface (UI) in real web browsers. It allows testers to test user interaction by clicking buttons, filling out forms and moving between pages across a number of browsers and platforms to assure consistent behavior of the UI

in different execution environments [6]. The ability to communicate with the application through interaction Selenium will give a high degree of confidence that the important user facing functionality will behave as desired. However, compared to unit tests or API-level testing, UI tests are typically slower, more susceptible, and more challenging to maintain. However, compared to unit tests or API-level testing, UI tests are typically slower, more susceptible, and more challenging to maintain.

The key user interactions, which were tested using Selenium in this project include:

- New task addition by the UI, making sure that form entries and submission processes are functional.
and displaying the feedback's visual confirmation.
- Removal of the task, confirmation of proper processing, and UI presentation of removal activities.
- Checking the behavior of UIs with multiple browsers to detect the compatibility problems and guarantee the same user experience.

8 Benefits and Limitations of SwDevOps Adoption

Software Development and Operations (SwDevOps) practice are intended to overcome the divide between the development and the operations teams by encouraging collaboration, automation, and continuous feedback during the software delivery lifecycle. SwDevOps can be used to considerable beneficial effect on the performance, reliability and quality of software systems when properly applied. Nevertheless, its implementation also has a number of technical and organizational issues.

8.1 Benefits of Adopting SwDevOps

- Automated software delivery to occur faster.
- Early detection of bugs and integration issues
- Better co-ordination between development and operations teams.
- Increased quality and reliability of software.
- Reduced mistakes in manual deployment

8.2 Limitations of SwDevOps Adoption

- Primary installation may be complicated and time consuming.
- It needs knowledgeable members of staff who are aware of DevOps tools.
- Automated tests may not cover edge scenarios.
- The adoption may be slowed by cultural resistance in organisations.

9 Conclusion

The report has critically assessed how Scrum reflects the core beliefs of Agile by transforming the core beliefs into structured but flexible practices that enable the use of iterative and incremental software development. The roles, events, and artifacts of Scrum are well-defined offering structure, clarity, and accountability and at the same time facilitate quick adaptation to the changing specifications by teams. Through short development cycles, continuous feedback, and close stakeholder involvement, Scrum helps businesses to produce value quickly, continuously, reduce risk, and improve responsiveness to customer needs. Also, the effectiveness of Scrum is greatly increased by the integration of Software Development and Operations (SwDevOps) practices. Burndown charts and estimation tools enhance visibility, transparency, and tracking of the progress, which make informed decision-making throughout the sprint life cycle possible. The quality ownership is supported through continuous integration, automated testing, and deployment pipelines, which identify the defects in the first stages, consistency between the environments, and short release cycles. The combination of these practices allows sustaining a sustainable level of development and providing the quality software that is reliable and high quality. Although Scrum and SwDevOps have these benefits, implementation is not devoid of some problems. Technical tool complexity, cultural resistance to change, and the challenge of collaboration in big or dispersed teams can all impede successful implementation. The advantages of these approaches can be lower or not always achieved without adequate organizational support, maturity of the team, and technical expertise.

However, the joint deployment of Scrum and SwDevOps can offer a strong model that can handle the challenges of contemporary software development when, however, the two are implemented with a balanced approach and consideration of the situation. This hybrid model allows software teams to be adaptable to changing business needs by encouraging teamwork, flexibility, processes of continual enhancements, and innovation that help them sustain high-quality, scalable, and reliable software deliverables.

10 REFERENCES

- [1] K. Beck et al., "Manifesto for Agile Software Development," 2001. [Online]. Available: <https://agilemanifesto.org/>
- [2] K. Schwaber and J. Sutherland, The Scrum Guide, 2020. [Online]. Available: <https://scrumguides.org/>
- [3] M. Cohn, Agile Estimating and Planning. Upper Saddle River, NJ, USA: Prentice Hall, 2005.
- [4] J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston, MA, USA: Addison-Wesley, 2010.
- [5] Postman Inc., "Postman API Platform," 2024. [Online]. Available: <https://www.postman.com/>
- [6] Selenium Project, "Selenium Browser Automation," 2024. [Online]. Available: <https://www.selenium.dev/>