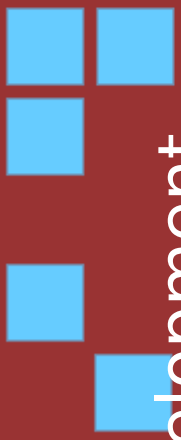

B.Sc. (Hons) in Software Development

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Streamlining CI/CD Pipeline into Web Development

By
GABRIEL HANG

for
JOSEPH CORR

April 24, 2023

Minor Dissertation

Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the digital transformation continues to accelerate, software development has become more crucial than ever before, with businesses relying heavily on applications to meet the demands of customers [2]. However, developing a decent software is a complex process which takes a long period of time and effort. With the ever-increasing need for faster and more reliable software delivery, traditional development methods are no longer sufficient [3]. As software development processes can involve multiple stages, it takes a significant amount of time and effort to complete. Moreover, developers may be assigned with multiple projects simultaneously, which can result in further delays. As such, this can lead to customer dissatisfaction and lost opportunities [4].

Automation has emerged as a solution to the issue of slow software development delivery. It improves software development's time and cost effectiveness by allowing software development teams to use resources more efficiently. Automation, for example, can aid in tasks requiring creative thinking or technical expertise, such as system integration, feature design, and debugging. Furthermore, the implementation of automation can decrease the duration needed to accomplish repetitive and mundane tasks, leading to a decrease in operational expenses. The use of automation, whether implemented partially or completely, can significantly improve software development processes [5].

Continuous Integration and Continuous Deployment (CI/CD) development practices have gained widespread popularity in recent years due to their effectiveness and efficiency in software development. Despite its growing popularity, some may argue that setting up CI/CD can be a time-consuming and complex process [6]. Therefore, it is vital to conduct research on this topic to ensure that development teams can effectively implement and reap the benefits of CI/CD.

1.1 Background Information

Continuous Integration and Continuous Deployment have become crucial components for efficient and effective software delivery. According to a survey conducted by a software testing company, Mabl, about half of the 500 participants are already implementing Continuous Integration (CI), and a quarter plan to do so in the near future. Conversely, 190 out of 500 respondents use Continuous Deployment (CD) while another 150 respondents have plans to implement it [7]. These findings demonstrate the increasing recognition and adoption of CI/CD in software development, highlighting the need for developers to keep up with this trend in order to remain competitive in the industry.

According to Kubinyi, automating testing, deployment, and delivery processes by streamlining the CI/CD pipeline can reduce the risk of errors and ensure faster release cycles for development teams [8]. These practices help developers automate the process of building, testing, and deploying their applications, allowing them to deliver software updates more quickly and efficiently. This approach can also improve collaboration among development teams, operations, and other stakeholders, ultimately leading to better software quality and user satisfaction [9]. Hence, CI/CD brings numerous advantages to software development teams and companies.

1.2 Problem Statement

According to multiple studies [4, 5, 10, 11, 12, 13], CI/CD pipelines enable the quick and dependable delivery of software changes, making it possible to implement new or amended product and service features within a shorter time frame. This significantly reduces the time required to implement ideas or fixes from weeks or months to a matter of days. This has shown that CI/CD pipelines have a positive impact on software development process.

However, it is true that setting up the pipelines can be difficult especially for those who are new to it or want to integrate an existing project [6]. Hence, it is crucial to have the be prepared to acknowledge the challenges that come during implementation as it requires a deep understanding of the development environment, structure and tools involved.

In response to the problem, the purpose of this research paper is to study and examine the effectiveness of CI/CD pipeline while designing a web application to test and deploy.

1.3 Significance of the Study

The outcome of this project can help organisations to benefit greatly from the adoption of CI/CD pipelines. Organisations can accelerate the development and delivery of software features by automating the process of testing and deploying software. This can result in a shorter time-to-market, increased productivity, and a better overall for customers experience [2].

Thus, clients or customers can receive more and faster feedback, as well as higher quality software as developers can detect and fix bugs earlier in the development cycle With CI/CD pipelines. Customers can also benefit from a more responsive and dynamic product with the ability to deploy new features and updates more frequently. [11, 14].

Finally, this research can help the researcher and other learners to gain a deeper understanding of CI/CD principles. Teams can achieve a more efficient workflow, better collaboration, and higher code quality by automating the software development process, all of which are critical for project success [6]. In this context, for instance, a website has been developed and implemented with a CI/CD pipeline to better understand everything from the ground up.

1.4 Objectives

1. To demonstrate an understanding of principles and practices of continuous integration and deployment for better efficiency.

2. To enhance designing and developing a scalable web application using modern front-end frameworks and back-end technologies.

3. To implement best practices for software development, such as testing and documentation.

4. To develop skills in project management, including project planning and tracking.

1.5 Project Repository Overview and Key Components

This dissertation presents the design and implementation of a CI/CD pipeline using GitHub Actions, while also developing a simple student grade management website using MERN stack with basic CRUD functionalities. The website is built

using React.js as the front-end and MongoDB as the database, with Node.js and Express.js serving as the back-end server.

A functioning student grade management website that incorporates basic CRUD functionalities and demonstrates continuous integration and deployment principles and practices has been designed. The code for this project is hosted on GitHub, and can be found here. The website is built using React.js as the front-end and MongoDB as the database. Initially, the back-end server was developed in Go language, but it was later changed to Node.js and Express.js due to deployment compatibility and issues. Next, Jest and Supertest were used to write tests for testing, and Heroku was used for deployment. Lastly, Jira, a project management software, is utilised to plan and track project progress.

The student grade system deployed from this repository can be accessed here and the following directory layout shows the hierarchy of directories and files for the project with description:

```
student-grade-system
├── .github/workflows
│   └── checks.yml
├── BACKEND
│   └── server.js
├── dissertation
├── public
├── src
│   ├── components
│   │   ├── create.SG.js
│   │   ├── listings.js
│   │   └── updateSG.js
│   ├── App.js
│   └── App.css
├── test
│   └── crud.test.js
├── .gitignore
├── Procfile
├── README.md
├── package-lock.json
└── package.json
```

- .github/workflows/checks.yml: This file sets up GitHub Actions to run checks on each push request to ensure that the code passes all tests and deploys automatically to Heroku.
- BACKEND/server.js: This file contains the back-end code for the project,

which handles CRUD API requests and database interactions.

- dissertation: This directory contains the author's dissertation in LaTeX format.
- public: This directory contains public files, such as images and static HTML files.
- src: This directory contains the main React source code for the project, including the components sub-directory that contains different pages of the website. The App.js file provides a navigation bar for every page, while the App.css file manages the CSS styles for the website.
- tests/crud.test.js: This file contains tests for the project, testing the Create, Read, Update, and Delete (CRUD) functionality.
- .gitignore: This file specifies files and directories to be ignored by Git when committing changes. For this project, the node_modules and build folders are ignored, for instance.
- Procfile: This file is used by Heroku to specify the commands to run when the app is deployed. In this project, this file is used to start the server.
- README.md: This file provides an overview of the project.
- package-lock.json: This file specifies version numbers for dependencies to ensure consistency across installations.
- package.json: This file specifies the project's dependencies and scripts for running the app.

Chapter 2

Methodology

The focus of this project is to design and implement a continuous integration and continuous delivery (CI/CD) pipeline. The pipeline would automate the building, testing, and deployment of the application to a production environment. In addition, a MERN (MongoDB, Express.js, React.js, Node.js) website is designed to integrate and test the pipeline automatically. The decision to carry out a CI/CD pipeline project was made due to the increasing popularity of automation in software development.

The project started with Agile development methodologies and transitioned to DevOps methodologies after completing the design and implementation of the CI/CD pipeline. Initially, the developer decided to switch the web application's backend from Node.js and Express.js to GO to make the project more challenging and the learn a new programming language. Basically, the idea was to build a pipeline and a website while pick up a new language.

Throughout the coding process, secondary resources were used frequently to set up the workflow and to learn the GO programming language. After the website was fully designed and the features were fully implemented, the developer started designing the CI/CD part of the workflow. The CI part went smoothly as expected, but the CD part encountered some difficulties. The section on system evaluation will provide explanations and evidence. After attempting several solutions, the developer decided to switch back to the initial MERN stack as the focus was on designing the CI/CD pipeline.

This section will be divided into two parts: software development approaches and tools and technologies used in this project.

2.1 Software Development Approaches

The developer used Agile development methodologies initially for designing the website and the pipeline, and then transitioned to DevOps after completing the design and implementation of the CI/CD pipeline. However, it is important to note that DevOps is not limited to only CI/CD but is a combination of development and operations practices that emphasize automation and monitoring throughout the software development lifecycle.

By combining Agile development with DevOps practices, the developer aimed to create an efficient, reliable, and scalable software development process [15]. Agile provided flexibility and adaptation to changing requirements during the design and development phases, while DevOps ensured automation, testing, and monitoring during deployment and production phases. The integration of these two methodologies aimed to create a seamless and streamlined software development process from conception to production.

2.1.1 Agile Development

Moving away from traditional software development methodologies that are incapable of adapting to changes, the developer chose to use the agile method for the development phase as the project involves incremental changes and iterations and does not have fixed features as they may change due to factors such as time constraints. Furthermore, Agile methods strive for a faster release schedule, making them ideal for projects that require quick adaptability to changes [1, 15, 16, 17]. Table 2.1 below shows how agile was chosen over traditional methodologies like waterfall, for instance.

Parameter	**Traditional**	**Agile**
Ease of Modification	Difficult	Easy
Development Approach	Predictable	Adaptive
Development Orientation	Process-focused	Customer-focused
Team Size	Medium	Small
Budget	High	Low

Table 2.1: Comparison of Software Development Methodologies: Traditional and Agile [1]

Scrum

The Scrum framework from Agile was adopted for this project. Scrum is an iterative and incremental framework for managing product development that emphasizes teamwork, accountability, and adaptability [1, 17]. The Scrum framework consists of several roles, including the Product Owner, Scrum Master, and Development Team. Figure B.1 in Appendix B contains a visual representation of the Scrum process.

Despite being a one-man team, the developer included the product backlog, utilized sprint planning to break down user stories into manageable tasks and run sprints using Jira. Examples of the product backlog, sprint planning and sprint execution can be found in Appendix A.

2.1.2 DevOps Implementation

After completing the design and implementation of the project's CI/CD pipeline, the developer began the transition to the "DevOps era". DevOps is a fundamental aspect of Agile that incorporates operations and it is all about "continuous" [18, 19]. Figure C.1 shows a simple illustration of DevOps in Appendix C [20]. DevOps can be considered an improved version of Agile that emerged to address the bottleneck that prevented development teams from delivering to operations more quickly and frequently [15].

The DevOps approach aims for faster deployment of software and quicker response to changes. This is possible because a well-designed pipeline can deploy changes in a matter of minutes as opposed to several days for manual deployment [21, 22]. DevOps was used to continuously monitor, maintain, and improve the CI/CD pipeline. It assists in aligning and automating the process across development, testing, deployment, and support phases, and includes best practices such as code repositories, build automation, continuous deployment, and others [23].

What is CI/CD? [6, 24]

- Continuous Integration (CI): A practice that involves merging code changes from a development team into a shared repository on a regular basis, followed by an automated process of building and testing the application to identify any integration issues.
- Continuous Deployment (CD): The process of automating the release of software changes to production (aka deployment).

In this project, GitHub Actions is used as the tool to set up the pipeline using workflow file. When the developer makes changes to the project, mainly the

coding part, and pushes it up to GitHub, the workflow begins to build, test, and deploy the application automatically. For the testing part of the workflow, the developer wrote a test program and added it to the workflow while Heroku is used for deployment. Further information regarding these tools and technologies will be discussed in the following section.

2.2 Tools and Technologies

2.2.1 Version Control System

Git

The use of a version control system is an important aspect of this project's methodology. For managing the application's source code, Git is used as the primary version control system. GitHub is used as a remote repository to store the project's source code, allowing the developer to manage changes and to track changes over time thanks to Git's ability to push code as checkpoints called commits.

2.2.2 CI/CD implementation

GitHub Actions

To ensure that changes to the application were automatically built, tested, and deployed to the production environment, GitHub Actions is used to set up the pipeline. GitHub Actions is a popular CI/CD platform that provides developers with the ability to automate their software development workflows. It offers several features such as building, testing, and deploying code from within GitHub. Th developer has to design and implement a workflow for automation.

Supertest and Jest

Supertest and Jest are popular testing frameworks used for CI. Supertest is a Node.js library used to test HTTP requests, while Jest is a testing framework for JavaScript projects that provides a complete testing solution with a focus on simplicity.

Heroku

Heroku, a Platform as a Servive (PaaS) is a popular cloud-based platform used for CD. It allows developers to deploy, manage, and scale their applications quickly and easily. Heroku provides a wide range of features, including support for multiple programming languages, integration with third-party services, and a powerful API.

2.2.3 Database

MongoDB

A database is needed to store simple information of students. MongoDB is an open source, non-relational database management system that processes and stores data in the form of flexible documents rather than tables and rows. MongoDB does not require a relational database management system, so it provides an elastic data storage model that allows users to easily store and query the database. This simplifies database management for developers and provides a highly scalable environment, which is why MongoDB is chosen to be the database of this project.

2.2.4 Backend Server API and Frontend Development

The initial plan was to develop a MERN stack application with CRUD functionalities to support the pipeline. However, for additional knowledge, the developer decided to use GO language for the backend server as mentioned previously. GO language was chosen due to its portability, efficiency, and sufficient library resources, which will be further discussed in the technical review section [25, 26, 27]. However, due to deployment issues, the developer had to switch back to the MERN stack to ensure a smooth implementation of the pipeline.

The MERN stack is a full JavaScript stack and a lightweight web development framework that is used in this project due to its high efficiency, loose coupling, and high cohesion. Appendix D states what each component of MERN stands for individually. [28, 29].

The CRUD functionalities that are implemented in the student grade system for this project are as follows:

- Create: Add a student with grades and other information.
- Read: Get all students' information and filter certain categories of students.
- Update: Update student grades and/or information.
- Delete: Delete a student.

2.2.5 Project Management

Jira

Project management plays an important role in ensuring the success of software development projects. In this project, Jira was used as the project management tool. Jira provides a powerful set of features that allow the developer to plan,

track, and manage their work effectively. The following steps were taken in the project management process:

1. Planning: Tasks were created with story points and organised in the backlog, which were then prioritised and moved into sprints for development accordingly. Figure A.1 in Appendix A shows an example of the backlog with tasks/issues grouped into sprints.

2. Task management: Tasks were categorised into different epics, which represent a larger features or milestones in the project. The project roadmap, which can be seen in Figure A.2 in Appendix A, illustrates the different epics and their progress.

3. Progress tracking: The progress of the project can be tracked using the roadmap in general or by using Jira's board feature, which allows the developer to view and change the state of tasks (to do/in progress/done) in real-time. Figure A.3 in Appendix A provides an example of the board feature.

Chapter 3

Technology Review

3.1 Continuous Integration and Continuous Deployment (CI/CD)

CI/CD is a set of DevOps practices that automate the software delivery process [5]. It ensures that new functionality is pushed through an automated workflow, known as a pipeline, which runs software tests and quality assurance, before the code is deployed. The pipeline enables developers to deploy new code changes in a short period of time, with the ability to roll back at any stage safely [9]. To achieve CD, it is important to practice CI beforehand, as this step is essential for achieving the continuous delivery of software [3]. This is because the pipeline ensures that code is tested and quality assured prior to deployment, lowering the chances of errors and defects in production, which ensures that the software is reliable and performs as expected.

One of the major advantages of the CI/CD pipeline is the instant feedback that developers get based on their code [9]. This helps developers identify and fix bugs quickly, thus reducing the time and effort needed for fixing bugs. According to the 2022 State of DevOps Report [7], CI and CD can increase code deployment efficiency by as much as 208 times. This shows that this approach is an important step for ensuring efficiency in software development.

The CI/CD pipeline also promotes agile software development and DevOps practices. By automating the deployment process, developers can quickly and easily deploy new code changes, reducing the time to market for new features, which then saves time and money [9, 13]. The faster software is deployed, the more revenue/profit can be generated. In addition, the automated process reduces the need for manual intervention, which saves time and reduces the risk of errors. More than half of the respondents in a survey are currently using or are planning to use the CI/CD approach [7]. This clearly shows that more and more developers

are looking into the advantages of CI/CD.

However, the CI/CD pipeline can be complex and takes time to learn [6]. Developers need to understand the process and tools involved, which can be daunting for beginners. Although the process is automated, some steps in the pipeline may require manual approval. This can slow down the deployment process and introduce delays which cuts down company revenue [4]. Some tests require human supervision, which can slow down the deployment process [30]. This is particularly true for tests that require visual inspection, such as user interface tests.

Furthermore, the CI/CD pipeline can introduce security risks if not properly implemented. Developers need to be aware of potential vulnerabilities and take steps to mitigate them. The introduction of CI/CD has not been flawless. For example, half of organisations doing so do not include any security testing elements, according to a 451 Research survey of 350 enterprise IT decision-makers in North America and Europe fielded in May 2018 [7]. This might be due to that the developers want to deliver the features faster then it should take. While speed is important, it is important to go as fast as possible but no faster. Going too fast can introduce errors and defects, which can be costly to fix.

To summarise that, the CI/CD pipeline is an essential DevOps practice that automates the software delivery process. It provides immediate feedback to developers, promotes efficiency and agility, and saves time and money. However, it can be complex and require a significant learning curve. CI/CD pipeline can encourage developers to go too fast before they are ready. Developers need to be aware of potential risks, such as security vulnerabilities and the need for human supervision in some tests. Ultimately, the CI/CD pipeline can help organizations deliver software faster and more reliably, but it requires a disciplined approach to implementation and management.

3.2 Agile Methodology

Agile software development methodology is becoming increasingly popular as it aims to address the insufficiency in traditional software development processes. With over 90% of companies utilizing an agile approach for software development, it seeks to ensure a close link between the customer and developers to ensure that software meets market needs while striving for a more rapid release schedule [15]. Table 2.1 from the previous section compares agile with traditional methods which proves why agile is preferable than traditional methods.

Agile methodology is characterised by its light nature, which makes it response to change quickly [17]. Survey studies by the Standish group have reported that less than 50% of software development projects are successfully delivered within the set time, budget, and scope. As such, Agile technique ensures that each feature

is continuously enhanced until it satisfies the ultimate user requirement [1]. This is feasible as Agile methods emphasize continuous feedback and improvement, leading to quicker responses and faster delivery [16]. Each release adds new features or functionalities, which improves the overall quality of the software. Agile also optimizes communication, favoring face-to-face communication, which enhances the overall quality of communication between the customer and developers[17].

Although Agile methodology is effective in direct communication, it is considered weak in documentation, making it less suitable for long-term software maintenance [1, 17]. The lack of emphasis on documentation can lead to different recollections of the same exchange, which can cause confusion and errors in the development process. Despite the advantages of increasing the frequency of software development, an issue has developed because the Operations function (Ops) and the Development function (Dev) are not aligned [15]. This is because Agile focuses on the Dev part but not the Ops part. This misalignment can cause long delays in software releases to customers, ultimately hindering the overall success of the project.

Overall, Agile is a suitable methodology for projects that demand flexibility, quick responses to change, and close collaboration between developers and customers. Conversely, long-term projects or maintenance is not recommended to use Agile as documentation is not prioritised which may cause problem in the long run. It is important to consider the nature of the software development project and choose the most suitable methodology accordingly.

3.3 DevOps Approach

DevOps is the combination of development and operations practices that emphasize automation, collaboration, and monitoring throughout the software development lifecycle [21]. DevOps is an extension of Agile methodology, incorporating operations practices to provide an end-to-end solution from design to monitoring, with a focus on automation and continuous feedback. DevOps emerged to address the deficiency that prevented development teams from delivering to operations in a faster and more frequent manner [15]. Many organizations have adopted DevOps practices to improve their software development processes as shown in various surveys and studies [23].

DevOps' primary benefit is the little manual intervention required for speedy and smooth application deployment [23]. DevOps may improve performance and efficiency, making the software development process quicker, more dependable, and more efficient by automating development, build, test, and deployment. Also, the automated process eliminates manual labor, allowing team members to concentrate on other duties [21]. Moreover, DevOps emphasizes collaboration between

development and operations teams, enabling faster delivery speed, reliability, and scalability. DevOps also enhances operations by configuring, deploying, and monitoring applications, leading to better utilisation of resources and less time wasted [20]. The continuous integration (CI) aspect of DevOps makes applications more robust and reliable. By continuously testing and integrating code, DevOps ensures that code is always production-ready. The automation aspect of DevOps also helps in identifying and fixing issues quickly, leading to improved software quality.

However, putting DevOps into practice has its share of difficulties. Working with DevOps implementations presents several difficulties, one of which is that it is simple to lose sight of the intended outcome [21]. DevOps is about delivering increased business value, not just about doing things faster. Therefore, it is important to keep track of the goal and measure the outcomes to ensure that the DevOps implementation is delivering value. Another challenge with DevOps is that it requires a significant cultural shift in the organization. The cultural shift can be difficult to achieve, as it requires all team members to work together and take ownership of the process. Additionally, the automation aspect of DevOps requires expertise in the use of automation tools, which can be a challenge for some team members. Finally, DevOps can be expensive to implement, requiring the use of expensive automation tools and a dedicated team to manage the automation process. The initial investment may be high, and it may take some time to realize the benefits of DevOps.

In conclusion, DevOps is a software development methodology that has become increasingly popular in recent years due to its ability to enhance automation, collaboration, and monitoring throughout the software development lifecycle. DevOps has several advantages, such as minimal manual intervention, faster deployment, and improved reliability. However, there are also challenges associated with DevOps, including the need for a cultural shift, expertise in automation tools, and high implementation costs. Organizations must weigh the pros and cons of DevOps and determine if it is the right methodology for their software development needs.

3.4 Jira

Jira is a project management tool that has gained increasing popularity for software development teams, especially those that use agile methodologies. The issue tracking system in Jira is particularly useful for Scrum, which emphasize iterative and incremental development [31, 32, 33]. It allows teams to track their work, manage projects, and collaborate in a single platform.

Jira is known for its user-friendly interface. The user interface is very straightforward and easy to understand which enables users to get the hang of it in a short

period of time. This feature is particularly important for teams that need to get started quickly and do not have a lot of time for training or research [34]. Apart from its ease of use, Jira also provides extensive data reporting capabilities that enable teams to generate custom reports and visualize data to track progress and identify areas for improvement. Reports such as burnup chart burndown chart and cumulative flow diagram can be generated automatically by Jira, Additionally, Jira offers a wide range of features that can enhance efficiency, such as a robust searching facility that enables users to search for issues and tasks using various criteria such as keywords, assignees, and due dates [34, 32]. This feature makes it easier for software development teams to find what they need quickly and efficiently.

Despite its many advantages, Jira has some limitations, one of which is its cost, particularly for small teams or startups, as some of its features require payment. This cost can add up as the team size increases [32]. However, Jira does offer a free plan that teams can choose to use. Therefore, development teams should carefully consider the costs associated with using Jira before deciding to adopt it for their projects.

In summary, Jira is an effective project management tool that provides numerous benefits for agile software development teams. Its user-friendly interface, extensive data reporting capabilities, and integration with other tools make it a valuable asset for teams looking to enhance their efficiency and collaboration. However, teams should carefully consider the costs associated with using Jira before deciding to adopt it for their projects.

3.5 Heroku

Heroku is a platform as a service (PaaS) that changes the way web applications are deployed by providing an easy-to-use platform that allows developers to deploy, manage, and scale their applications [35]. In 2012, Heroku won the InfoWorld Technology of the Year award, cementing its position as a leading platform as a service provider with focus on infrastructure [36, 37].

One of the significant advantages of Heroku is its flexibility [36, 38, 39]. Heroku supports a wide range of programming languages, including many modern open-source languages such as GO and Ruby. The list of supported languages is still growing, which makes it an attractive option for developers who want to work with the latest tools and technologies. Additionally, Heroku is highly scalable, and it scales transparently as traffic spikes, making it capable of serving applications with over 10,000 sustained requests per second today [38].

Heroku's user-friendly interface and ease of use are also an advantage [36]. Developers can easily connect to Heroku from Git, GitHub, and Docker, and the platform's adaptability means that it can adapt itself to the needs and requirements

of the software. This makes it a popular choice for developers who want to focus on coding and development, rather than infrastructure management.

However, there are some limitations to using Heroku. One of the significant disadvantages is that heavy projects do not always run well on the platform [36]. Heroku is designed to be easy to use and manage, which means that it may not be the best option for large, complex applications that require high levels of customization and control. Additionally, Heroku's pricing structure can be relatively expensive compared to other cloud platforms, especially for projects that require a lot of resources.

Overall, Heroku is a popular platform as a service that provides developers with a flexible, scalable, and user-friendly way to deploy and manage their web applications. Its focus on infrastructure and support for modern open-source languages makes it a popular choice for developers. However, its limitations, such as poor performance for heavy projects and a relatively expensive pricing structure, may make it unsuitable for certain projects. Developers should consider the specific requirements of their project before deciding to use Heroku as their deployment platform.

3.6 GitHub Actions

GitHub Actions is an innovative CI/CD solution provided by GitHub. It is a cloud-based, online service that can help researchers track, organize, discuss, share, and collaborate on software and other materials related to research production, including data, code for analyses, and protocols [40, 41]. And...it is free!

GitHub Actions is a CI/CD solution that allows for the automation of various stages in software development, including build, test, and deployment. It enables developers to create workflows using YAML configuration files. One of the main benefits of using GitHub Actions is that it is easy to design pipelines for code that is already hosted on GitHub. This makes it easier to automate the entire software development process, from code changes to production deployment. Another advantage of GitHub Actions is that it can alert developers of failing processes through email [41]. This means that developers can be notified of any issues in real-time and can address them quickly. This can help reduce downtime and increase the reliability of the software development process.

However, one of the disadvantages of using GitHub Actions is that it requires familiarity with YAML configuration files. Developers who are new to YAML may need to spend some time learning and understanding the syntax [40]. This initial research and learning process can be challenging for some, but once developers become familiar with YAML, they can benefit from the power and flexibility that GitHub Actions provides.

All in all, GitHub Actions is a powerful tool that can help researchers streamline their software development process. It offers many benefits, including free usage, easy pipeline design, and real-time alerts for failing processes. Although there is a learning curve associated with YAML configuration files, the benefits of using GitHub Actions make it a worthwhile investment for researchers looking to improve their software development process.

3.7 GO Language

GO is an open-source programming language developed by Google that is popular among web services and web applications, including Docker and Netflix. The language was designed with the primary goal of creating a modern language that could solve real-world problems, making it easier for developers to write code with practical warning and error messages [25].

One of the significant advantages of GO is its rich library, which allows developers to leverage existing code to solve common problems and build complex applications more efficiently, saving time and effort. Additionally, GO is portable and supports every operating system fully, making it possible to run on any platform which is not feasible in other modern languages[26]. GO is famous for its concurrency, procedural, and distributed programming. Goroutines in GO can handle up to several thousand threads at the same time, making it useful for building web applications which handle multiple transactions [26, 27]. Furthermore, GO features garbage collection, which frees developers from worrying about memory allocation or deallocation [26]. Objects that are not needed anymore will be freed automatically.

However, GO also has some limitations, including slower processing speed compared to languages like C, and a lack of direct support for object-oriented programming, which can be challenging for developers accustomed to that programming style [25]. Additionally, while GO's standard library is rich in features, it may not have as many third-party libraries as other languages, which could limit the options available for developers when they want to use a specific library in their project.

In conclusion, GO is a versatile language that prioritizes practicality and ease of use, making it popular among many developers. Its rich library, portability, and support for concurrency and distributed programming make it ideal for building web applications. However, developers must consider its limitations, such as slower processing speed and lack of direct support for object-oriented programming, before deciding to use GO for their project.

3.8 MERN Stack

MERN is a full JavaScript stack that comprises of four different technologies: MongoDB, Express, React, and Node.js. It is a popular stack that is widely used for developing web applications, especially for projects that require rapid prototyping.

One of the significant advantages of using MERN is that all four components use the same language, JavaScript. This allows developers to save time and increase productivity as they don't have to switch between different programming languages. Moreover, using a consistent language across the stack ensures that the codebase is uniform and easy to maintain [29]. Another benefit of MERN is that it provides better performance compared to the MEAN stack, which uses Angular. React, a component of MERN, is known for its high performance and efficient rendering. This makes MERN a great choice for building large-scale, data-intensive applications.Security is another advantage of using MERN. Since MERN uses JavaScript for both server-side and client-side development, it provides a secure environment for sensitive data. Moreover, MERN has an extremely low latency, which ensures that the application loads quickly and efficiently.

Despite its advantages, MERN has a steep learning curve, which can be challenging for new developers. It requires a solid understanding of JavaScript and its associated frameworks, as well as the ability to integrate them seamlessly [28]. Another disadvantage of MERN is that it runs in the browser, not on the user's server [42]. This can lead to slower performance if the user has a poor internet connection or if the application is not optimized for performance. However, this can be mitigated by using caching and optimizing the application for speed.

In conclusion, the MERN stack has several advantages, including a consistent language across all components, better performance compared to MEAN stack, and enhanced security. However, it also has a steep learning curve and may suffer from slow performance if the user has a poor connection. Overall, the MERN stack is a useful tool for developers who want to build fast and secure web applications, but it requires a certain level of expertise in JavaScript to be able to use it effectively.

Chapter 4

System Design

This section is based on the final product of the project. Any compatibility or deployment issues that have been mentioned will be discussed in the system evaluation section.

4.1 CI/CD Design and Implementation

GitHub Actions is used as the CI/CD tool in this project. Workflow files (.yml or .yaml) that automate tests, deployment, etc. are assigned using GitHub Actions. It is possible to run the workflow automatically after certain GitHub events like commit push. A green tick or a red cross will be shown beside the commit in the repository to indicate if the workflow has been completed successfully.

Workflow files can be designed manually or by using recommended workflows on GitHub such as Jekyll or readily made template like JupyterLite. In this project, the pipeline is designed manually. The workflow is set up at the start of the project alongside with the coding environment. The testing and the deployment part of the workflow is designed after the website is fully working. Redirect to the GitHub repository page here to access the workflow file.

Figure 4.1 shows the yml file of the workflow. The workflow is triggered whenever a commit is pushed to the main branch. The workflow has one job named "build" that will run on an Ubuntu operating system that has five steps.

1. Checkout from the repository for automation.
2. Setup Node.js version 18.x.
3. Install dependencies using npm
4. Run tests using npm with designed test suite integrated in the project with `npm install` command.

5. Deploy the code to Heroku using the *akhileshns/heroku-deploy* action with the Heroku API key the app name and the email of the account. The Heroku API key is stored in GitHub secrets and is accessed through the ${{ secrets.HEROKU_API_KEY }} variable.

```yaml
# This workflow will ensure website features work perfectly
name: Checks

# Controls when the action will run
on:
  # Workflow runs every time when a commit is pushed to the main branch
  push:
    branches:
      - main

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  build:
    # The type of runner that the job will run on
    name: Build
    runs-on: ubuntu-latest

    # Steps represent a sequence of tasks that will be executed as part of the job
    steps:
      # Checkout repository for workflow to access
      - name: Checkout
        uses: actions/checkout@v3

      # Setup environment
      - name: Setup Node.js
        uses: actions/setup-node@v2
        with:
          node-version: "18.x"
      - name: Install dependencies
        run: npm install

      # Run test (CI)
      - name: Run test
        run: npm test

      # Deploy (CD)
      - name: Deploy to Heroku
        uses: akhileshns/heroku-deploy@v3.12.12
        with:
          heroku_api_key: ${{ secrets.HEROKU_API_KEY }}
          heroku_app_name: "student-grade-system"
          heroku_email: "g00377198@atu.ie"
```

Figure 4.1: GitHub Actions Workflow

Overall, this workflow ensures that the website's features work perfectly by

building, testing, and deploying the system automatically in response to changes in the main branch.

The `npm test` in the workflow file runs the tests automatically from the directory. The test file was design using Supertest and Jest to test out CRUD API with a test suite with five test cases:

1. Able to CREATE a new student information to database
2. Able to READ all student information from database
3. Able to READ a specific student information from the database for update
4. Able to UPDATE a student information
5. Able to DELETE a student information

The code snippet from Figure 4.2 below tests the functionality of a create functionality, specifically, the `createGrade` function that creates a new student record. It sends a HTTP POST request to the */createGrade* endpoint with test data and expects a response status code of 200 and a response message of 'Student Grade Added'. It stores the ID of the newly created student record in a variable called `testId` to test out other functionalities. This test case ensures that the `createGrade` function is able to create new student records without errors.

```
// Initialize testId variable to store document ID during creation
// to test other functionalities
let testId = '';

// Test Create function
it('should create a new student', async () => {
  const response = await request(app)
    .post('/createGrade')
    .send({
      // Test data
      studentNumber: 'T001',
      name: 'Test',
      grade: 1,
      year: '1',
      class: '1'
    });
  expect(response.statusCode).toEqual(200); // Expect no errors
  testId = response.body.data._id; // Store testId when created
  expect(response.body.message).toEqual('Student Grade Added');  // Expect response message to show successful creation
});
```

Figure 4.2: Test Create Functionality

After ensuring the application is working as expected by running the test suite, the workflow uses Heroku, a Platform as a Service (PaaS), to deploy the application. It is vital to note that *Procfile* is included in the directory as it specifies the commands that are executed by the application on startup, and the server will be started up in this context. After running the server side, the Heroku post-build is used to run the build file to show the frontend. Figure 4.3 below shows that `heroku-postbuild` is added under the script in package-json.

```
"scripts": {
  "start": "node BACKEND/server.js",
  "build": "react-scripts build",
  "test": "cross-env NODE_ENV=test jest --testTimeout=5000",
  "eject": "react-scripts eject",
  "heroku-postbuild": "NPM_CONFIG_PRODUCTION=false && npm install && npm run build"
},
```

Figure 4.3: Code Snippet from package.json to show heroku-postbuild

4.2 MongoDB Database Design

For this project, there is a collection in the database called *grades* with the following structure:

Field	**Datatype**	**Special Conditions**
_id	ObjectId	Auto-generated by database
studentNumber	String	Required
name	String	Required
grade	String	Required
year	String	Required
class	String	Required

Table 4.1: MongoDB Structure

The *_id* that is generated automatically by MongoDB is used to get specific document from the database while the rest is just the data for the application which is required as stated.

4.3 MERN Stack

As discussed previously in methodology with reference to Figure D in Appendix D, The database used for the system is MongoDB, the frontend or the webpage is developed using React.js and the backend or the server is built using Node.js with Express.js framework. The CRUD functionalities that are implemented in the student grade system for this project are as follows:

- Create: Add a student with grades and other information.
- Read: Get all students' information and filter certain categories of students.
- Update: Update student grades and/or information.

- Delete: Delete a student.

The above functionalities are designed as APIs in the backend with the frontend used for interaction and the database utilized for managing the data.

Chapter 5

System Evaluation

This section assesses and identifies potential areas for improvement and effectiveness in meeting the objectives. It also addresses any problems or difficulties that may have impacted or affected the project throughout the whole process.

5.1 Project Management-Jira

The project lasted for a total of five months with 15 sprints conducted. Due to some scopes being infeasible or undersized, they had to be adjusted. Also, a few sprints did not complete on time as planned as shown in the sprint burndown chart in Figure A.4 from Appendix A. All sprints were initiated and completed within a month or less, fulfilling the qualification criteria that typically takes one to two weeks, but not more than one month.

5.2 GitHub Commits

At the beginning, there were fewer commits as more research was carried out. Once the research process was completed, the number of commits gradually increased as shown in Figure 5.1 below.

Figure 5.1: GitHub Commits

5.3 GitHub Actions Workflow

A large research work had been done to understand the design of workflow, unsuccessful initial attempts to build the workflows at the beginning of the development process is shown in Figure E.1 in Appendix E. Errors that the developer encountered included indentation and missing job names in the workflow file, as well as the correct path to start a process (refer to Figure E.2 in Appendix E).

After completing the server using GO language and successfully running it locally, compatibility issues surfaced when attempting to deploy the initial application. At first, Heroku was not able to detect the language of the application as GO and Javascript were used and it can only auto-detect one language. When the developer specified the language by adding Node.js and GO buildpacks, the server side still did not set up as one with the client to deploy as a single application due to compatibility issues. Deployment was constantly being rejected by Heroku as shown in Figure E.3 in Appendix E. After few tries, the developer was still unable to deploy the frontend and backend due to the lack of online resources and information as this approach had not been taken by many developers. The developer also tried different deployment approaches using AWS and Docker. However, when the student grade system is finally deployed using Docker and Heroku, only the client side worked.

Due to time pressure, the developer decided to redesign and redevelop the backend with Node.js and Express.js instead of GO. The MERN stack was deployed successfully using the `Procfile` to start up the server and used the Heroku post-build feature to start up the client side as discussed in the system design section. The app is successfully deployed as shown in Figure E.4 in Appendix E and can be accessed here

5.4 Database

Initially, the MongoDB URI for the database was set to *localhost:27017/* which was the port number of MongoDB, but it was later changed to a more specific address that accesses that collection. The other problem is that the URI is exposed in the file, hence, it is not secure enough. Furthermore, the developer realised that the data was not written into the correct collection in the database during the development process. Hence, the developer changed the URI with the correct collection name by specifying the collection name it which is *students*. Here is the final MongoDB URI:

mongodb+srv://gabhang:gabrielfyp@cluster0.g0aphtm.mongodb.net/students?retryWrites=true&w=majority.

5.5 GO Language

The developer spent a significant amount of time researching GO language with its documentation and simple syntax. GO was difficult and different from the languages the developer had previously learned and it was not as easy as stated on the internet. Despite this, the developer was able to develop a functional application in GO. Unfortunately, due to deployment issues, the developer was unable to demonstrate the final product with GO and was forced to switch to the MERN stack. Despite this setback, the developer found it beneficial to learn GO and develop an application that can run on it.

5.6 Testing

The testing part for the project was written in GO at the beginning, but was later switched to JavaScript due to changes in the backend to Node.js and Express.js. For testing, Supertest and Jest were utilized as they were found to be very straightforward and easy to understand. Though Mocha and Chai were considered, ultimately they were not used. While Jest is mainly for testing out React, it was still deemed suitable for the project. But, the testing focused only on the backend of the project. During the testing process, an issue was encountered as Jest detected one open handle that could potentially keep Jest from exiting. Upon further investigation, it was discovered that the port was not closed as stated by the console. To resolve this, the code was updated to include a code snippet to close up the application or port after testing, enable Jest to finish execution. Nonetheless, the test suite with five cases passed perfectly at the end.

5.7 Frontend and backend development

Overall, both frontend and backend development progressed smoothly without any major hiccups. However, one issue did arise when the frontend attempted to make an HTTP request with hardcoded routes. Consequently, the application failed to function properly when deployed on Heroku as it was unable to locate the original port 4000 used on local machines. To overcome this, the developer modified the route to be accessible even when the code is deployed on Github.

Chapter 6

Conclusion

The significance of software development has increased significantly, and businesses heavily rely on applications to meet customer demands [2]. However, traditional development methods are no longer sufficient to meet the ever-increasing need for faster and more reliable software delivery [4]. Automation has emerged as a solution to tackle the problem of slow software development delivery. Continuous integration and continuous deployment (CI/CD) development practices have become increasingly popular for productivity and efficiency in recent years.

During this research, it became evident that Go, despite being initially considered easy to learn and use, can be quite challenging. Additionally, setting up CI/CD pipelines can be difficult, especially for those who are new to it or want to integrate an existing project. Different projects have to set up the pipeline in their way or according to different factors. However, the effort of setting up the pipelines is worth it, as it saves a lot of time when making changes to the code. CI/CD should be used more widely as it involves testing before deployment, which increases reliability.

The developer successfully incorporated a CI/CD pipeline into the project, meeting the objectives of the project. However, there is room for improvement in terms of planning and research in future projects. The use of tools like Jira can aid in better development planning and management, helping teams stay on track. Despite the need to recode the entire backend, the project was completed on time. To enhance the project's efficiency, it is crucial to consider the security of the database and take necessary measures to safeguard it against external threats. In summary, while the developer achieved the project's objectives, there is still room for improvement in terms of better planning and research for future projects, as well as enhancing the security of the database.

Overall, the adoption of continuous integration and continuous deployment (CI/CD) pipelines can greatly benefit organizations looking to streamline their software development processes [8]. By automating the process of testing and

deploying software, organizations can accelerate the development and delivery of software features, resulting in a shorter time-to-market, increased productivity, and a better overall customer experience [8].

6.1 Implications

This paper contributed to the subject area by examine the effectiveness of CI/CD pipeline to an application. The adoption of continuous integration and continuous deployment pipelines can greatly improve the efficiency and effectiveness of software development processes. By automating the testing, deployment, and delivery of software, developers can reduce the risk of errors and ensure faster release cycles. This approach can also improve collaboration among development teams, operations, and other stakeholders, ultimately leading to better software quality and user satisfaction. More research or projects should be conducted to examine different aspects that will contribute to the CI/CD approach field so that developers and companies will be conscious of how it can affect software development processes.

6.2 Limitations

Despite the significant implications of this study, there were several limitations that must be acknowledged. First, the study was limited by the availability of data and resources. The developer did not have the final system written in GO for the backend due to this. Second, due to the time constraints of the project, the research focused on a limited number of CI/CD tools and practices. As a result of a short time frame, the study did not evaluate the long-term effectiveness of CI/CD pipelines, and additional research might need to determine the sustainability of these practices in the long run. Not only that, since the project team consisted of only one member, it may not have been comprehensive enough to include all potential or related sections and components.

6.3 Recommendations

In spite of the limitations of this study, the researcher anticipates further researchers can deal with the limitations of this study, hence, overcome them. Researchers should aim to replicate this study in different contexts to determine the generalisability of the findings. Future studies should explore the effectiveness of a wider range of CI/CD tools and practices to determine the most effective approaches. Researchers should investigate the long-term effectiveness of CI/CD

pipelines and evaluate their sustainability over time. Organizations that are considering the adoption of CI/CD pipelines should conduct a thorough analysis of their development environment, structure, and tools to ensure that they have the necessary resources and expertise to effectively implement these practices. Furthermore, project teams should consist of more than one person to ensure quality, and scope needs to be planned properly with efficient research to avoid encountering problems in the middle of the development process.

Bibliography

[1] Amel DŽANIĆ, Amel TOROMAN, and Alma DŽANIĆ. Agile software development: Model, methods, advantages and disadvantages . *Acta Technica Corviniensis - Bulletin of Engineering*, 15(4):95 – 100, 2022.

[2] Jez Humble and David Farley. *Continuous delivery : reliable software releases through build, test, and deployment automation ;.* A Martin Fowlers Signature Book. Addison Wesley, 2011.

[3] Ionut-Catalin Donca, Ovidiu Petru Stan, Marius Misaros, Dan Gota, and Liviu Miclea. Method for continuous integration and deployment using a pipeline generator for agile software projects. *Sensors (14248220)*, 22(12):N.PAG, 2022.

[4] Cliff Saran and Adrian Bridgwater. Turning ideas into software - quickly. *Computer Weekly*, pages 22 – 25, 2019.

[5] Joonas Saarenpää. Creating an azure ci/cd pipeline for a react web application. 2020.

[6] Rossel Sander. *Continuous Integration, Delivery, and Deployment : Getting Started with the Processes and the Tools to Continuously Deliver High-quality Software.* Packt Publishing, 2017.

[7] Lindsay Clark. Deliver at speed with ci/cd: Continuous integration and delivery promises a remarkable acceleration in software development, but adaptors first need to get some basics in place to reap the rewards. *Computer Weekly*, pages 19 – 22, 2019.

[8] Iván Kubinyi. Building an effective CI/CD pipeline for a content management system. pages 44+4, 2022.

[9] Adrian Bridgwater and Cliff Saran. What changes are in the ci/cd pipeline?. *Computer Weekly*, pages 17 – 20, 2019.

[10] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5:3909–3943, 2017.

[11] Lianping Chen. Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, 32(2):50 – 54, 2015.

[12] Vidroha Debroy and Senecca Miller. Overcoming challenges with continuous integration and deployment pipelines: An experience report from a small company. *IEEE SOFTWARE*, 37(3):21 – 29, 2020.

[13] A Phillips, M Sens, A De Jonge, and M Van Holsteijn. The it manager's guide to continuous delivery: Delivering business value in hours, not months. *XebiaLabs*, 2015.

[14] Marko Leppanen, Simo Makinen, Max Pagels, Veli-Pekka Eloranta, Juha Itkonen, Mika V. Mantyla, and Tomi Mannisto. The highways and country roads to continuous deployment. *IEEE Software*, 32(2):64 – 72, 2015.

[15] Aymeric Hemon, Barbara Lyonnet, Frantz Rowe, and Brian Fitzgerald. From agile to devops: Smart skills and collaborations. *INFORMATION SYSTEMS FRONTIERS*, 22(4):927 – 945, 2020.

[16] Paul Dragos. The impact of stakeholders in agile software development. *Annals of the University of Oradea, Economic Science Series*, 30(2):353 – 362, 2021.

[17] Alan S. Koch. *Agile Software Development : Evaluating the Methods for Your Organization.* Artech House Computing Library. Artech House, Inc, 2005.

[18] Joakim Verona, Paul Swartout, and Michael Duffy. *Learning DevOps: Continuously Deliver Better Software.* Packt Publishing, 2016.

[19] Soni Mitesh. *DevOps for Web Development.* Packt Publishing, 2016.

[20] Yilmaz Onur and Akbaş Süleyman. *Introduction to DevOps with Kubernetes : Build Scalable Cloud-native Applications Using DevOps Patterns Created with Kubernetes.* Packt Publishing, 2019.

[21] Verona Joakim. *Practical DevOps.* Community Experience Distilled. Packt Publishing, 2016.

[22] Gene Kim, Jez Humble, Patrick Debois, John Willis, Nicole Forsgren, and John Allspaw. *DevOps handbook : how to create world-class agility, reliability, and security in technology organizations. [electronic book].* IT Revolution Press, 2021.

[23] Vadapalli Sricharan, Sarma Prakash, and Myerscough Jason. *Hands-on DevOps : Explore the Concept of Continuous Delivery and Integrate It with Data Science Concepts.* Packt Publishing, 2017.

[24] Pathania Nikhil. *Learning Continuous Integration with Jenkins.* Packt Publishing, 2016.

[25] Tsoukalos Mihalis. *Mastering Go : Create Golang Production Applications Using Network Libraries, Concurrency, and Advanced Go Data Structures.* Packt Publishing, 2018.

[26] Williams Andrew. *Hands-On GUI Application Development in Go : Build Responsive, Cross-platform, Graphical Applications with the Go Programming Language.* Packt Publishing, 2019.

[27] RUSS COX, ROBERT GRIESEMER, ROB PIKE, IAN LANCE TAYLOR, and KEN THOMPSON. The go programming language and environment. *Communications of the ACM*, 65(5):70 – 78, 2022.

[28] Wilson Eddy. *MERN Quick Start Guide : Build Web Applications with MongoDB, Express.js, React, and Node.* Packt Publishing, 2018.

[29] Hoque Shama. *Full-Stack React Projects : Learn MERN Stack Development by Building Modern Web Apps Using MongoDB, Express, React, and Node.js, 2nd Edition.*, volume Second edition. Packt Publishing, 2020.

[30] B. Laster. *Continuous Integration Vs. Continuous Delivery Vs. Continuous Deployment: The Processes and Tools of Effective Continuous Delivery Pipelines.* O'Reilly Media, 2020.

[31] Li Patrick. *JIRA Agile Essentials.* Professional Expertise Distilled. Packt Publishing, 2015.

[32] Sagar Ravi. *Jira Quick Start Guide : Manage Your Projects Efficiently Using the All-new Jira.* Packt Publishing, 2019.

[33] Harned David. *Hands-On Agile Software Development with JIRA : Design and Manage Software Projects Using the Agile Methodology.* Packt Publishing, 2018.

[34] Li Patrick. *Jira 8 Essentials : Effective Issue Management and Project Tracking with the Latest Jira Features, 5th Edition.*, volume Fifth edition. Packt Publishing, 2019.

[35] *Heroku Cookbook.* Packt Publishing, 2014.

[36] Samuel Greengard. Heroku vs. aws: 2022 cloud platform comparison. *eWeek*, page N.PAG, 2022.

[37] Newswire PR. Heroku receives infoworld's technology of the year award. *PR Newswire US*, 2012.

[38] Hanjura Anubhav. *Heroku Cloud Application Development.* Packt Publishing, 2014.

[39] Espake Patrick. *Learning Heroku Postgres.* Packt Publishing, 2015.

[40] Sasidharan Deepu K and N Sendil Kumar. *Full Stack Development with JHipster : Build Full Stack Applications and Microservices with Spring Boot and Modern JavaScript Frameworks, 2nd Edition.* Packt Publishing, 2020.

[41] Albert Y. Kim, Valentine Herrmann, Ross Bareto, Brianna Calkins, Erika Gonzalez-Akre, Daniel J. Johnson, Jennifer A. Jordan, Lukas Magee, Ian R. McGregor, Nicolle Montero, Karl Novak, Teagan Rogers, Jessica Shue, and Kristina J. Anderson-Teixeira. Implementing github actions continuous integration to reduce error rates in ecological data collection. *METHODS IN ECOLOGY AND EVOLUTION*, 2022.

[42] Sanchit Aggarwal and Jyoti Verma. Comparative analysis of mean stack and mern stack. *International Journal of Recent Research Aspects*, 5(1):133 – 137, 2018.

Appendix A

Jira

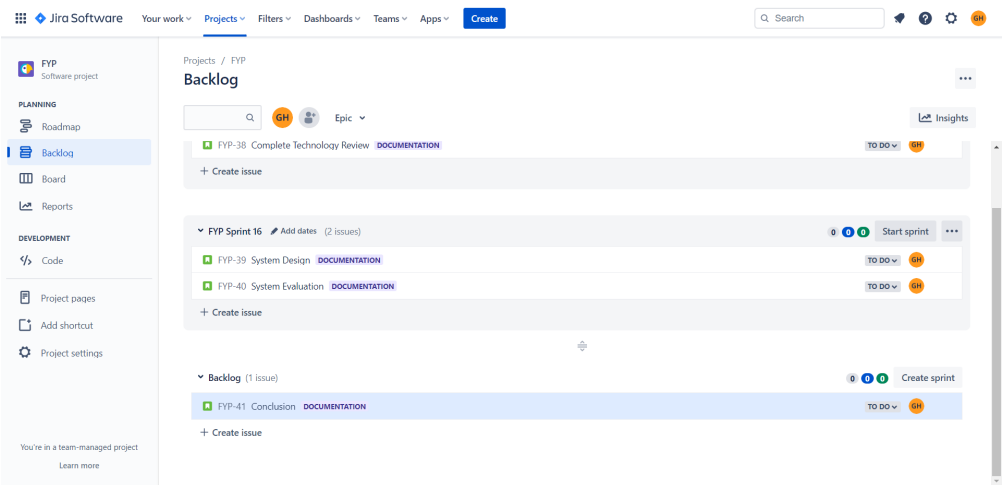

Figure A.1: Jira backlog

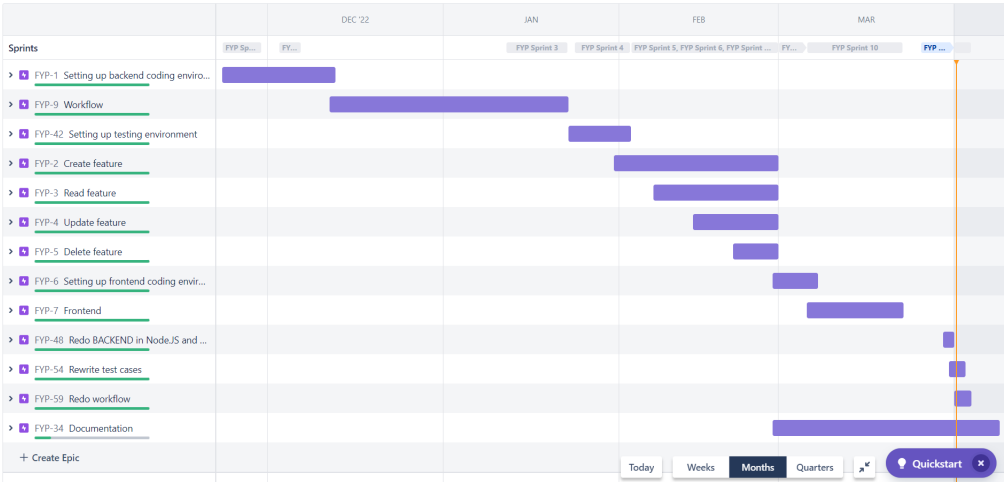

Figure A.2: Jira roadmap

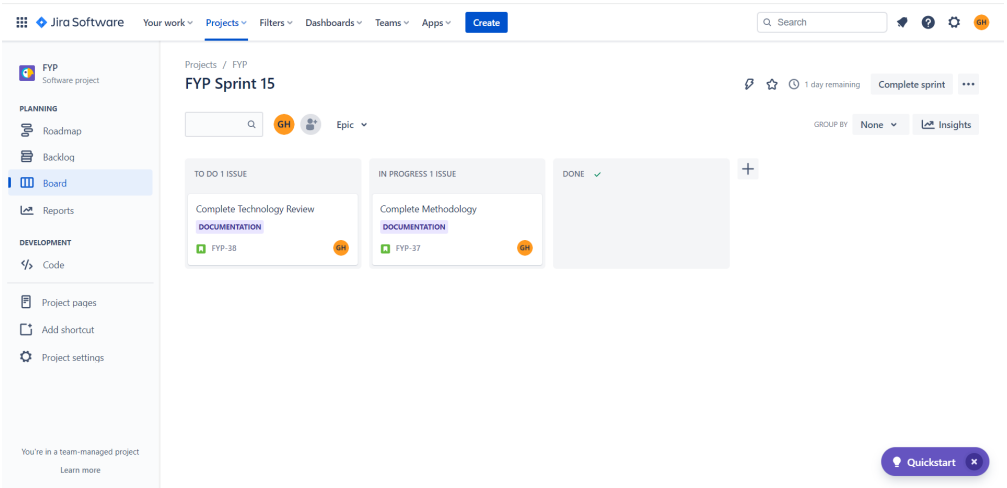

Figure A.3: Jira board

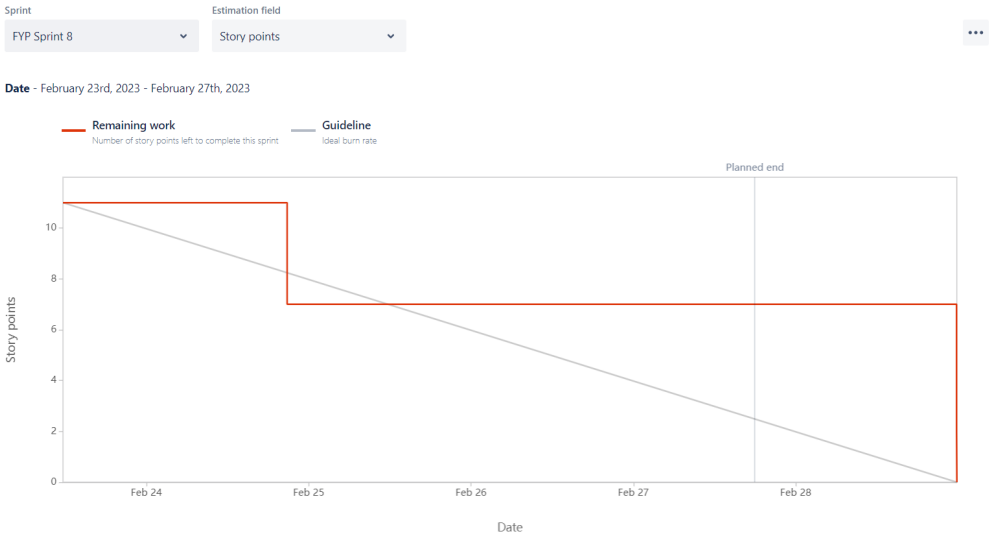

Figure A.4: A Sprint Burndown Chart

Appendix B

Scrum

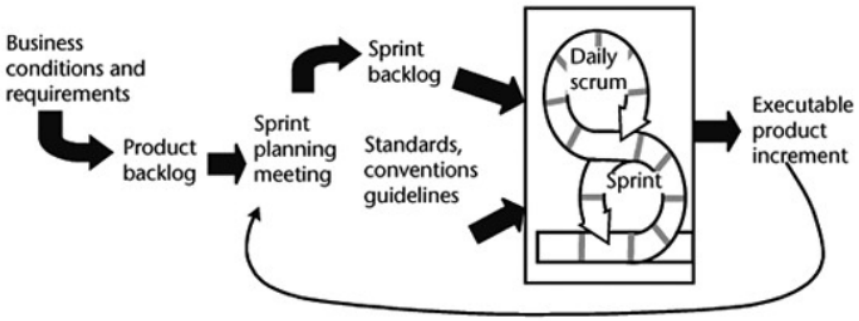

Figure B.1: Scrum Process Diagram

Appendix C

DevOps

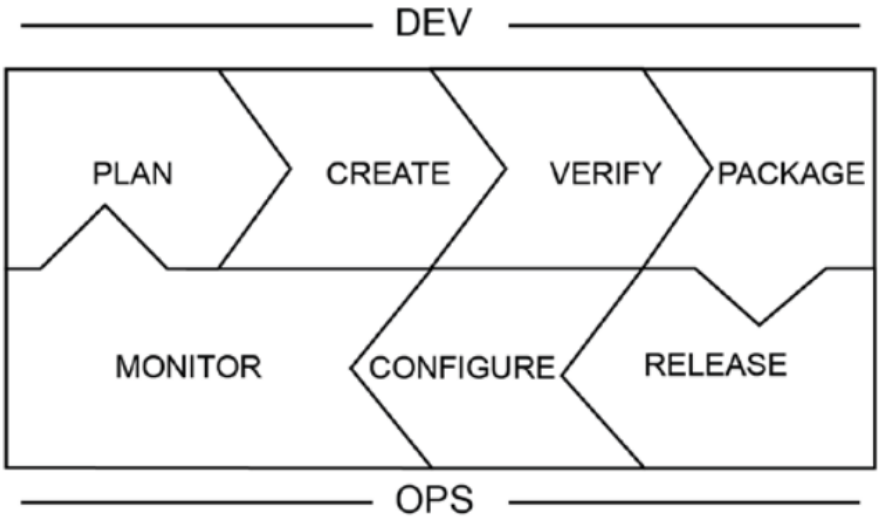

Figure C.1: The DevOps Toolchain

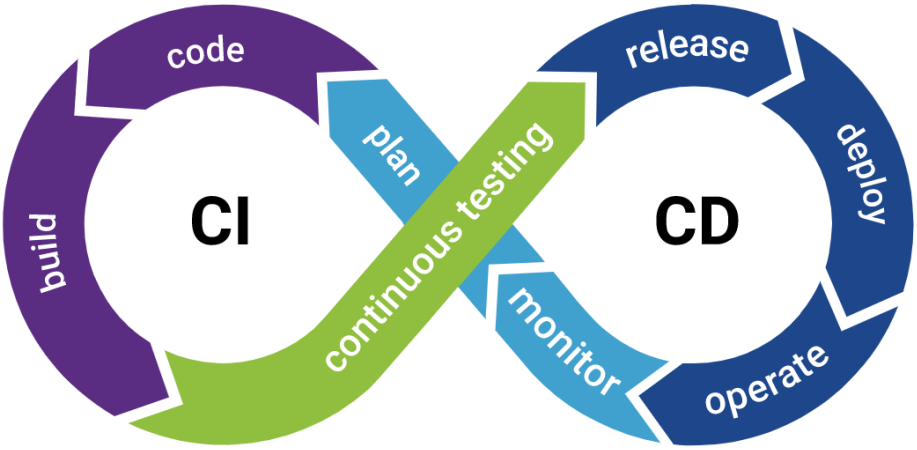

Figure C.2: The CI/CD Process

Appendix D

MERN

- MongoDB: A database based on the document-oriented data model.
- Express.js: A web application framework that allows for the creation of APIs and websites.
- React.js: A JavaScript library for creating user interfaces that takes a declarative, component-based approach.
- Node.js: A JavaScript runtime environment that enables the development of platform-independent servers, tools, and applications.

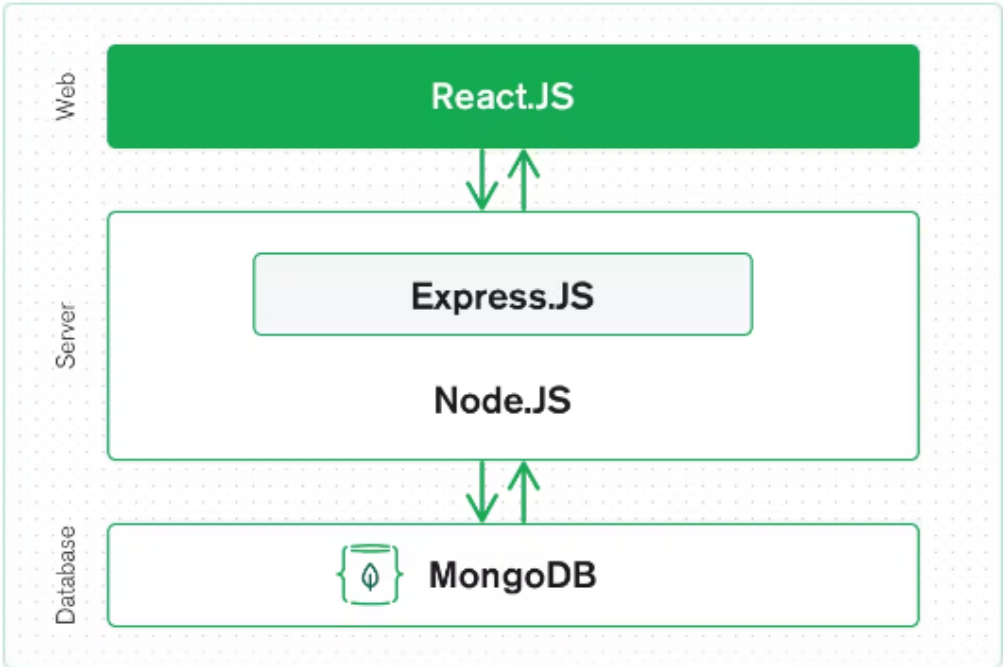

Figure D.1: The MERN Stack Architecture

Appendix E

CI/CD Design and Implementation

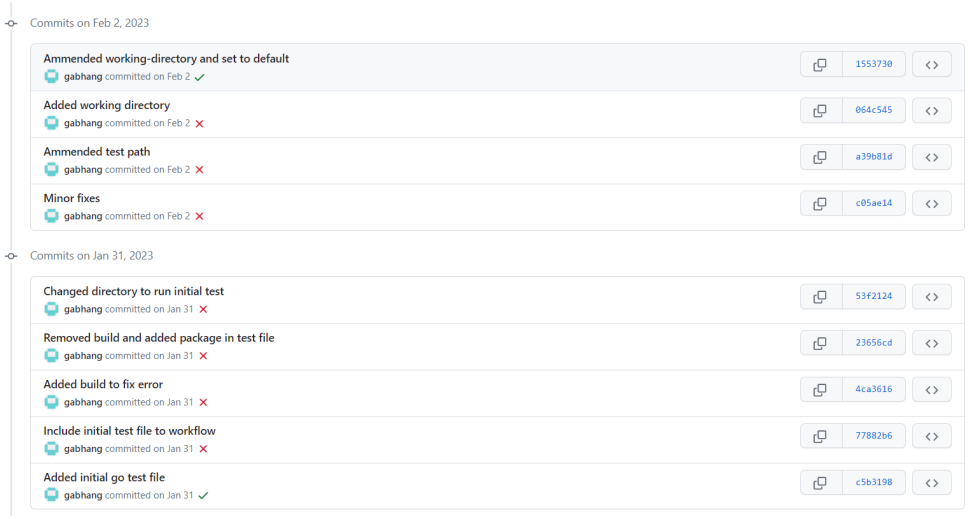

Figure E.1: Evidence of mistakes and corrections during workflow design

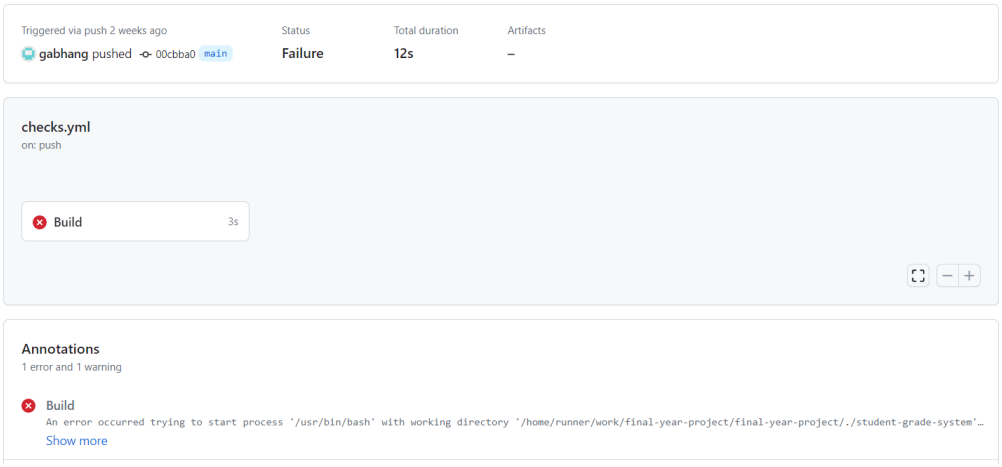

Figure E.2: Directory mismatch error

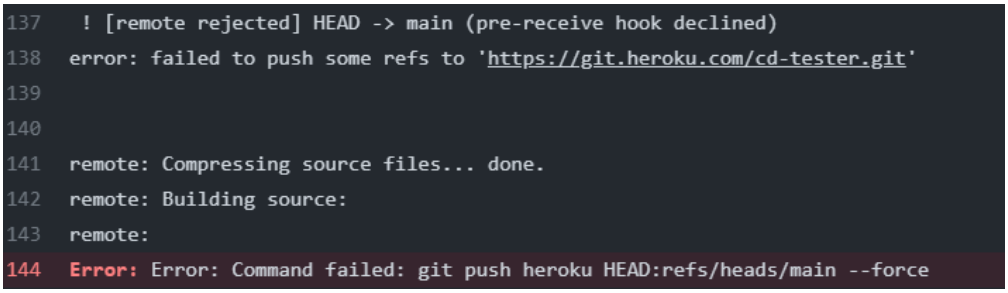

Figure E.3: Deployment rejection from using GO

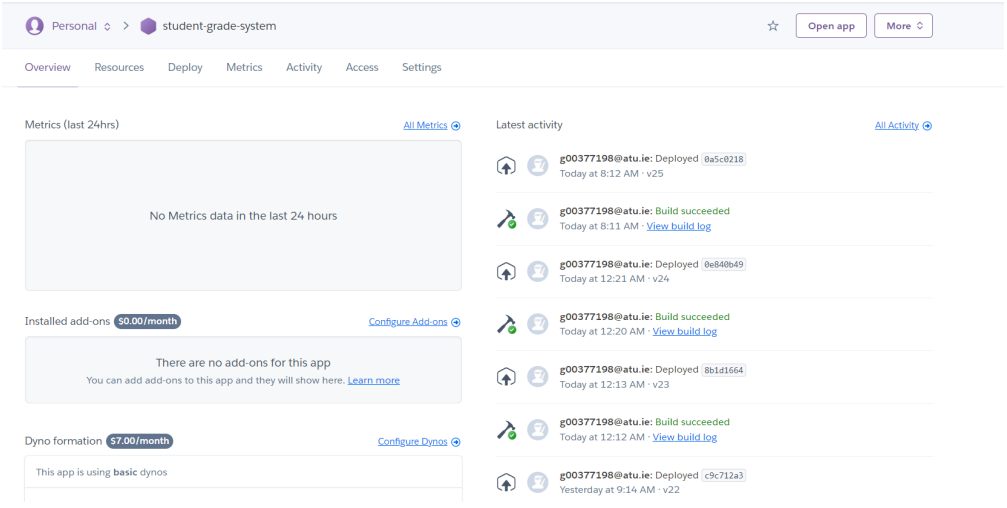

Figure E.4: Successful deployment in Heroku