

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Programme: | MSc Cloud and Network Security |
| Module and title: | CLD7303: DevOps |
| Student ID: | 2424365,2424380,2424171,2408764 |
| Marking Tutor: | Dr. Aamir Abbas |
| Assessment Name: | Practical project-Assessment 1 |
| Submission date: | 14/05/2025 |

This report is submitted in part fulfilment of the MSc degree in Cloud and Network Security at the University of Bolton, Deane Road, BL3 5AB, United Kingdom

I hereby declare that this report represents by own work at the University of Bolton. All sources used have been appropriately cited and referenced.

Date: 14/05/2025

# WEB BASED NOTE APPLICATION

COHORT J GROUP 3

CHIKA VANESSA NWAFOR 2424380

EGBE JONAH CHUKWUDI 2424365 IDENYI JOHN CHUKWUEMEKA 2424171 CHIJIOKE JEREMIAH HENRY 2408764

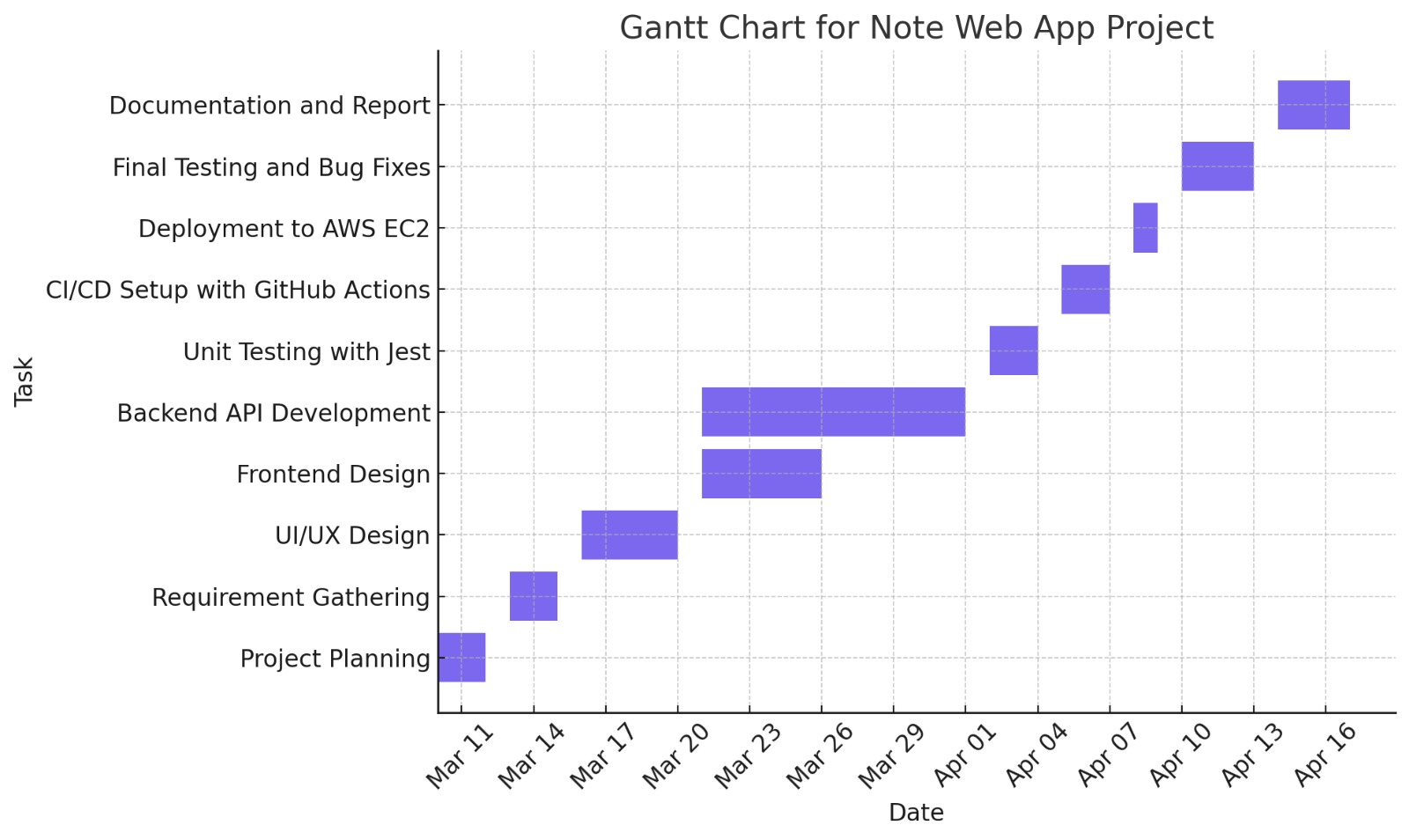
### **Abstract**

This project presents the development of a responsive, full-stack note-taking web application designed to enable users to create, edit, and manage personal notes in real time. The application features a structured user interface that integrates seamlessly with a RESTful API, supporting full CRUD (Create, Read, Update, Delete) operations. User authentication is implemented to ensure secure access, with a login system managing user sessions and data protection. The project followed an Agile Methodological development approach, enabling iterative progress through short sprints and continuous feedback. A Trello-based board was used to manage tasks, organize the backlog, and track progress. Development was divided into cycles that delivered small, functional components incrementally. The frontend was developed using a modern JavaScript framework, emphasizing modular component architecture and dynamic rendering. The backend interfaces with a MongoDB database, facilitating scalable data storage and retrieval. Project management followed an Agile-inspired workflow, with tasks organized via a digital kanban board to ensure iterative development and team coordination. The system prioritizes usability, responsiveness, and maintainability, making it suitable as a foundational model for more advanced personal productivity tools.

# Keywords

* Web application development
* Front-end design
* User interface
* Team collaboration
* Project management
* Note-taking app
* Login system
* Software development process
* Task planning
* Reflective practice

**GANTT CHART**



The Gantt chart outlines the key phases, tasks, and timelines, giving a clear overview of how the project was structured and executed. The project kicked off with Project Planning and Requirement Gathering, helping the team establish a shared understanding of the app’s goals and user needs. Following that, a dedicated phase for UI/UX Design was introduced to create intuitive and user-friendly interfaces before moving into Frontend Design and Backend API Development, which were done in parallel to maximize productivity. After the core development, the team focused on Unit Testing using Jest to ensure code reliability. This was followed by the CI/CD setup using GitHub Actions, where automation pipelines were configured for testing, building, and deploying the application. Once the CI/CD process was in place, the app was deployed to an AWS EC2 instance, enabling real-time access and usage. To wrap up, time was allocated for Final Testing and Bug Fixes to catch any issues before delivery, and then for Documentation and Report Writing to reflect on the project journey and outcomes. The chart effectively shows the overlapping nature of tasks, the logical sequencing of development activities, and the importance of planning and automation in delivering a functioning DevOps project on schedule.

**WORK DIVISION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student ID/Name | Tools Used | Task | Challenges | Deliverables |
| **2424380**          **Chika Vanessa Nwafor** | **Programming Language(s)**: JavaScript, NodeJS.    **Frameworks & Libraries**: ReactJS, Tailwind CSS, Express, Figma.    **Database(s)**: MongoDB.    **DevOps Tools & CI/CD Pipeline**: GitHub, Jenkins, AWS    **Hosting & Deployment**: AWS | Design and develop sign up and login page.   Product Manager – encompasses road map planning, requirement gathering, performance monitoring. | • Limited pages on Figma which may require payment. |  |
| **2408764**        **Henry Chijioke** **Jeremiah** | **Programming Language(s)**: JavaScript, NodeJS.    **Frameworks & Libraries**: ReactJS, Tailwind CSS, Express.    **Database(s)**: MongoDB.    **DevOps Tools & CI/CD Pipeline**: GitHub, Jenkins, AWS    **Hosting & Deployment**: AWS | Database Connection, CRUD implementation,  Search bar implementation.      Backend | N/A |  |
| **2424171**            **John Chukwuemeka Idenyi** | **Programming Language(s)**: JavaScript, NodeJS.    **Frameworks & Libraries**: ReactJS, Tailwind CSS, Express.    **Database(s)**: MongoDB.    **DevOps Tools & CI/CD Pipeline**: GitHub, Jenkins, AWS    **Hosting & Deployment**: AWS | Testing using Jest, CI/CD pipeline implantation | • Major Tailwind update from the creator. |  |
| **2424365**        **Jonah Chukwudi Egbe** | **Programming Language(s)**: JavaScript, NodeJS.    **Frameworks & Libraries**: ReactJS, Tailwind CSS, Express, Figma.    **Database(s)**: MongoDB.    **DevOps Tools & CI/CD Pipeline**: GitHub, Jenkins, AWS    **Hosting & Deployment**: AWS | Designed and developed the Landing page, User dashboard and the overall UI including Signup page.  Deployment of the application on EC2. | • Create more note cards and add the reminder and alarm function |  |

**INTRODUCTION**

**The Importance of Note Taking DevOps-Oriented Web Application for Academic and Corporate Scheduling**

The proliferation of digital learning environments and remote work cultures has transformed how individuals manage their academic and professional responsibilities. Despite these advancements, significant gaps persist in tools that facilitate seamless task tracking and appointment management. Students continue to face difficulties in keeping pace with academic deadlines, managing overlapping assignments, and organizing study routines. Similarly, businesses especially those operating in client-facing domains such as recruitment, consultancy, or healthcare experience operational bottlenecks arising from poor appointment scheduling systems. In both contexts, the lack of efficient, integrated digital tools for notetaking and scheduling represents a systemic barrier to productivity.

Concurrently, the complexity of developing and maintaining responsive, scalable web applications has increased. Traditional software engineering approaches—characterized by siloed development and operations teams, manual deployment processes, and limited post-deployment monitoring are no longer viable in meeting the expectations of modern users. These limitations necessitate a transition towards DevOps methodologies, which fuse software development with IT operations to create a unified, automated, and iterative system of software delivery (Bass et al., 2015; Forsgren et al., 2018).

**BUSINESS SCENARIO AND PROBLEM DEFINITION**

At the core of this study lies a dual-faceted problem which is the need for a centralized, user-friendly web application that caters to both academic and professional time management. From the perspective of students, a recurring issue is the fragmentation of task-tracking methods. While some rely on manual planners, others attempt to piece together their schedules using disparate digital tools such as Google Docs, notepad apps, or calendar integrations. This fragmentation often results in lost information, confusion around deadlines, and ultimately, poor academic performance (Krause et al., 2017).

On the corporate front, especially in small- to medium-sized enterprises (SMEs), there exists a pervasive lack of robust scheduling systems capable of managing multiple client appointments, interviews, and internal tasks simultaneously. Current practices often involve static spreadsheets or proprietary scheduling software that lacks integration, automation, and real-time adaptability. This inefficiency leads to duplicated bookings, missed appointments, and customer dissatisfaction.

The proposed solution is the development of a cloud-based note-taking and scheduling web application, designed to bridge these operational and academic gaps. It is built to facilitate real-time collaboration, cross-platform access, and intelligent reminders, thereby enhancing personal organization and institutional efficiency. Given the complexity of the application, the speed at which updates must be deployed, and the demand for continuous feature enhancement, a DevOps-based development and deployment strategy is essential.

**AIMS AND OBJECTIVES**

The overarching aim of this research and development initiative is to architect a DevOps-aligned workflow for the delivery of a scalable, cloud-based note and scheduling application. This application is intended to serve as a productivity hub for two primary user groups: university students and corporate professionals. Specifically, the project seeks to design a system capable of:

* Ensuring seamless integration of frontend and backend components using a microservices-based architecture.
* Facilitating consistent and reliable deployment through automation tools and continuous integration/continuous deployment (CI/CD) practices.
* Implementing real-time monitoring and alerting to maintain high availability.
* Providing containerized infrastructure for environment parity across development, testing, and production.

Through this integration, the application will not only solve the defined problem but also serve as a case study in effective DevOps implementation in web development.

**DEVOPS SOLUTION AND APPLICATION DEVELOPMENT WORKFLOW**

The application was developed using a full-stack JavaScript framework. The frontend interface was created using Figma and React.js, chosen for its component-based architecture and strong community support. The backend was implemented using Node.js and Express.js, enabling asynchronous, event-driven communication between the user interface and the database. MongoDB was selected as the NoSQL database for its flexibility in handling unstructured data such as notes, task lists, and appointments.

Development began by initializing a Git repository for version control. GitHub was used not only as a code repository but also as a collaborative platform where issues were tracked, branches were managed, and pull requests were reviewed. The development team adopted a feature-branch workflow, wherein each new feature such as calendar integration, rich-text note editing, or user authentication was developed in a separate branch and only merged into the main branch after thorough peer review and successful test execution.

To enforce rigorous code quality standards and streamline the integration process, GitHub Actions was employed to automate CI/CD pipelines. Upon each push to a feature branch or pull request to main, a GitHub Actions workflow was triggered. This workflow executed unit tests written in Jest, followed by integration tests using Postman and Cypress. Successful test completion was a prerequisite for merging code into the production-ready branch.

After merging, the code was automatically built and containerized using Docker. Docker files were crafted for both frontend and backend services, ensuring environment consistency across different stages of the deployment lifecycle. The containers were then pushed to Docker Hub and deployed on Amazon Web Services (AWS) using Elastic Beanstalk for simplified orchestration. Elastic Beanstalk was selected for its capacity to handle scalable deployment with minimal infrastructure management.

Further enhancements included the integration of Prometheus for system monitoring and Grafana for visualizing application health metrics such as CPU usage, memory consumption, and request-response time. Application logs were shipped to the ELK (Elasticsearch, Logstash, and Kibana) stack to aid in debugging and post-deployment analysis.

This integrated DevOps pipeline significantly reduced deployment friction, enabled faster iteration cycles, and ensured the application remained responsive and reliable. Moreover, by storing code in public or private GitHub repositories with detailed documentation and change logs, the development process remained transparent, reproducible, and easily extensible. DevOps has emerged as a transformative approach to software engineering, enabling organizations to enhance efficiency, collaboration, and system reliability. It combines software development (Dev) and IT operations (Ops) to foster continuous integration, continuous delivery (CI/CD), and automated infrastructure management. The core idea behind DevOps is to break down silos between traditionally separate teams and to create a culture of shared responsibility for the development lifecycle (Kim et al., 2016). The need for rapid and reliable software delivery in today’s competitive digital environment has driven organizations to adopt DevOps practices. This implementation typically involves the integration of CI/CD pipelines, infrastructure as code (IaC), containerization, and robust monitoring frameworks. Through these practices, DevOps aims to accelerate deployment frequencies, reduce failure rates, and enable faster recovery from incidents.

In parallel, IaC allows infrastructure to be managed and provisioned using code rather than manual processes. This not only ensures consistency across development, staging, and production environments but also supports version control and auditability. Popular IaC tools include Terraform and AWS CloudFormation, which can automatically deploy entire environments in a repeatable and scalable manner (Winkler, 2016). The use of IaC significantly reduces human error and enhances system reliability.

Containerization, through technologies like Docker, provides lightweight and consistent environments for application development and deployment. When combined with orchestration tools such as Kubernetes, it becomes possible to manage complex application clusters, perform rolling updates, and scale services dynamically in response to demand. This contributes to the high availability and elasticity of applications, which are key characteristics of modern cloud-native systems.

Despite the technical advantages, DevOps implementation is not without challenges. Cultural resistance from teams unaccustomed to rapid iteration and shared responsibilities can hinder adoption. Complex toolchains and steep learning curves also pose a barrier. To mitigate these issues, organizations must invest in training, promote collaboration through agile methodologies, and implement change management strategies to support gradual transformation (Bass, Weber and Zhu, 2015).

The benefits of DevOps are substantial. It enables shorter development cycles, leading to quicker time-to-market for new features and services. It improves system stability through continuous testing and monitoring and fosters a culture of accountability and transparency. Furthermore, the automation of deployment processes reduces operational overhead and frees up resources to focus on innovation.

## **DevOps Lifecycle in the Context of Note Web Application**

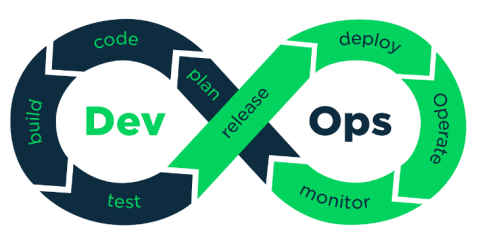


Fig 2: DevOps Architecture – source: [IEEE Xplore Full-Text PDF:](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9142895)

The above image represents the infinity loop format of DevOps lifecycle diagram; we followed in developing and deploying our Note Web App. Each phase emphases the integration of development (Dev) and operations (Ops) team ensuring that our application fulfilled functionality, security, and scalability requirements.

**Stage 1: Plan**

This planning phase involved team group brainstorming and task division to align with the project objectives. We start by identifying the core features of the Note Web App, such as note modifications that includes creation, editing, and deletion, with emphasis on user simplicity and accessibility.

**Stage 2: Code**

We employed industry best practices during our development ensuring that we used a collaborative version control system through Git and GitHub to write a modular, clean, and reusable code. This approach helped us to track our application progress, resolve merge conflicts, and maintain code quality.

**Stage 3: Build**

We built and tested our Note Web App regularly using automation where necessary while integrating new features on our application. This ensured that the codebase was always in a deployable state and reduced integration issues.

**Stage 4: Test**

Automated and manual testing was a key part of our workflows. We validated core functionalities of the Note Web App, such as creating and retrieving notes, to ensure stability. This helped us to maintain high quality and identify bugs early.

**Stage 5: Release**

We were able to consistently and quickly release new updates through CI/CD pipeline. New features or patches are made available through each successful build without delays or service interruptions.

**Stage 6: Deploy**

We utilised AWS EC2 instances to deploy our Note Web App in the Cloud ensuring scalability and cost-effectiveness.

**Stage 7: Operate**

After deployment we monitored our App performance and server health enabling us to proactively fix small problems before they had an impact on user experience.

**Stage 8: Monitor**

We implemented a basic monitoring mechanism to track user interactions and performance. This feedback loop helped us optimise app functionality and enhances usability in subsequent iterations.

**Significance of the DevOps lifecycle on the Note Web App**

The DevOps lifecycle followed a practical roadmap to deliver a functioning, reliable, and user-friendly application. Each stage of the lifecycle contributed to a continuous improvement cycle, fostering collaboration, rapid delivering, and high service quality. The seamless integration of development and operations principle was fundamental to the success of our projects. This approach aligns with Agile methodologies and modern Cloud computing.

**PROJECT/APPLICATION DESCRIPTION**

This is a typical Digital Note Web Application, designed to assist users in creating, organizing, and storing ideas, information, and notes in a manner that is searchable and easily accessible. It focuses on solving the problems of losing track of crucial information, organizing and labelling notes for convenient access, and improving teamwork and productivity.

This app is most useful to students, artists, researchers, and professionals that want a dependable and effective method of organizing their notes and ideas.

Below are the key functionalities of the application:

• User Authentication: Users can sign up, log in, and log out securely.

• Search Bar: Search functionality allows user to locate notes using keywords or tags.

• Note Management: Users can create, edit, and delete notes. Ability to organize notes into categories/tags.

• Cloud: User note or information are stored in MongoDB which is hosted in AWS

• Calendar: The function allows users to link notes to certain dates or deadlines by so doing keeping track of their start/end date. this is ideal for professionals and students managing tasks or assignments.

• Folder creation: This function allows user to create new folders.

**CLOUD PLATFORM**

Our team choose Amazon Web Service (AWS) as the main Cloud provider for the deployment and hosting of the Web-based Note App. AWS provided a secure, cost-efficient and scalable service stem from AWS, (2024) that met the requirements of a modern web application and aligned with our DevOps pipeline. We were able to ensure flexible scaling, automated resources provisioning and continuous availability by utilising AWS service, which are crucial for delivering a reliable and responsive applications.

AWS EC2 which offers a full control of virtual server environment and scalable compute capacity, was used to host the app`s backend services, AWS S3 was used to store static content, like frontend components because of its seamless connection with other AWS services. To further strengthen the system security, AWS Identity and Access Management (IAM) was employed to manage user roles and permissions (AWS, 2024). Other AWS services like CloudWatch and CloudTrail were used for monitoring and logging systems to keep track of user activity and system performance, which helped in compliance auditing and proactive problem solving. These tools are essential for ensuring business trust and quick incident response within DevOps culture (Jha, et al. 2023).

AWS improved operational efficiency, performance, and security across the Software Development Life Cycle (SDLC) by providing a DevOps-friendly environment for the deployment and management of the Web-based Note App. Deployment is carried out on Amazon Web Services (AWS) using Elastic Beanstalk, which orchestrates container provisioning, service scaling, and load balancing. This platform-as-a-service approach abstracts infrastructure concerns and allows developers to focus on application features rather than system administration. Zhang et al. (2010) note that such cloud services facilitate dynamic scaling and provide a robust backbone for high-availability applications. MongoDB Atlas is used as the managed database backend, offering built-in replication, automated backups, and performance analytics.

To ensure the application remains resilient under load and transparent in operation, observability mechanisms have been incorporated. Metrics on request latency, memory usage, and error rates are collected using Prometheus, while Grafana dashboards enable real-time visualization of system health. The importance of proactive observability is well established in the work of Turnbull (2014), who notes that effective monitoring systems can significantly reduce system downtime and aid in capacity planning. Additionally, logs from the application are shipped to the ELK Stack (Elasticsearch, Logstash, and Kibana), providing powerful capabilities for log querying and incident analysis. Jones et al. (2020) argue that such logging frameworks are indispensable in modern site reliability engineering, especially for debugging distributed systems.

**VERSION CONTROL PROCESS AND PROPOSED TOOL**

For version control, our team adopted Git as the core tool and used AWS EC2 as the remote hosting platform. This combination provided us with a powerful and collaborative environment to manage our codebase efficiently. We found Git to be not just a technical necessity but a tool that fostered discipline in how we handled code changes. Every change, no matter how small, was tracked, committed, and pushed in a way that could always be reviewed or rolled back if needed.

Using GitHub allowed our group to work concurrently on different parts of the project without stepping on each other’s toes. We created branches for different features and fixes, which were later merged into the main branch after passing tests and reviews. Pull requests (PRs) were used for code reviews, and they became a central point of discussion and collaboration. This process taught me a lot about peer reviews, constructive feedback, and the value of writing clean, understandable code from the start.

Beyond just code storage, GitHub served as a central integration point for our DevOps pipeline. It was from here that we triggered our test and deployment workflows through GitHub Actions. This integration helped us maintain a consistent and reliable development cycle.

**BUILD/TEST PROCESS AND TOOL**

For building the frontend, we used Vite a modern frontend build tool that made the process fast and developer friendly. Vite helped bundle our frontend assets and prepare them for deployment in a much quicker and more efficient way than older tools like webpack.

In terms of testing, we relied on Jest, a JavaScript testing framework, to handle our unit tests. Jest was chosen because of its simplicity, speed, and compatibility with our JavaScript codebase. We wrote unit tests primarily for our backend logic, ensuring that JWT was properly authenticated before signing up or login. Using tests gave us more confidence in our code, especially when making changes or adding new features.

One of the things we learned from this experience was the importance of writing tests early and integrating them into the development workflow. We made it a rule to ensure tests pass before merging any code into the main branch.

**CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY (CI/CD) PROCESS AND TOOL**

For our CI/CD process, we used GitHub Actions, which can be set up in the project directory or directly on the GitHub actions on the repository. We set up two workflow files in our project,test.yml and deploy.yml. These workflows helped automate key parts of our development pipeline and reduced the manual effort required to test, build and deploy our application.

The test.yml workflow was triggered automatically whenever a pull request was made to the main branch. This workflow ran our Jest unit tests and built the frontend using Vite to ensure that all changes were stable and production-ready before being merged. Setting up this file for continuous integration helps in catching bugs on time and stops broken code from being merged in the main branch.

The deploy.yml workflow, on the other hand, handled the automatic deployment of our application. Whenever a push is made to the main branch, this workflow gets triggered and connects to our AWS EC2 instance to deploy the updated application. To make this connection secure, we stored the EC2 instance's public IP address and private key pair as GitHub repository secrets. This method kept sensitive information out of the codebase while allowing GitHub Actions to access the server securely.

Setting this up was a bit tricky at first, especially configuring the SSH connection and ensuring the EC2 instance could run the deployment commands properly. But once everything was working, it felt incredibly rewarding to see our updates go live automatically after just pushing to the main branch.

## **Personal Reflection: Note Web Application Project**

### **Role and Responsibilities**

**Vanessa**

In this project, I took on a central role as the front-end developer responsible for just designing and implementing key interactive features. My contributions were focused primarily on the user interface, functionality tied to note management, and user authentication. Beyond coding, I also played an active role in project planning and task organization.

### **Front-End Design and Development**

I built the part of the front-end structure of the application using a modern JavaScript framework (React, tailwind CSS). One of my key components was the note card interface, which allows users to add, edit, and delete notes. This component interacts with the backend through API calls to ensure that changes are reflected in real-time. I handled:

* API Integration: Using axios (helps to call API to collect data from back end to send and receive data for note operations
* State Management: Ensuring that the UI updated correctly based on API responses using hooks like useState and useEffect
* Error Handling: Displaying feedback messages in case of failed API calls
* Component Reusability: Building modular UI components that can be reused for consistency and maintainability

This hands-on work greatly improved my skills in JavaScript, React, and RESTful integration.

### **Login Page and Authentication**

I also developed the login page, which involved:

* Designing form elements such as input fields and login buttons
* Managing form state and validating user input
* Sending login credentials to the backend and storing the authentication token securely (e.g., in local storage)
* Handling redirection after successful login and displaying error messages for failed attempts

This task taught me a lot about secure front-end handling of authentication, and how front-end and back-end layers interact in a secure login flow.

### **Project Management and Collaboration**

In addition to coding, I actively contributed to project organization. We used a Trello board to manage tasks across the team. I helped break the project down into smaller features, assigned deadlines, and updated the board regularly to reflect progress.

This involvement taught me how to stay organized in a team setting, how to follow a development workflow (similar to Agile), and how to track issues or blockers that arise during a sprint. It also kept communication clear and made collaboration smoother.

### **API Understanding and Backend Communication**

Although I didn’t build the backend myself, I made it a priority to understand how our API worked. I learned:

* The difference between HTTP methods (GET, POST, PUT, DELETE)
* How endpoints are structured and how to format requests
* How to handle errors and parse responses for use in the front-end

Understanding these principles allowed me to write more reliable and user-friendly front-end code, and also made collaboration with back-end developers more effective.

### **MongoDB and Data Flow Awareness**

On the database side, we used MongoDB, and while I didn’t write queries directly, I learned how the data was structured in the database and how it traveled from MongoDB to the front-end via the API. This understanding helped me:

* Work more efficiently with note IDs and timestamps
* Format and display content in a user-friendly way
* Anticipate data-related issues like null fields or missing entries

This gave me a deeper appreciation for how NoSQL databases work in real-time applications.

### **Technical Challenges and Solutions**

Some of the challenges I faced included:

* Dealing with asynchronous logic (e.g., waiting for data before rendering components)
* Preventing state bugs when quickly adding or deleting multiple notes
* Making the UI responsive across different screen sizes
* Ensuring form inputs didn’t break with empty fields or invalid formats

I solved these issues by carefully structuring API logic, using loading states, validating inputs, and leveraging tools like browser dev tools and console logs for debugging.

### **Skills and Knowledge Gained**

This project helped me grow significantly in several areas:

* React framework development
* JavaScript ES6+ features
* API communication and integration
* UI/UX design fundamentals
* Collaboration and workflow management

I also became more confident working in a full-stack environment, where front-end and back-end development are tightly connected.

**John**

**Personal Reflections**

My team and I were tasked with developing a note-taking web application to help people manage their thoughts and ensure they don't forget important tasks. The project was designed with a simple but crucial purpose: to make note-taking more efficient and accessible. As I worked on the backend of the application, focusing primarily on building APIs, I found myself not only improving my technical skills but also learning important lessons about collaboration and problem-solving in a real-world environment.

One of the challenges I faced during this project was setting up the Continuous Integration/Continuous Deployment (CI/CD) pipeline for the Note Web App. I had never worked with CI/CD before, and the process was not as straightforward as I initially thought. The pipeline setup involved configuring CI/CD tools like GitHub Actions to automate the test and deployment process, but I ran into an issue when it came to integrating the environment variables into the system. The challenge arose because the VM required access to the environment variables that weren't directly available in the repository.

To solve this issue, I created an .env file within the backend directory of my project to ensure that the environment variables could be accessed during deployment. This small decision, while seemingly simple, saved us from a lot of issues, highlighting the importance of knowing cloudconfiguration and understanding how the project is supposed to work in production.

Using GitHub Actions for configuring my CI/CD was another important part of the experience. GitHub Actions provided an easy way for automating the testing, building, and deployment of our project. However, deploying our application automatically to an EC2 instance in the cloud was a challenge because I haven’t worked with cloud platforms before. For deployment to happen automatically through GitHub Actions, I needed to ensure that I could securely store the public IP and private key pair for the EC2 instance in GitHub Secrets. This allowed the system to access the EC2 instance and deploy the latest build of the project directly from GitHub without logging in manually.

Throughout this project, I realized the importance of communication and collaboration in a team environment. Although I worked primarily on the backend, the project required constant interaction with the frontend team to ensure that the API endpoints were integrated smoothly. We had to agree on data formats, UI design and overall project architecture which taught me how important it is for teams to collaborate effectively. In any tech-related project, it's not enough to simply focus on individual tasks, you have to consider the bigger picture and work closely with others to ensure that every aspect of the project functions seamlessly.

The experience taught me how much work goes into testing and deployments, and why you should automate the process. It might be difficult to set up at the beginning, but it is important in saving developers and engineers time and also in ensuring that the application works efficiently and securely. This project also helped me refine my troubleshooting and debugging skills, particularly when working with complex DevOps tools and environments. I learned how to identify and fix issues within the pipeline, ensuring that each deployment was successful and seamless.

**Learning Outcomes**

This DevOps course has been instrumental in shaping my understanding of modern software development practices. Prior to taking this course, I had a general idea of what DevOps entailed, but I was not fully aware of the depth and significance of its role within the software development lifecycle. Through the course, I gained a clear and practical understanding of how DevOps practices are applied to real-world projects, and how essential they are for improving both the efficiency and quality of software development.

The first thing I learnt from the course was the importance of software development lifecycle(SDLC). I had previously learnt the concept of SDLC, but I didn’t fully know its iterative nature or the significance of agile methodologies within it. The course provided a detailed overview of how agile technologies work, and how they are key to the continuous improvement and flexibility required in modern software projects. Agile methodologies, which emphasize short, iterative development cycles and constant feedback, are closely tied to DevOps practices. This understanding helped me appreciate how teams can quickly adapt to changing requirements and deliver value to customers in an efficient and timely manner.

Another key learning outcome was understanding the role of DevOps solutions in accelerating quick release of products. DevOps enables teams to integrate and deploy code more frequently and more reliably than traditional software development methods. Through the course, I learned how the integration of automated testing, continuous integration, and continuous deployment helps to streamline the development pipeline and reduce the time to market for new features or fixes. DevOps emphasis on rapid deployment and collaboration is important in today’s competitive tech landscape, where the speed of innovation can make or break a product's success.

The course also expanded my knowledge of cloud platforms such as AWS and Azure. I had previously never used any cloud platform, but the course provided hands-on experience in working with these services. Specifically, I learned how cloud instances can be provisioned, managed, and deployed. This was important to the project I worked on, where I was tasked with deploying an application to an EC2 instance. Understanding how these cloud platforms workshelped me troubleshoot deployment issues.

Lastly, my knowledge of Node.js was further refined during the course. As part of the backend development for the note-taking application, I used Node.js to build and manage the backend API. Through this project, I improved my skills in asynchronous programming, API design, and handling server-side requests. I also gained insight into how Node.js interacts with other parts of the application, such as the frontend, and how important it is to ensure smooth communication between different layers of the application.

**CHIJIOKE**

As a backend Engineer on the group note web application, it was both a rewarding and challenging experience. my role spread across database implementation and the project backend but not limited to the frontend. Configuring the database connection with mongoDB was like a road to me at first since was familiar working with mysql but the moment i dived into it, and the integration was seamless. configuring the MongoDB cluster, i created a username and password which i then linked to my config.js file. overcoming this really solidified my ability to manage cloud base database and guaranteeing reliable data access for the application. Also, implementing the CRUD functionality endpoint for note management involves writing expressJS route to manage frontend request and making sure MongoDB queries were optimised. My knowledge of database optimization and RESTful API design has improved as a result of this practical effort, and it was very fulfilling to watch these endpoints effectively power the app's main functions. My experience with the back-end architecture and MongoDB connection forced me to hone my technical abilities, adjust to challenging situations, and recognise the value of cross-functional cooperation.

**REFRENCE**

Amazon Web Service. (2024) Overview of Amazon Web Services: AWS Whitepaper. [Online] pp. 1 – 163. Available from: [Overview of Amazon Web Services - AWS Whitepaper](https://euc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en-GB&rs=en-US&wopisrc=https%3A%2F%2Fboltonuni-my.sharepoint.com%2Fpersonal%2Fcvn1crt_bolton_ac_uk%2F_vti_bin%2Fwopi.ashx%2Ffiles%2Fdb31f62a1b234e92814d6d6a98d16c55&wdorigin=BrowserReload&wdprevioussessionsrc=HarmonyWeb&wdprevioussession=45f3d2e0-b066-40b6-8e74-6e95508a5ca5&wdenableroaming=1&mscc=1&wdodb=1&hid=9C1E9CA1-50EC-C000-51A9-4BAE7EE9AA0B.0&uih=sharepointcom&wdlcid=en-GB&jsapi=1&jsapiver=v2&corrid=d14ded92-2c37-f3a2-b42a-47363d26ae36&usid=d14ded92-2c37-f3a2-b42a-47363d26ae36&newsession=1&sftc=1&uihit=docaspx&muv=1&ats=PairwiseBroker&cac=1&sams=1&mtf=1&sfp=1&sdp=1&hch=1&hwfh=1&dchat=1&sc=%7B%22pmo%22%3A%22https%3A%2F%2Fboltonuni-my.sharepoint.com%22%2C%22pmshare%22%3Atrue%7D&ctp=LeastProtected&rct=Normal&wdhostclicktime=1746701254926&afdflight=96&csc=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush#introduction). [Accessed 08 April 2025].

Bass, Len. (2023) DevOps: A Software Architect's Perspective. [online] pp. 1 – 239. Available from: [Full details and actions for DevOps: A Software Architect's Perspective](https://www-vlebooks-com.ezproxy.bolton.ac.uk/Product/Index/747819?page=0&startBookmarkId=-1). [Accessed 18 Apr. 2025].

Bass, L., Weber, I. and Zhu, L. (2015) *DevOps: A Software Architect’s Perspective*. Boston: Addison-Wesley.

Cherukuri, B., R. (2024) Containerization in cloud computing: comparing Docker and Kubernetes for scalable web applications. International Journal of Science and Research Archive, [Online] 13(01), pp. 1 – 14. Available from: [IJSRA-2024-2035-libre.pdf](https://d1wqtxts1xzle7.cloudfront.net/119319887/IJSRA-2024-2035-libre.pdf?1730580856=&response-content-disposition=inline%3B+filename%3DContainerization_in_cloud_computing_comp.pdf&Expires=1746515828&Signature=TyrR1HbXLUZ6r~99hdKTeYosmzLVbFQ1tk58YUvQa~ZvF4Tx2Sg22sg2gpUBPKN3C4F1OlztxQMHF9inRP7FLNbv1xLNDsocXqP6Aos~34src0gudxUWWB0gOJhJZowrkZFy0DFeXafl3TgIskvMiisowQeeI3UKteCG7xZiWjoH00686vzCHLgXNGiHzR-eW37rL7FomahCBgZDQrOt0jHykfhaYOLgB3PwD0dAGu88VUi-MiD4HQPHESqFZuZdCGakNmnQNfp5NFZc1wLlIEmuoMAJZngLmis6umNTZlzqHQn7LbWAEQxnzVDCL2PlC4If2QOZfEszupGQumpZkA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA). [Accessed 14 April 2025].

Dominguez-Mayo, F. J., Escalona, M. J., Mejías, M. and García-García, J. A. (2020) ‘A systematic mapping study on front-end web development frameworks’, *Journal of Systems and Software*, 169, p. 110710.

Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R. and Safina, L. (2017) ‘Microservices: Yesterday, Today, and Tomorrow’, in *Present and Ulterior Software Engineering*. Cham: Springer, pp. 195–216.

Duvall, P. M., Matyas, S. and Glover, A. (2007) *Continuous Integration: Improving Software Quality and Reducing Risk*. Boston: Addison-Wesley.

Fowler, M. and Sadalage, P. (2012) *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Boston: Addison-Wesley.

Humble, J. and Farley, D. (2010) *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Boston: Addison-Wesley.

Jha, A. V., Teri, R., Verma, S., Tarafder, S., Bhowmik, W., Mishra, S. k., Appasani, B., Srinivasulu, A., and Philibert, N. (2023). From theory to practice: Understanding DevOps culture and mindset. Cogent Engineering, [Online] 10(01). Available from: [Full article: From theory to practice: Understanding DevOps culture and mindset](https://euc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en-GB&rs=en-US&wopisrc=https%3A%2F%2Fboltonuni-my.sharepoint.com%2Fpersonal%2Fcvn1crt_bolton_ac_uk%2F_vti_bin%2Fwopi.ashx%2Ffiles%2Fdb31f62a1b234e92814d6d6a98d16c55&wdorigin=BrowserReload&wdprevioussessionsrc=HarmonyWeb&wdprevioussession=45f3d2e0-b066-40b6-8e74-6e95508a5ca5&wdenableroaming=1&mscc=1&wdodb=1&hid=9C1E9CA1-50EC-C000-51A9-4BAE7EE9AA0B.0&uih=sharepointcom&wdlcid=en-GB&jsapi=1&jsapiver=v2&corrid=d14ded92-2c37-f3a2-b42a-47363d26ae36&usid=d14ded92-2c37-f3a2-b42a-47363d26ae36&newsession=1&sftc=1&uihit=docaspx&muv=1&ats=PairwiseBroker&cac=1&sams=1&mtf=1&sfp=1&sdp=1&hch=1&hwfh=1&dchat=1&sc=%7B%22pmo%22%3A%22https%3A%2F%2Fboltonuni-my.sharepoint.com%22%2C%22pmshare%22%3Atrue%7D&ctp=LeastProtected&rct=Normal&wdhostclicktime=1746701254926&afdflight=96&csc=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush#abstract). [Accessed 10 April 2025].

Jones, B., Burns, A., Hohpe, G., Neward, T. and Nygard, M. (2020) *97 Things Every SRE Should Know: Collective Wisdom from the Experts*. Sebastopol: O’Reilly Media.

Kim, G., Humble, J., Debois, P., Willis, J. and Forsgren, N. (2021) The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations, 2nd ed. [Online] Available from: [The DevOps Handbook: How to Create World-Class Agility, Reliability ... - Gene Kim, Jez Humble, Patrick Debois, John Willis, Nicole Forsgren - Google Books](https://euc-word-edit.officeapps.live.com/we/wordeditorframe.aspx?ui=en-GB&rs=en-US&wopisrc=https%3A%2F%2Fboltonuni-my.sharepoint.com%2Fpersonal%2Fcvn1crt_bolton_ac_uk%2F_vti_bin%2Fwopi.ashx%2Ffiles%2Fdb31f62a1b234e92814d6d6a98d16c55&wdorigin=BrowserReload&wdprevioussessionsrc=HarmonyWeb&wdprevioussession=45f3d2e0-b066-40b6-8e74-6e95508a5ca5&wdenableroaming=1&mscc=1&wdodb=1&hid=9C1E9CA1-50EC-C000-51A9-4BAE7EE9AA0B.0&uih=sharepointcom&wdlcid=en-GB&jsapi=1&jsapiver=v2&corrid=d14ded92-2c37-f3a2-b42a-47363d26ae36&usid=d14ded92-2c37-f3a2-b42a-47363d26ae36&newsession=1&sftc=1&uihit=docaspx&muv=1&ats=PairwiseBroker&cac=1&sams=1&mtf=1&sfp=1&sdp=1&hch=1&hwfh=1&dchat=1&sc=%7B%22pmo%22%3A%22https%3A%2F%2Fboltonuni-my.sharepoint.com%22%2C%22pmshare%22%3Atrue%7D&ctp=LeastProtected&rct=Normal&wdhostclicktime=1746701254926&afdflight=96&csc=1&instantedit=1&wopicomplete=1&wdredirectionreason=Unified_SingleFlush#v=onepage&q&f=false). [Accessed 10 April 2025].

Kritikos, K., Pahl, C., Xiong, H., Bush, I. and Loiret, F. (2019) ‘A survey on service composition middleware in cloud computing’, *Journal of Systems and Software*, 149, pp. 138–169.

Merkel, D. (2014) ‘Docker: Lightweight Linux containers for consistent development and deployment’, *Linux Journal*, 2014(239), p. 2.

Ugwueze, V. U., and Chukwunweike, J., N. (2024) Continuous Integration and Deployment Strategies for Streamlined DevOps in Software Engineering and Application Delivery. International Journal of Computer Applications Technology and Research, [Online] 14(01), pp. 1 – 24. Available from: [Continuous-Integration-and-Deployment-Strategies-for-Streamlined-DevOps-in-Software-Engineering-and-Application-Delivery.pdf](https://www.researchgate.net/profile/Joseph-Chukwunweike/publication/387601097_Continuous_Integration_and_Deployment_Strategies_for_Streamlined_DevOps_in_Software_Engineering_and_Application_Delivery/links/67752a7fc1b013546506a3c5/Continuous-Integration-and-Deployment-Strategies-for-Streamlined-DevOps-in-Software-Engineering-and-Application-Delivery.pdf). [Accessed 14 April 2025].

Zhang, Q., Cheng, L. and Boutaba, R. (2010) ‘Cloud computing: state-of-the-art and research challenges’, *Journal of Internet Services and Applications*, 1(1), pp. 7–18.

Turnbull, J. (2014) *The Art of Monitoring*. Sydney: Turnbull Press.