DevOps Strategy

We approached the stakeholders, the process that we came up with was to introduce DevOps to the company. The reason for DevOps is it will bring a strong community/culture to the company. The process of DevOps takes away focus from the individual performances and department performances who don't work well with others and aims to focus on creating a strong workflow between business, development, operations, and customers. (Image Three)

The process of DevOps offers a feedback loop of a fail-fast mindset. With the fail-fast mindset process that DevOps offers you are preventing employees from spending too long working on a task and losing time working on said task. You also prevent repeats of problems occurring. This is done through the feedback loop as it offers feedback as early as possible and allows teams to sharing knowledge where needed and reducing work to create faster workflows.

Unplan work can led to staff anxiety and disrupts innovation within the team and individuals DevOps encourages unplanned work; this provides value to any company. It does this by been agile and factoring in time or been able to move other task around to complete any unplanned work that might arise. instead of been held back and losing track of project targets that could leave the company at a loss.

DevOps offers a culture of continual learning and experimentation of new ideas. DevOps encourages a safe culture of failure; failure can teach you a lot of lessons and DevOps encourages that.

For the business aspect of DevOps, it offers quick releases which will lead to less of a cost when developing a product and it offers productive teamwork. Speed is needed to deliver projects on time and to the customer. DevOps offers frequent delivery cycles allowing for faster delivery of new features and updates to a product.

The continuous integration of DevOps leads to safe and quality outputs. Resulting in good feedback from end users. As the name suggest DevOps is a cultural that focuses on collaboration between development and operations which reduces inefficiency and time complexity.

Benefits of doing this is that it reduces work in progress, removes bottlenecks, cuts down on technical debt and offers a continues cycle of trying to simplify and improve workflow. It lets the company know the larger goals that each team is working towards to complete and solve together, this includes both the business and the IT goals. each department gets to understand the role that the other departments play and how everything pieces together. This allows for different teams to get an understanding of what is and isn't working. the work gets delivered in the pursuit of a common goal instead of fixating on individual goals with no context. (Ben Lutkevich 2020)

Modernization of Code Strategy

The aim of this is to add value to the code, reduce defects and allow developers to work faster. For the moderation of the code, we are going to tackle it in three areas refactoring, unit tests and ci/cd pipeline. This will improve the standards of the code and offer quicker delivery to the customer.

Refactoring is a skill and can take many hours to learn. You need to develop a keen eye for bad code smells in your codebase. Ideally this would need training for new developers, but a senior coder could be able to "sniff" out any easy to find code smell. The benefit of refactoring your codebase is that it is easier to read, sometimes comments will not be needed to be added as the code will be easy to read also make it easier to implement changes in your code base. The fear of making one change in a large code base can lead to a defect which when fixed is followed up by another defect caused by the fix. Code that is highly coupled together can lead to a domino effect of bugs once a change is introduced. By refactoring the code and making it more modular it makes it easier to introduce changes and new features can be introduced quicker and without introducing bugs.

Introducing Unit tests into any code base is a good strategy as it a great way to test that small functionality of your codebase to make sure it's working as expected before the users get to use it. Ideally you do want 80% code coverage. Unit testing is also acts as a documentation of the product. Developers can look at the unit tests to get the basic of logic needed for each functionality been tested. As this leads to less error it also leads to less time spent on debugging for developers trying to solve defects and reduces bug fixing cost. (Oksana Mikhalchuk 2020), (Image Two)

Moderation of the code is much bigger process then just how the code is structured and testing. When we create the code, our aim is to have a stratifying experience for the end user. To offer this to the customer we are going to implement a CI/CD pipeline. This act of modernize has less to do with the code itself but what happens to the code when section of the code has been change or new features have been added to the codebase.

The CI/CD Pipeline in short automates the software delivery process, it will build the code, then run the tests and deploy a new version of the application. The CI stand for the continuous Integration where developers will merge code into a central repository multiple times a day. CD stands for continuous Delivery which will automate the entire software release process of our application. With CI a commit into the codebase triggers an automated build and test sequence of the giving project providing feedback for the developer who made the change. This feedback might be that the project had a build failure, or a unit test has failed. This quick feed back loop saves cost as less time is spent working on defects. CD consists of infrastructure provisioning and deployment which may be manual and can consists of multiple stages. if a pipeline is not automated the developer would still need to take the same steps but manually.

The benefits of the CI/CD pipeline are that stockholders have access to the latest version of the latest release of the application. Developers can stay working on writing code and have the CI monitoring their changes for error, logs of old code changes are available for review so it offers an accessible history, updating the application is less stressful, rolling back to a previous update or change in the application is not a problem can be done very easily so users won't be impacted. This fast feedback loop helps to build a strong culture of learning and responsibility

Implementation

The company is losing money and staff have been laid off, this can happen in a company which remains static in their tech and business model. This is what is happening in Shinty Software. A new approach is needed to keep the company afloat and hopefully turn the company around back to making a profit. To do this business/development needs to be convinced that the old processes that's been currently used within the company is no longer working and a new methodology needs to be applied to move the company forward. The proposal been proposed is the DevOps methodology. DevOps is the Bridge between development and Operations in a company, its goal is to improve productivity throughout a project. It brings deployment and release activities closer to development and testing. Resulting in cost reduction spent on delivery.

There are many good reasons to adopt the DevOps methodology here in Shinty Software. As we are a company that is losing staff and profit, DevOps can increase the time saved getting new features/updates out to our clients. DevOps aims to deliver faster deployments. Over time the DevOps methodology leads to increase in software lifecycle predictability which can be used for project planning by business to scale projects more accurately in the future.

Feedback is another great strength of DevOps as DevOps encourages an atmosphere of mutual collaboration, communication and integration making project work more transparent as individuals work on common goals. DevOps provides short development lifecycle which will lead to fast and frequent delivery and the shorter development cycles can build a corporate culture of innovations as results are being pushed out rapidly and can increase motivation levels throughout the company building a stronger team-engagement . The benefits of short development cycles are a quick feedback cycle, small batch sizes are at lower risk and easier to test and spot defects which lead to faster error fixes and encourages an efficient feedback loop between operations and developers resulting in continues delivery.(Mohammad Adil 2022)

From these changes that DevOps brings will increase the customer experiences, can improve deployment frequency, recovery time and lower the chances of failure rate. The continues monitoring and the continues testing of code offers early defect detection this will improve the overall build quality of the code. Continuous release and deployment resulting in delivering quality software reduce go to market timelines and leading to a shorter release cycle this is all because DevOps aims to automate as much as the work as possible by creating a pipeline. Overall, DevOps promotes better efficiency, higher quality, reduces general cost and faster & continuous releases, these are some of the benefits that shinty software will make when moving to DevOps.

Unfortunately, here in Shinty Software the code hasn't been keeping up with good coding standards, this does happen with legacy code, methods become too large, code isn't broken up into modules, classes take multiple methods that don't belong. A codebase should look like it's been worked on by one person, but in reality, it's been worked on by a team that uses the same standards throughout the codebase. When the standard drops you can inherit technical debt, and this can slow down development. Ren having worked on one project for his working life knows the importance of coding standards, and Jalen just fresh out of college does not. To get familiar with the codebase both will start off by refactoring the code with Ren reviewing Jalen work with pull requests.

Technical debt has many tells such as, poor naming of methods that prevents a developer from easily understanding what the method is supposed to be doing. God classes, where one class seems to do everything instead of having multiple classes been their own instance within the codes. Modernization process of the existing code, for this a new process must be put into place. Most coding language have coding standard that should be used throughout the team. A version control should be setup(if not already in place) with the codebase and good version control practices should be agreed upon beforehand. Such as commit frequently, commit often. Add commit messages that are short and to the point, not long paragraphs for each commit. Clearly explaining the change been committing. And each developer should be working from a feature branch so that no code is added to the main branch without first been reviewed by another colleague.

With the code there is a lot of improvements that can be made that will help development, such as Added more descriptive method names, good code should allow the developer to read the code and be able to understand what the code is doing even without the help of comment. methods should only do one thing, so breaking down the code into smaller blocks with detailed method names will help anyone who looks at the code to read it easier. Separating the code into classes to make it more modules. Also, a good codebase should have test cases and a test suite to confirm that the code is running as expected. These steps will help structure the project, make the code more reusable, making it easier to read when adding new features and help in debugging. Refactoring can be a large and time-consuming step plus the goal is to demonstrate how and DevOps can help with, and time is not something we have. The idea of both developers working on the refactoring is to get them familiar with the code and other tools that they will be using such as version control.(Kseniia Goldovska 2022)

The real changes to the company and our team will come from applying DevOps methodology, so once the team is somewhat familiar with the codebase it would be good to focus on the DevOps work that needs to be carried out to show over time the improvements to how we work as a team and move forward as a company. Jalen, been a new graduate has exposer to a wide and modern stack of technologies and he is a strong asset to share his knowledge with the team. He might not be familiar with large applications, but he will work with Ren to integrate the new tech into our current application. Jalen is the one with the knowledge of working with the new stack, but Ren knows the importance of good practices with any technology. A modern version control should be added to the code base if not already in use, following the process of adding any new changes to a feature branch with the aim to of making small commits with short and to the point commit messages. Each feature branch should be reviewed before moved into the main branch.

once the version control is selected and pushed into the main branches, we will need a way to make sure nothing is broken with in our application, we can apply continuous integration this will make sure are application builds and tests cases are running. The commits will tracker an automated workflow, this will alert the developers of any issues in their changes. Pushing small changes like this can prevent merge conflicts. if any of the test fail or the build doesn't run none of the code will be deployed to the and a message will be giving back. The messaging acts as feedback to our developers, monitors the steps need to get our code deployed. If there are no issues the code will deploy this is continues deployment.

The workload can be managed by using an agile methodology this is based on the idea of creating Features and the features can be broken down into small tasks called stories. By using the agile methodology tasks are tracked based on story points the higher the point the more time consuming that story might be. If a story is too large it should be broken down into smaller stories. All this happens with in a timeframe known as a sprint. The breakdown of story points gives a good representation of the amount of work that can be done within a sprint. Everyone understand what's expected to be delivered in the time frame of a sprint. Feedback will be shared in Daily Standup this is basically a short 15min meeting updating the rest of the team what you might have finished, what you are currently working on and what you plan to work on next. This offers a chance for every team member to be heard and ask questions to the rest of the team if they have any blockers. this will lead to developers not been stuck too long on a certain task and been able to complete their work faster. Also, at the end of a sprint there is a sprint retrospective in which every team member should provide feedback on what went well, what could have gone better and suggestion on how to improve the current flow of work. This is all valuable information to business to track the current work been completed and figuring out how to improve the process. Continuous feedback. (Tom Hall n.d.)

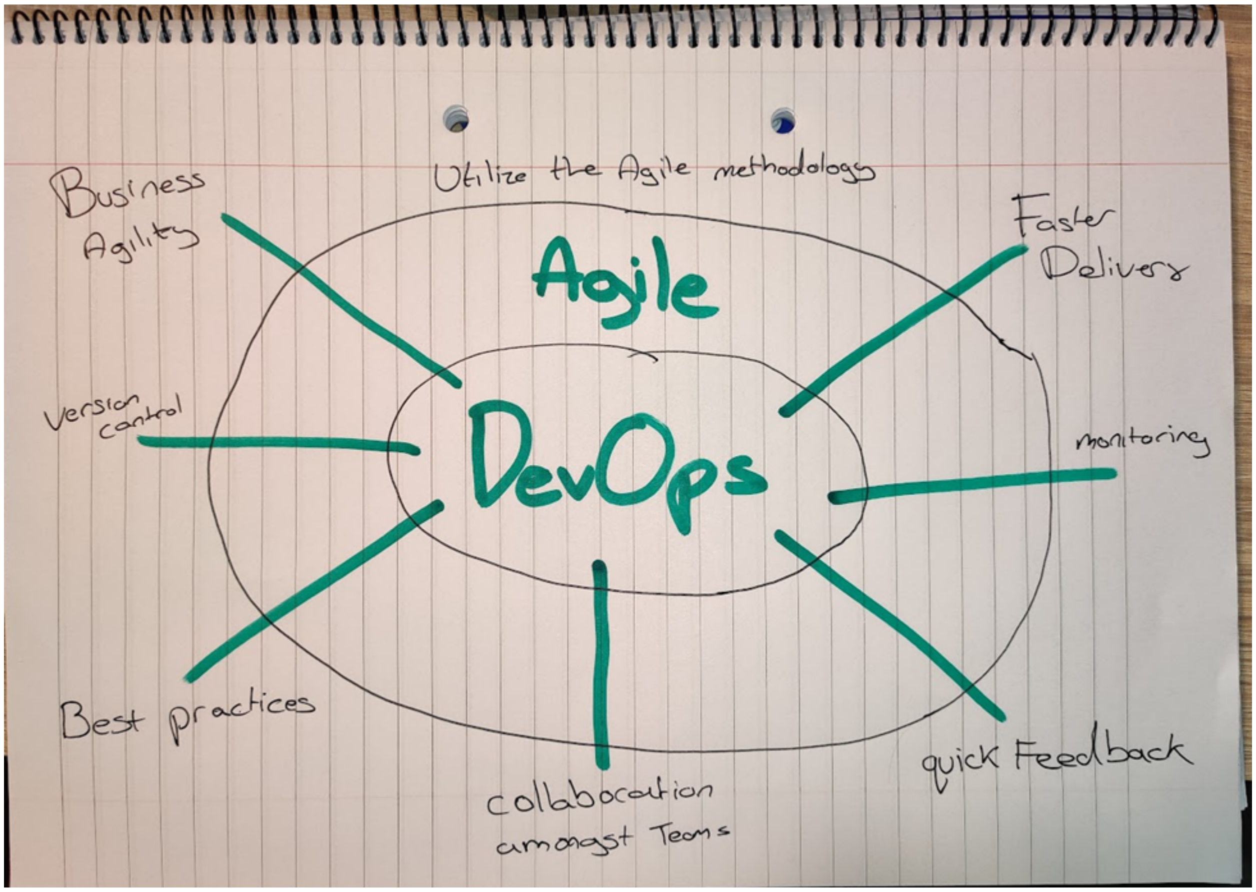
Conclusion

In conclusion the team will work together to bring in a DevOps methodology while working as an agile team. Ren and Jalen will need to work close to execute as both their experience will be needed. Ren has worked on a large application for years and knows the importance of testing and best practices. where Jalen is just new graduate developer and has never worked on a large-scale project. he is unfamiliar with the importance of testing your code, reviewing your peers work before adding it to your code base, the importance of agile fail fast process and much more that only Ren can bring. But what Jalen does bring is his knowledge of the new coding stack and a hunger to learn from Ren and team. Been an Agile team offers good communication amongst the team and allow me to keep up with my teammates progress and form me to offer and receive feedback. It also gives the team a chance to discuss what went well and wrong with each sprint. The improvements made to the codebase by refactoring offered the team a chance to get familiar with the codebase and make it more structured. the best practices such as testing, peer reviews before merging feature branches into main branch makes sure that the codebase stays readable. Adding a pipeline is good for monitoring our tests and making sure our build is still working. Plus allows us to deploy frequently and rapidly, making us a stronger and more reliable team, here at Shinty Software. (Image Three)

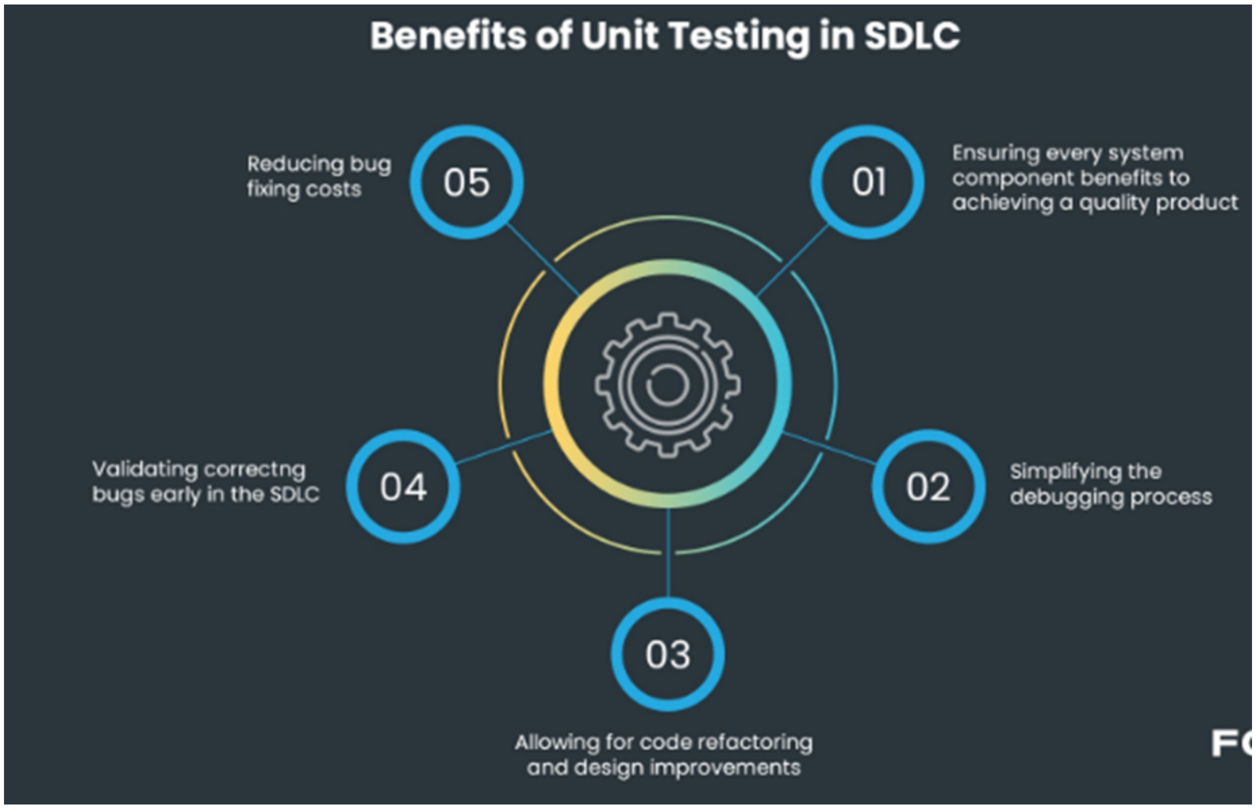
GitHub Repo:

https://github.com/L00170144-Michael/Assignment-2-Broken-Legacy-Code

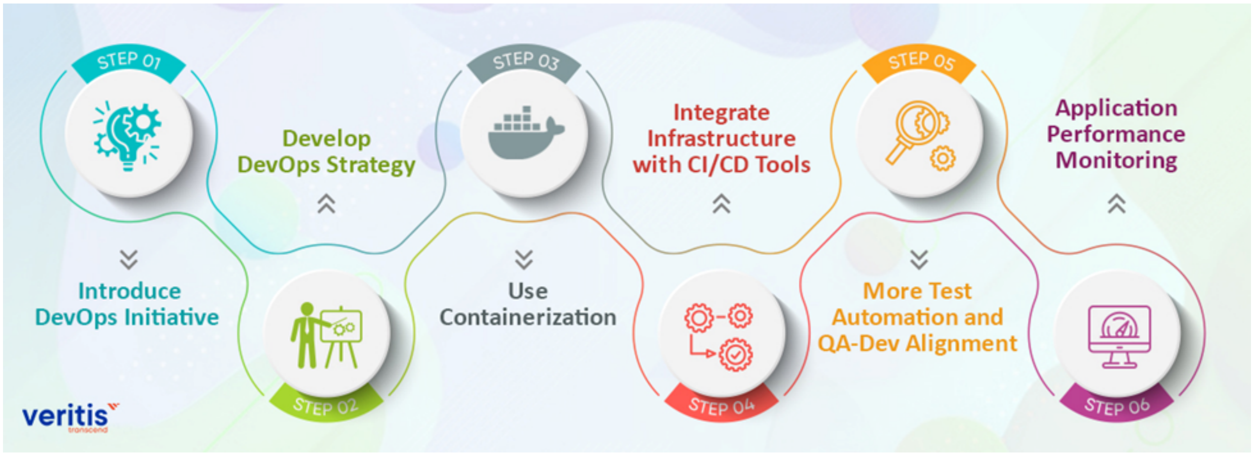
Images:



(Image One)


Benefits of Unit Testing in SDLC
01
Ensuring every system component benefits to achieving a quality product
02
Simplifying the debugging process
03
Allowing for code refactoring and design improvements
04
Validating correctng bugs early in the SDLC
05
Reducing bug fixing costs
FC

(Image Two)


STEP 01
Introduce DevOps Initiative
STEP 02
Develop DevOps Strategy
STEP 03
Use Containerization
STEP 04
Integrate Infrastructure with CI/CD Tools
STEP 05
More Test Automation and QA-Dev Alignment
STEP 06
Application Performance Monitoring
veritis

(Image There)

References:

Fireship (2020) ‘DevOps CI/CD Explained in 100 Seconds’, available: https://www.youtube.com/watch?v=scEDHsr3APg [accessed 11 Nov 2022].

Cigniti, Cigniti Technologies, available:
https://cigniti.com/blog/6-compelling-business-benefits-of-devops/ [accessed 06 Jan 2022].

Mohammad Adil, A. (2022) Why DevOps is Important for Modern Businesses, testsigma, available: https://testsigma.com/blog/why-devops-is-important-for-modern-businesses/ [accessed 06 Jan 2022].

Kseniia Goldovska, KG. (2022) Legacy Software Modernization Services: Key Steps & Strategies, jellyfish.tech, available:
https://jellyfish.tech/blog/legacy-software-modernization-services/ [accessed 06 Jan 2022].

Tom Hall, TH. (n.d.) DevOps VS. Agile: Atlassian, available:
https://www.atlassian.com/devops/what-is-devops/agile-vs-devops

Ben Lutkevich, BL. (2020) 6 lessons learned from 'The Phoenix Project', available:
https://www.techtarget.com/whatis/reference/6-lessons-learned-from-The-Phoenix-Project [accessed 06 Jan 2022].

Oksana Mikhalchuk, OM. (2020) The importance of unit testing, or how bugs found in time will save you money, available:
https://fortegrp.com/the-importance-of-unit-testing/#:~:text=Unit%20testing%20ensures%20that%20all,write%20better%20code%2C%20more%20efficiently. [accessed 06 Jan 2022].