JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

PIPELINE MAESTRO – ORCHESTRATING BUILD, TEST, AND DEPLOY

[1]Ishant Sontakke, [2]Prof. Najeefa Nasreen
[1]PG Student, [2]Assistant Professor
[1,2]Department of Computer Science
[1,2]G.H.Raisoni University, Amravati, India

Abstract: Pipeline Maestro is a complete software solution that optimizes and automates the software development lifecycle, with an emphasis on the orchestration of build, test, and deployment tasks. This project provides a single platform for pipeline definition, management, and execution with the goal of streamlining development workflows, increasing efficiency, and improving code quality. Automated build and test capabilities, smooth interface with version control systems, deployment automation in various settings, functionality for monitoring and reporting, and customization choices to meet a range of development demands are some of the key features.

Index Terms - Continuous Integration, Continuous Delivery, Continuous Deployment, Continuous Software Engineering, Systematic Literature Review, Empirical Software Engineering.

I. INTRODUCTION

With increasing competition in software market, organizations pay significant attention and allocate resources to develop and deliver high-quality software at much accelerated pace [1]. Continuous Integration (CI), Continuous Delivery (CDE), and Continuous Deployment (CD), called continuous practices for this study, are some of the practices aimed at helping organisations to accelerate their development and delivery of software features without compromising quality [2]. Whilst CI advocates integrating work-in-progress multiple times per day, CDE and CD are about ability to release values quickly and reliably to customers by bringing automation support as much as possible [3 - 4].

(A) Continuous practices are expected to provide several benefits .
(B) Getting more and quick feedback from the software development process and customers.
Having frequent and reliable releases, which lead to improved customer satisfaction and product quality.
Through CD, the connection between development and operations teams is strengthened and manual tasks can be eliminated
[5 - 6]. A growing number of industrial cases indicate that the continuous practices are making inroad in software development industrial practices across various domains and sizes of organizations [5 -7 -8]. At the same time, adopting continuous practices is not a trivial task since organizational processes, practices, and tool may not be ready to support the highly complex and challenging nature of these practices. Due to the growing importance of continuous practices, an increasing amount of literature describing approaches, tools, practices, and challenges has been published through diverse venues. Evidence for this trend is the existence of five secondary studies on CI, rapid release, CDE and CD [9 - 13]. These practices are highly correlated and intertwined, in which distinguishing these practices are sometimes hard and their meanings highly depends on how a given organization interprets and employs them [14].

II. RELATED WORK

Software development workflows are streamlined and optimized using several automation tools and platforms in related work in the field of coordinating build, test, and deployment processes. By automating the development, testing, and deployment of software applications, these technologies assist companies in improving productivity, decreasing mistakes, and shortening time to market.
Typical relevant work in this field includes the following:
Tools for continuous deployment and integration (CI/CD): The process of merging code changes, executing tests, and delivering apps to production environments are all automated by these technologies.
Tools for configuration management: Programs like as Ansible, Puppet, and Chef facilitate the automation and administration of server and infrastructure setup.
Tools for Infrastructure as Code (IaC): Code is used to automate infrastructure provisioning and maintenance on platforms like

Terraform and Cloud Formation.
Tools for orchestration: The deployment and scaling of containerized applications may be managed with the aid of platforms such as Dockers Swarm, Apache Mesos, and Cabernets.
Tools for testing automation: Programs such as Selenium, JUnit, and TestNG automate the process of executing tests to guarantee the dependability and quality of software programs.
In general, the associated research in this area is on using automation and orchestration technologies to build dependable and effective software delivery pipelines that let businesses produce high-caliber software more quickly.

III. PROPOSED WORK

A proposed CI/CD pipeline for web applications would include several components that work together to automate the build, testing, and deployment processes. A high-level overview of a proposed system for a CI/CD pipeline for web applications is provided below:
Version Control System (VCS): To manage the codebase and track changes, a version control system such as Git would be used. Git would be used by developers to commit changes to the codebase, and Git would be used by the pipeline to retrieve the most recent version of the code.
Containerization: The application and its dependencies would be packaged into containers using Dockers. The build process would produce a Dockers image, which would be uploaded to a container registry like Dockers Hub.
Continuous Deployment (CD) Tools: A CD tool, such as AWS Code Deploy, would be used to automate the application's deployment across multiple environments. The CD tool would be set up to retrieve the Dockers image from the container registry and deploy it to a staging environment for additional testing. The CD tool would automatically promote the image to the production environment once the application passed all tests in the staging environment.
Testing Frameworks: Testing frameworks like Selenium would be used to automate application testing. A suite of automated tests would be run during the build process and deployment to the staging environment as part of the pipeline. This ensures that any problems are identified and resolved before the application is deployed to production. Monitoring and logging tools, such as New Relic, would be used to monitor the application's performance and provide feedback to the development team. The monitoring tools would be set up to notify the team if any problems arose in production, allowing them to quickly identify and resolve problems.
The proposed CI/CD pipeline for web applications would streamline and automate the process of developing, testing, and deploying applications. It would reduce the likelihood of errors and accelerate development and deployment, resulting in a more efficient and effective development process.
There are numerous crucial processes involved in designing and implementing a CI/CD (Continuous Integration/Continuous Deployment) pipeline for a web application. An overview of the system analysis and method for establishing such a pipeline is given below:

- **Define Requirements**: To begin, determine the precise specifications of your web application and the results you hope to achieve with your CI/CD pipeline. Consider elements like the target platforms, the development and deployment environments, the programming languages, frameworks, and technologies utilized, as well as any quality or security standards.
- **Version Control**: Manage your source code and make sure that all changes are tracked by implementing a version control system (such as Git). Use branching and merging techniques that are compatible with your development workflow (such as GitFlow) and lay out precise rules for code review and teamwork.
- **Automated Build**: Establish an automated build procedure that will translate your application code into deployable artefacts by assembling and packaging them. Create a build script or configuration file that lists the necessary dependencies, build steps, and other requirements (using, for instance, Maven or Cradle tools).

Creating a thorough testing approach can help you ensure the dependability and quality of your web application. End-to-end tests, integration tests, and unit tests are frequently included in this. Use testing frameworks and tools appropriate for the frameworks and programming languages you select (JUnit, Selenium, etc.). To run these tests automatically after each build, incorporate them into your CI/CD pipeline.

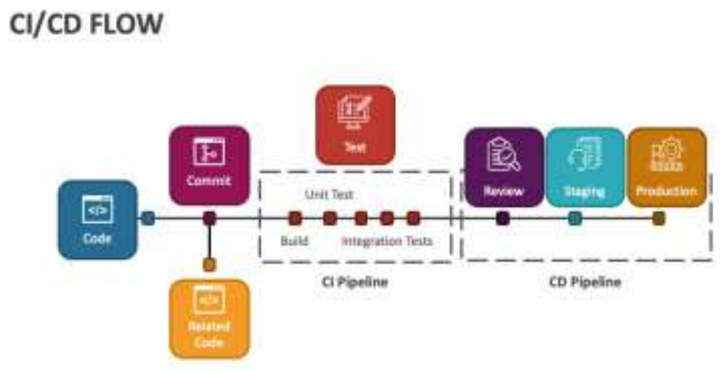


Fig. 1: Flow Chart.

IV. DETAILED SYSTEM ANALYSIS

Detailed system and analysis about deployed project in the Tomcat server using CI/CD pipeline Setup:-
Doctor and patient appointment portals have revolutionized the way healthcare services are delivered. These portals provide a convenient and efficient way for patients to schedule appointments with their healthcare providers, access medical records, communicate with their doctors, and receive important updates and reminders.
Doctor Patient Portal is an Advance Java Project. Technology used in this project: Advance JAVA concepts like JSP, JSTL, Servlet, HTML, CSS, Bootstrap 5 and MySQL.

(A). System Flow Diagram:-

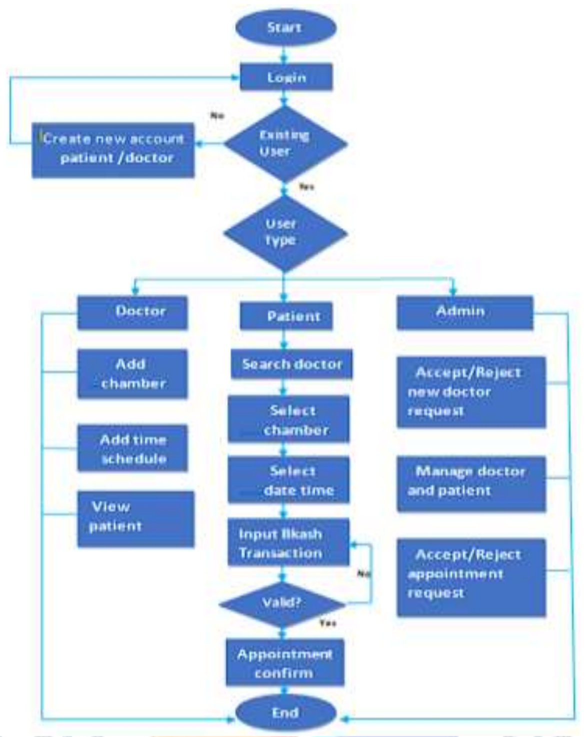


Fig 2: System Flow Diagram of Doctor Patient Appointment Portal

(B). Data Structures and Tables: -

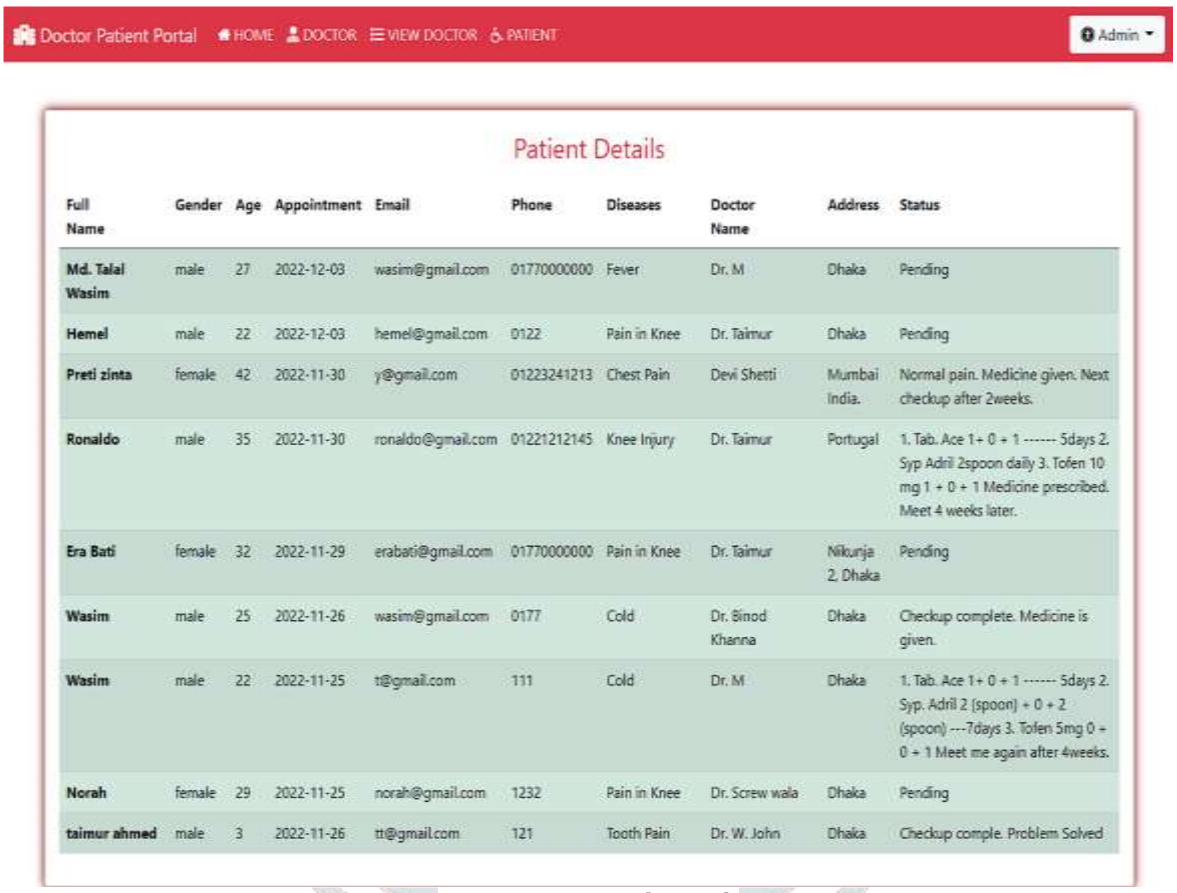
Table 1:- Patient Details

Full Name	Gender	Age	Appointment	Email	Phone	Diseases	Doctor Name	Address	Status
Md. Talal Wasim	Male	27	2022-012-03	wasim@gmail.com	01770000000	Fever	Dr. M	Dhaka	Pending
Hemel	Male	22	2022-12-03	hemel@gmail.com	0122	Pain in Knee	Dr.Taimur	Dhaka	Pending
Preti Zinta	Female	42	2022-11-30	y@gmail.com	01223241213	Chest pain	Devi Shetty	Mumbai, India	Normal pain, Medicine given.

Table 2:- Appointment List

Full Name	Gender	Age	Appointment Date	Phone	Diseases	Doctor Name	status
Wasim	male	22	2022-11-25	111	Cold	Dr. M	1. Tab. Ace 1+0+1-------5 Days.
Md. Talal Wasim	male	27	2022-12-03	01770000000	Fever	Dr. M	Pending

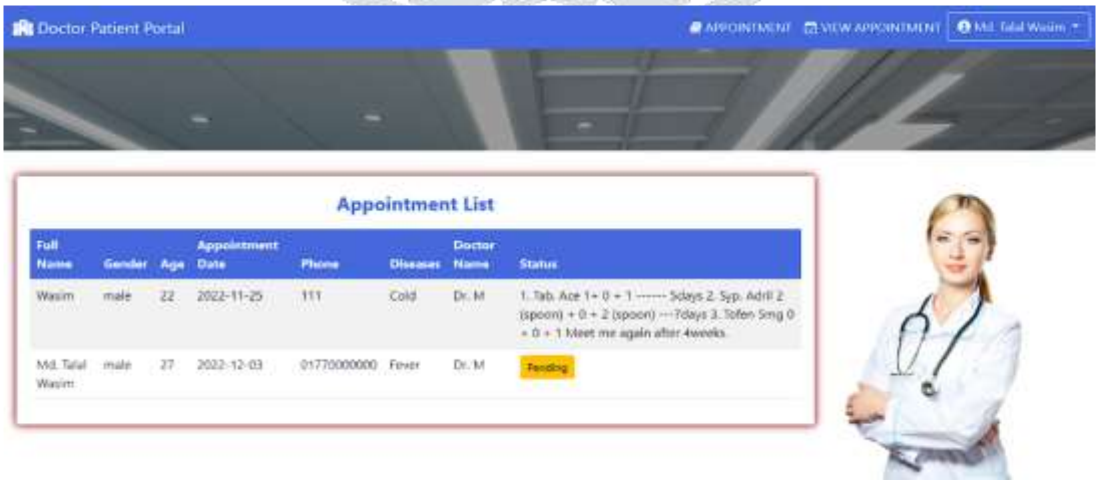
Database Results:-



Patient Details

Full Name	Gender	Age	Appointment	Email	Phone	Diseases	Doctor Name	Address	Status
Md. Talal Wasim	male	27	2022-12-03	wasim@gmail.com	01770000000	Fever	Dr. M	Dhaka	Pending
Hemel	male	22	2022-12-03	hemel@gmail.com	0122	Pain in Knee	Dr. Taimur	Dhaka	Pending
Preti zinta	female	42	2022-11-30	y@gmail.com	01223241213	Chest Pain	Devi Shetti	Mumbai India.	Normal pain. Medicine given. Next checkup after 2weeks.
Ronaldo	male	35	2022-11-30	ronaldo@gmail.com	01221212145	Knee Injury	Dr. Taimur	Portugal	1. Tab. Ace 1+ 0 + 1 ------ 5days 2. Syp Adril 2spoon daily 3. Tofen 10 mg 1 + 0 + 1 Medicine prescribed. Meet 4 weeks later.
Era Bati	female	32	2022-11-29	erabati@gmail.com	01770000000	Pain in Knee	Dr. Taimur	Nikunja 2, Dhaka	Pending
Wasim	male	25	2022-11-26	wasim@gmail.com	0177	Cold	Dr. Binod Khanna	Dhaka	Checkup complete. Medicine is given.
Wasim	male	22	2022-11-25	t@gmail.com	111	Cold	Dr. M	Dhaka	1. Tab. Ace 1+ 0 + 1 ------ 5days 2. Syp. Adril 2 (spoon) + 0 + 2 (spoon) ---7days 3. Tofen 5mg 0 + 0 + 1 Meet me again after 4weeks.
Norah	female	29	2022-11-25	norah@gmail.com	1232	Pain in Knee	Dr. Screw wala	Dhaka	Pending
taimur ahmed	male	3	2022-11-26	tt@gmail.com	121	Tooth Pain	Dr. W. John	Dhaka	Checkup comple. Problem Solved

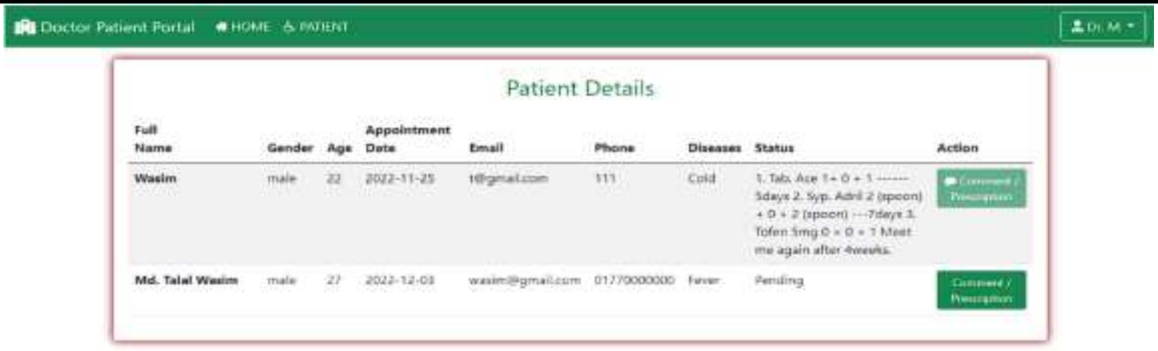
Fig 3: Patient Details Result



Appointment List

Full Name	Gender	Age	Appointment Date	Phone	Diseases	Doctor Name	Status
Wasim	male	22	2022-11-25	111	Cold	Dr. M	1. Tab. Ace 1+ 0 + 1 ------ 5days 2. Syp. Adril 2 (spoon) + 0 + 2 (spoon) ---7days 3. Tofen 5mg 0 + 0 + 1 Meet me again after 4weeks.
Md. Talal Wasim	male	27	2022-12-03	01770000000	Fever	Dr. M	Pending

Fig 4: Appointment List



Patient Details

Full Name	Gender	Age	Appointment Date	Email	Phone	Diseases	Status	Action
Wasim	male	22	2022-11-25	t@gmail.com	111	Cold	1. Tab. Ace 1 + 0 + 1 ------ 5days 2. Syp. Adril 2 (spoon) + 0 + 2 (spoon) ---7days 3. Tofen 1mg 0 + 0 + 1 Meet me again after 4weeks.	Comment / Prescription
Md. Talal Wasim	male	27	2022-12-03	wasim@gmail.com	01770000000	Fever	Pending	Comment / Prescription

Fig 5: Patient Details

V. PROPOSED RESEARCH MODEL

The following elements may be included in a suggested research model for Pipeline Maestro - Orchestrating Build, Test, and Deploy:

Problem Statement: Clearly state the difficulties and problems that businesses have while trying to manage the software application development, testing, and deployment processes.

Describe the precise aims and objectives of the study, such as enhancing productivity, decreasing mistakes, and shortening deployment schedules.

Review of the Literature: Examine the body of knowledge about software development deployment techniques, testing methodologies, CI/CD procedures, and automation technologies. Theoretical Framework: Create a theoretical framework that delineates the fundamental ideas and precepts that underpin the coordination of the activities involved in build, test, and deployment.

Study Methodology: Describe the study methodology, taking into account the procedures for data analysis, sampling strategies, and data gathering.

Data Collection: Gather information on the experiences, difficulties, and results of companies who are coordinating build, test, and deployment processes by utilizing Pipeline Maestro or comparable platforms.

Data Analysis: Examine the gathered information to find trends, patterns, and insights into how Pipeline Maestro works and how it affects software delivery pipelines.

Results and Suggestions: Outline the study's conclusions and include suggestions on how to best utilize Pipeline Maestro to coordinate the build, test, and deployment processes.

Provide a summary of the study's main conclusions and talk about how they could affect software development procedures and future lines of inquiry.

Highlight the study model's contributions to the advancement of knowledge and comprehension about the use of automation tools in software delivery pipelines.

VI. PERFORMANCE EVALUATION

The following key performance indicators (KPIs) and metrics can be used to assess how well Pipeline Maestro performs in coordinating build, test, and deployment processes:

- **Build Time**: Calculate how long it takes to compile a software program from its source code into executable binaries. To evaluate the effect on efficiency, compare the build times prior to and following the implementation of Pipeline Maestro.
- **Test Execution Time**: Calculate how long it takes to run the application's automated tests. To ascertain the effect on testing efficiency, compare the test execution time with and without Pipeline Maestro.

Determine the frequency with which updates or new features are rolled out to production environments. Evaluate if Pipeline Maestro's streamlined deployment procedure has resulted in more frequent deployments. Deployment Success Rate: Monitor the Pipeline Maestro deployments' success rate. Track the proportion of successful vs unsuccessful deployments to determine the orchestration tool's dependability and efficacy.

VII. RESULT ANALYSIS

Faster Time-to-Market: By automating the build, testing, and deployment processes, CI/CD makes it possible to bring features and bug fixes to production more quickly. Developers may concentrate on writing code while the pipeline handles monotonous chores, which reduces manual lab hour and quickens the release cycle.
Continuous Integration (CI) is the practice of developers routinely merging their changes into a shared repository, which starts automated builds and tests. As a result, conflicts are promptly identified, and code updates are regularly implemented. A more stable codebase results from continuous integration's assistance in quickly identifying and resolving integration problems.
Early Bug Detection: The CI/CD pipeline includes automated testing, such as unit tests, integration tests, and end-to-end tests. Testing every update to the code helps find problems and issues early, enhancing the quality of the product.
Collaboration among developers, testers, and operations teams is improved thanks to CI/CD. Developers can work on several additions or fixes at once using version control, and teams can evaluate code changes and offer input. This teamwork improves communication and makes sure that everyone is striving to provide a trustworthy online application.
Deployment Reliability: The pipeline's CD component offers automation that makes deployments consistent and repeatable. The application is deployed in a known state and deployments are carried out in a controlled and standardized manner, minimizing the possibility of human mistakes.

Fig 6: Model Training and Accuracy

VIII. CONCLUSION

Continuous Integration/Continuous Deployment (CI/CD) is a development methodology that allows developers to automate the process of creating, testing, and deploying software applications. The CI/CD pipeline is a collection of automated processes that assist developers in ensuring that code changes are constantly tested and integrated into the application without errors or issues. In the world of software development, CI/CD is becoming increasingly popular, particularly for web applications. This is because web applications are typically more complex than other types of software applications, necessitating more testing and integration to ensure proper operation. This article will go over the CI/CD pipeline for web applications and how it can help developers.
Overall, Pipeline Maestro has proven to be a valuable asset in optimizing software delivery pipelines, enhancing productivity, and driving continuous improvement in software development practices.

REFERENCES

[1] Phillips, M. Sens, A. de Jonge and M. van Holsteijn, The IT Manager's Guide to Continuous Delivery, XebiaLabs, Hilversum, The Netherlands, 2015.
[2] J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases Through Build Test and Deployment Automation, Reading, MA, USA: Addison-Wesley, 2010.
[3] M. Fowler, Continuous Integration, Oct. 2015, [online] Available: http://martinfowler.com/articles/continuousIntegration.html.
[4] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda", *J. Syst. Softw.*, vol. 123, pp. 176-189, Jan. 2017.
[5] M. LeppAďnen et al., "The highways and country roads to continuous deployment", *IEEE Softw.*, vol. 32, pp. 64-72, Mar. 2015.
[6] L. Chen, "Continuous delivery: Huge benefits but challenges too", *IEEE Softw.*, vol. 32, no. 2, pp. 50-54, Mar. 2015.
[7] A. A. U. Rahman, E. Helms, L. Williams and C. Parnin, "Synthesizing continuous deployment practices used in software development", *Proc. Agile Conf. (AGILE)*, pp. 1-10, Aug. 2015.
[8] H. H. Olsson, H. Alahyari and J. Bosch, "Climbing the 'stairway to heaven': A mulitiple-case study exploring barriers in the transition from agile deployment of software", *Proc. 38th Euromicro Conf. Softw. Eng. Adv. Appl.*, pp. 392-399, Sep. 2012.
[9] P. Rodríguez et al., "Continuous deployment of software intensive products and services: A systematic mapping study", *J. Syst. Softw.*, vol. 123, pp. 263-291, Jan. 2017.
[10] E. Laukkanen, J. Itkonen and C. Lassenius, "Problems causes and solutions when adopting continuous delivery ", *Inf. Softw. Technol.*, vol. 82, pp. 55-79, Feb. 2017.

[11] D. Ståhl and J. Bosch, "Modeling continuous integration practice differences in industry software development", *J. Syst. Softw.*, vol. 87, pp. 48-59, Jan. 2014.

[12] M. V. Mäntylä, B. Adams, F. Khomh, E. Engström and K. Petersen, "On rapid releases and software testing ", *Empirical Softw. Eng.*, vol. 20, no. 5, pp. 1384-1425, 2015.

[13] A. Eck, F. Uebernickel and W. Brenner, "Fit for continuous integration: How organizations assimilate an agile practice", *Proc. 20th Amer. Conf. Inf. Syst.*, 2014.

[14] A. Thiele, Continuous Delivery: An Easy Must-Have for Agile Development, Jul. 2016,https://blog.inf.ed.ac.uk/sapm/2014/02/04/continuous-delivery-an-easy-must-have-foragile-development/.

[15] Usha Kosarkar, Gopal Sakarkar, Shilpa Gedam (2022), "Revealing and Classification of Deepfakes Videos Images using a Customize Convolution Neural Network Model", *International Conference on Machine Learning and Data Engineering (ICMLDE)*, 7th & 8th September 2022, 2636-2652, Volume 218, PP. 2636-2652, https://doi.org/10.1016/j.procs.2023.01.237

[16] Usha Kosarkar, Gopal Sakarkar (2023), "Unmasking Deep Fakes: Advancements, Challenges, and Ethical Considerations", *4th International Conference on Electrical and Electronics Engineering (ICEEE)*,19th & 20th August 2023, 978-981-99-8661-3, Volume 1115, PP. 249-262, https://doi.org/10.1007/978-981-99-8661-3_19

[17] Usha Kosarkar, Gopal Sakarkar, Shilpa Gedam (2021), "Deepfakes, a threat to society", *International Journal of Scientific Research in Science and Technology (IJSRST)*, 13th October 2021, 2395-602X, Volume 9, Issue 6, PP. 1132-1140, https://ijsrst.com/IJSRST219682

[18] Usha Kosarkar, Gopal Sakarkar (2024), "Design an efficient VARMA LSTM GRU model for identification of deep-fake images via dynamic window-based spatio-temporal analysis", *International Journal of Multimedia Tools and Applications, 8th May 2024*, *https://doi.org/10.1007/s11042-024-19220-w*