- Peer reviews are complete with no outstanding issues.
- Deployed to the production environment or ready for deployment.

2. Move Task to Done

- Transition the task to the **Done** column in the tool.
- Update any documentation or notify stakeholders, if needed.

Step-6: Continuously Improve Workflow

1. Conduct Retrospectives

- Regularly review the workflow to identify bottlenecks or inefficiencies.

2. Optimize Automation

- Add more automation to reduce manual effort, such as deployment pipelines.

3. Enhance Communication

- Use tools like Slack or Microsoft Teams for instant updates on task progress.

SOURCE CODE

PHASE 1: To Do

- Objective : Identify and prioritize tasks or features to be developed
- Key Actions :
  - Define tasks clearly in a backlog
  - Prioritize tasks based on impact, urgency & dependencies
  - Assign owners or teams to each task
- Tools : Jira, Trello, GitHub Issues or Asana

PHASE 2: IN PROGRESS

- Objective : Actively work on tasks selected from the "To Do" phase.
- Key Actions :

- Begin coding or configuring based on task requirements
- Update the task status to reflect ongoing work
- Ensure team members collaborate effectively (eg: - stand ups, pair programming)

- Best Practices:
  - Use branches in version control systems for individual tasks (e.g. Git feature branches)
  - Write unit tests alongside development

PHASE 3: CODE REVIEW

- Objective: Validate the quality, functionality and security of the code
- Key Actions:
  - Submit pull requests for peer review
  - Review code for adherance to standards, logic & potential issues
  - Approve or request changes
- Tools:
  Github Pull Requests, Gitlab Merge Requests, Bitbucket
- Automation:
  Integrate CI/CD pipelines to run tests automatically during reviews.

PHASE 4: DONE

- Objective: Mark tasks as completed and deploy changes if necessary

- Key Actions:
  - Merge the approved code into main branch
  - Deploy to staging or production environments
  - Monitor deployment and validate functionality
- Post Completion:
  - Add documentation for the changes
  - Gather feedback from stakeholders or users

WORKFLOW VISUALIZATION

A kanban board or similar visual representation can help track the status of tasks across these phases. For example:

i) To do : Contains all pending tasks
ii) In Progress : Tasks currently being worked on
iii) Code Review : Tasks awaiting review or approval
iv) Done : Completed & deployed tasks

Tools:

Trello, Jira, Azure

SOURCE CODE

STEP-1 : Install Jdk - 17 and set the java path in system environment

STEP-2 : Download eclipse zip file and extract the contents of the eclipse file

STEP-3 : Create a maven project from eclipse as :
Click on File in left corner → Click on new → Click on Maven Project and follow the below steps : -

a. Click Next in the New Maven Project dialog box that appears.
b. Search for org.apache.maven.archetypes and select webapp file.
c. In groupId you can type anything like your name and in ArtifactId type anything like your roll number. Select Was as package
d. Click Finish
e. Type Y and press enter. You should see a Build Success message.

STEP-4 : Now open your pom.xml file and add your dependencies

STEP-5 : Update your project once (Right click on Project → Click on Maven → Click on Update project)

STEP-6 : Download Apache Tomcat v9 from official website

STEP-7 : After downloading the Apache Tomcat, extract the .zip file and paste your apache-tomcat 9.0.98 folder ℰ.

STEP-8 : Now, click on your project option in Menu → Click on Properties → Click on Targeted Runtime

STEP-9 : Click on new.

STEP-10 : Select Apache Tomcat v9.0. Click on Next.

STEP-11 : Click on Browse and select your extracted file and then click on finish. ~~as given image~~.

STEP-12 : Now click on Help menu → Click on Install new software

STEP-13 : Click on Add and it will show a popup dialog like g of Add Repository.

In the place of Name type : TestNG

In the place of Location type : https://testng.org/testng-eclipse-update-site/

STEP-14 : Click on Add → It will load a testNG dependencies → Select TestNG like given image and then click Next. It will take 10 minutes to update TestNG in our project.

STEP-15 : After downloading all the dependencies it will show some file select all and click on next.

STEP-16 : Accept Terms and Conditions and Click on Finish.

STEP-17 : After finishing, it will show Restart option (Restart the project) otherwise just update once of your project.

STEP-18 : Now login your Github Account.

STEP-19 : Create a New Repository and copy your repository and paste in Notepad.

Step-20 : After that, click on your Profile in right corner → Click on Setting.

STEP-21 : It will show a new page, scroll down and select the developer setting → Click on personal access token → Select Token (classic) → Click on Generate new token and select Generate new token (Classic) → Write your token name & select repo option and scroll down and click on Generate Token.

STEP-22 : After generating the token, copy the token id and paste it in a Notepad

STEP-23 : Now come on your project and right click on your project → Click on Team → Click on Share Project.

STEP-24: It will open a dialog Box for Github setup, select the option Use or create repository in parent folder of project → select your Project and click on Create Repository and click on Finish.

STEP-25: After that again, right click on your project and select the Team → Click on Commit → and stage your all file → and write a comment (i.e., First Commit) and click on Commit and push → After that it shows an error dialog → Click OK → Now again click on Push Head button.

STEP-26: After than, again click on Push Head. It will show a dialog box of Push Branch master. Paste your Repository URL in URL section and type your Github User Id and Password in User, password section → Click on Preview → Again click on Preview.

STEP-27: Then, it will again show a user Id & password option → Just type your Github id in user section and paste your token id in password token → Click on push → One more time it will ask your user id and password, just repeat your last step → Now check your repository on github, if your file is uploaded or not.

STEP-28: Create a simple java code in SRC file, so first open your project from file manager → Open src → Create folder in src → First name: java, second name: test → Open test folder and create two more folders in test folder → Come to your Eclipse IDE & update your project once → After that create a java class file with a statement "Hello World" in your src/test/java folder.

STEP-29: Now push again your all unstage files in your Github Repository with different version or Comment (its just for version control)

STEP-30: Now check your repository again to see whether your recent file is uploaded or not.

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.27.0</version>
  </dependency>
  <dependency>
    <groupId>org.testing</groupId>
    <artifactId>testing</artifactId>
    <version>7.10.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.5.2</version>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.13.0</version>
```

OUTPUT

```
        <configuration>
            <source>17</source>
            <target>21</target>
        </configuration>
    </plugin>
  </plugins>
<finalName>6229</finalName>
</build>
</project>
```