|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Domain** | **Design ThinkingC:\Users\stephanie.deguire\Downloads\PikPng.com_empathy-png_4008404.png** | **Agile** Scrum Icons - Download Free Vector Icons | Noun Project | | | **DevOps** Devops Process Icons - Download Free Vector Icons | Noun Project | | | | |
| **Digital Dojo  Principles** | **Continuous User Feedback** | **Continuous Planning** | **Continuous Collaboration** | **Continuous Improvement** | **Continuous Integration** | **Continuous Delivery** | **Continuous Operations** | **Continuous Quality** | **Continuous Security** |
| **Digital Dojo  Practices** | * Personas * Journey Mapping * Prototyping * User Research * Usability Testing * Accessibility * Minimum Viable Product (MVP) | * Backlog management * Kanban boards * Work/Sprint planning * Estimation and velocity * Prioritization * Limiting Work in Progress (WIP) | * Self-organized team * Daily stand-ups * Swarming * Show and Tell / Sprint Reviews * Participation in guilds, communities | * Retrospectives * Team Health Check * DevOps Kaizen * Values Stream Mapping | * Version control strategies * Trunk based development * Test automation * Push/Pull requests * Git | * Automated build and deployment  pipelines * Feature flags * Infrastructure as code | * Site Reliability Engineering * Incident management * Monitoring metrics * Cloud Infrastructure Management * Dynamic scaling * Administration task automation | * Definition of Done * Peer reviews * Test Automation * Integration into build pipeline * TDD * Shift left / right testing * Incident post mortems * Analyzing monitoring metrics | * Secure Architecture * Automated security tests * Identity Management * Vulnerability/ dependency/ patch scanning and monitoring. * White hat hackers |
|  | Our requirements are business rules, and we don't really know how users feel about our product/service. | Our team does not have a clear backlog of work, or know what a backlog is.  We don't use a work tracking system. | We don't always know who is working on what, and if they are struggling. | Our team doesn't have a mechanism to discuss pain points and suggest ways to improve how we work. | Our code base is in source control. | Only designated team members perform the release. It is risky, painful, has many manual steps and takes a long time. | We have very limited access to production logs and metrics.  Creating new environments is a manual effort. | We are not proud or confident in the quality of our code.  We rarely conduct code reviews.  Most of our tests are manual. | We put very little emphasis on security testing. |
|  | Our team uses end-user feedback to establish the requirements of our service(s). | We have regular meetings to plan/review work.  Tasks are tracked in a tool and not a document.  We can visualize our work in progress (Kanban). | We have daily stand-ups where we discuss impediments.  Our work is visible to other teams and stakeholders.  We participate in communities of practice relevant to our domain. | Our team meets on regular basis (retrospectives) to discuss how we work. | + Code commits result in an automated build of the software.  We have a version control strategy that works well for us. | Builds / deployments triggered manually or scheduled.  We release new features to users less 2-5 times a year.  Releases to production are complex and require manual intervention. | Procedures are in place to inform us of incidents.  We have access to log files in production. | We have a way to measure test coverage.  We engage with IT Security and Accessibility experts to review our code/product prior to each release.  Some tests are automated | We engage with IT Security to review our code/product prior to each release, code and environments reside in a secure store. |
|  | + We have defined personas and document requirements in the form of user stories.  We actively collect customer feedback on product and features quality. | + We have a backlog that is prioritized/maintained.  We plans our work in short cycles.  We understand when tasks are ready to be worked on, and when they can be marked as done.  We are good at limiting our work in process (WIP). | We have documented our ways of working as a team, and we have agreed on our use of tools for collaboration purposes.  Our team works out in the open. | + Action items coming from our retrospectives make their way into our backlog.  We run regular team health checks | + Code commits result in a series of automated tests being run.  All commits are tied to tasks. | Builds / deployments are triggered automatically, automated testing part of the pipeline.  Our features are sliced in a way that lend themselves to frequent production releases. | + We can quickly spin up new instances.  We have dynamic dashboards to visualize server status. | Testers work alongside developers throughout the software development and delivery process.  Manual test activities are performed continuously throughout the delivery process.  We maintain test data. | Security requirements are included in the automated testing process.  We run some tests in production to ensure there are no security risks. |
|  | + We have processes or tools in place to incorporate user feedback, including anonymous user feedback, into the design of the service.  We're proud of what we deliver because we know our users love it. | + We have a roadmap for our service/product.  We are able to forecast how long it would take us to do large features.  We work at a sustainable pace. | + We self-organize and self-manage through self-assigning tasks.  We feel safe speaking up, and feel there are opportunities for us to do so.  We share our knowledge in communities of practices | + Our retrospectives are useful, our team participates well, and they lead to positive change.  Our team relentlessly inspects, adapts and automates our processes to deliver value to users faster. | + Our software is in a deployable state throughout its lifecycle.  Dependencies are stable and rarely break teams’ code. | + Anyone can deploy our product to production, or to end users, at any time, on demand.  Releasing is simple, safe, painless and mostly automated. | + We can monitor and dynamically increase or decrease cloud/server resources available for the service or product that we support based on demand.  There are automated processes in place to quickly respond to incidents. | Our pipeline includes code reviews and multiple layers of testing. We have dashboards that show  code coverage and pass/fail information.  When the automated tests pass, we are confident the software is releasable.  We continuously review our test suite to better find defects. | +Security reviews do not slow down the development cycle.  Code is scanned for security issues.  There are automated processes in place to quickly respond to security or privacy breaches. |
| **TBS Digital Standards** | **Design with users**  **Design ethical services** | **Iterate and improve frequently** | * **Collaborate widely** * **Work in the open by default** * **Empower staff to deliver better services** * **Use open standards and solutions** | **Iterate and improve frequently** | **Iterate and improve frequently** | **Iterate and improve frequently** | * **Iterate and improve frequently** * **Address security and privacy risks** | * **Build in accessibility from the start** * **Use open standards and solutions** * **Be good data stewards** * **Design ethical services** | * **Address security and privacy risks** * **Be good data stewards** * **Design ethical services** |