

PRD DEVELOPER SERVICE DELIVERY REPORT TEMPLATE

**Proposed by,**

**Rithish sam m**



Developers typically deliver the various components outlined above through a combination of tools and practices. Here’s how to approach the delivery of each specific item, along with examples:

**After receiving a product requirements document (PRD), a developer typically delivers the following:**

**Service Delivery Report Template**

| SOFTWARE PRODUCT TITLE | Title |  |  |
| --- | --- | --- | --- |
| AUTHOR | Name |  |  |
| Git url |  |  |  |
| EMAIL |  |  |  |
| Delivery Date | MM/DD/YY |  |  |
| Reporting Date | MM/DD/YY |  |  |
| SOFTWARE OVERVIEW |  |  |  |
| brief summary of the software's purpose, target users, and key features | | | |
| SOFTWARE PRODUCT OBJECTIVES | |  |  |
| Primary Objective | Specify the main goal of the software | | |
| Secondary Objective | List any additional goals or functionalities | | |
| STAKEHOLDERS | |  |  |
| Primary Stakeholders | List individuals or teams involved and their roles | | |
| Secondary Stakeholders | Additional stakeholders impacting or influenced by the software | | |
| SPECIFICATIONS | |  |  |
| NODE VERSION | Specify compatibile node version | | |
| NPM VERSION | Specify compatibile npm version | | |
| SERVICE DEPENDENCIES | Frameworks, Libraries, Technologies, Images, etc. (FOR DOCKER IMAGE SPECIFICATION) | | |
| ENVIRONMENTAL VARIABLES | .env attributes - environmental variables - to define while composing services in docker | | |
| CHANGE LOG | Changelog. changes been made on top of previous version of the service or app. | | |
| README | Documentation. for further developer's reference and reusability. | | |
| SETUP GUIDE | Essential step-by-step procedures  For the service  To work  On a system | | |

| FUNCTIONAL REQUIREMENTS **(NIL if NOTHING)** | |  |  |
| --- | --- | --- | --- |
| FEATURE 1 | | | |
| Description | Detailed explanation of the feature | | |
| User Story / Use Case | Example scenario or interaction | | |
| Acceptance Criteria | Conditions for successful implementation | | |
| FEATURE 2 | | | |
| Description | Detailed explanation of the feature | | |
| User Story / Use Case | Example scenario or interaction | | |
| Acceptance Criteria | Conditions for successful implementation | | |
| NON-FUNCTIONAL REQUIREMENTS **(NIL if NOTHING)** | |  |  |
| PERFORMANCE | | | |
| Requirement | Specify performance expectations | | |
| Metrics | Measurement criteria | | |
| SECURITY | | | |
| Requirement | Security standards and protocols | | |
| Data Encryption | Encryption methodologies | | |
| ASSUMPTIONS AND CONSTRAINTS | |  |  |
| Assumptions | List any assumptions made during requirement gathering | | |
| Constraints | Specify limitations such as budget, time, or technology | | |
| DEPENDENCIES **(NIL if NOTHING)** | |  |  |
| External Dependencies | Third-party integrations, APIs, or services required | | |
| Other |  | | |

| VERSION HISTORY AND CHANGE LOG | |  |  |
| --- | --- | --- | --- |
| VERSION | EDITS COMPLETED BY | DATE | DESCRIPTION OF EDIT |
| 1.00 | Name | MM/DD/YY |  |
| 1.00 | Name | MM/DD/YY |  |
| 1.00 | Name | MM/DD/YY |  |
| 1.00 |  | MM/DD/YY |  |
| 1.00 |  | MM/DD/YY |  |
| 1.00 |  | MM/DD/YY |  |
| 1.00 |  | MM/DD/YY |  |
| 1.00 |  | MM/DD/YY |  |

**TEST REPORTS (NIL if NOTHING)**

| Unit Test Report |  |
| --- | --- |
| Integration Test Report |  |
| Performance Report |  |
| Other Tests |  |
| Bugs Identification, Patches and Improvements |  |
| POINT OF FAILURE, RED AREAS |  |

**1. Technical Specifications:** Detailed documentation outlining how they plan to implement the features described in the PRD, including architecture, technologies, and APIs.

**- Mode:** Document shared via collaboration tools (e.g., Confluence, Google Docs).

**- Example:** A developer might create a detailed spec document outlining the database schema, API endpoints, and user flows.

**2. Wireframes or Prototypes:** If applicable, they might create visual representations of the user interface based on the requirements.

**- Mode:** Design tools (e.g., Figma, Sketch, Adobe XD) shared via links or presentations.

**- Example:** A clickable prototype showcasing user interactions that can be reviewed by stakeholders for feedback.

**3. Code Implementation:** The actual code that fulfills the requirements outlined in the PRD, which may include backend, frontend, or both.

**- Mode:** Source code management (e.g., GitHub, GitLab).

**- Example:** The developer pushes code to a branch in a Git repository and opens a Pull Request (PR) for review.

**4. Unit Tests:** Automated tests that verify individual components or functions work as intended.

**- Mode:** Code repositories with test files included (e.g., Jest for JavaScript, JUnit for Java).

**- Example:** The developer adds test cases in a dedicated test folder alongside the implementation code.

**5. Integration Tests:** Tests ensuring that different parts of the application work together correctly.

**- Mode:** Included in the same code repository, often run through CI/CD pipelines (e.g., Jenkins, Travis CI).

**- Example:** A suite of integration tests that run automatically when the PR is created.

**6. Documentation:** Clear instructions or comments within the code, as well as external documentation for future reference.

**- Mode:** README files in the code repository, inline comments, or separate documentation sites (e.g., ReadTheDocs).

**- Example:** A comprehensive README that explains how to set up the project, including installation steps and usage examples.

**7. Deployment Scripts:** If applicable, scripts or instructions for deploying the code to production or staging environments.

**- Mode:** Versioned scripts in the code repository or configuration files for deployment tools (e.g., Docker, Kubernetes).

**- Example:** A `deploy.sh` script that automates the deployment process when executed.

**8. Bug Fixes and Improvements:** Based on testing or feedback, they might address any issues that arise during the development process.

**- Mode:** Continual updates in the same code repository, often linked to issue trackers (e.g., Jira, GitHub Issues).

**- Example:** The developer creates additional PRs to address bugs discovered during testing.

**9. Review Feedback:** Participation in code reviews and iterations based on feedback from peers or stakeholders.

**- Mode:** Code review tools integrated into the source code management system.

**- Example:** Feedback is provided directly on the PR via comments, and the developer makes necessary adjustments before merging.

**10. Post-Delivery Support:** Assistance in troubleshooting any issues that arise after the initial delivery, ensuring smooth integration and usage.

**- Mode:** Communication tools (e.g., Slack, email) for ongoing support and issue resolution.

**- Example:** The developer remains available for questions and bug reports after the code has been deployed.

**Example Workflow:**

**1. Development Phase:**

- Developer works on features based on the PRD.

- Writes code, tests, and documents it in a Git repository.

**2. Pull Request Creation:**

- Developer opens a PR in GitHub with a summary of changes and links to relevant documentation.

**3. Code Review:**

- Team members review the PR, provide feedback, and suggest changes directly on the PR.

**4. Testing:**

- CI/CD pipeline runs automated tests. Developer addresses any issues.

**5. Merging:**

- Once approved, the PR is merged into the main branch.

**6. Deployment:**

- Deployment scripts are executed, pushing the changes to production.

**7. Post-Deployment Support:**

- Developer monitors for issues and responds to user feedback.

**This workflow helps ensure that the development process is organized, collaborative, and transparent.**