PLANE APP TEST STRATEGY

[1. Brief Description of SUT 2](#_531oso1c9hd6)

[2. Team Size and Structure 5](#_gp8mguv92or2)

[3. Current Status of QA Process (QA Audit) 10](#_kd7hcpw75l86)

[4. Testing Approach 12](#_nz15fj6mm9ym)

[5. Testing Levels 12](#_esaf0a18wuxk)

[6. Compatibility Testing Priorities 12](#_4ugcaot1zfgl)

[5. Testing Types 15](#_okdk5rvtl5z)

[6. Impediment mitigation 17](#_7j9z5uxzhe0o)

[7. Testing phases 23](#_m38pwtsysvx2)

[8. Reporting 26](#_et35j6n97awv)

[9. Hotfix 28](#_xfcxy1np7gdv)

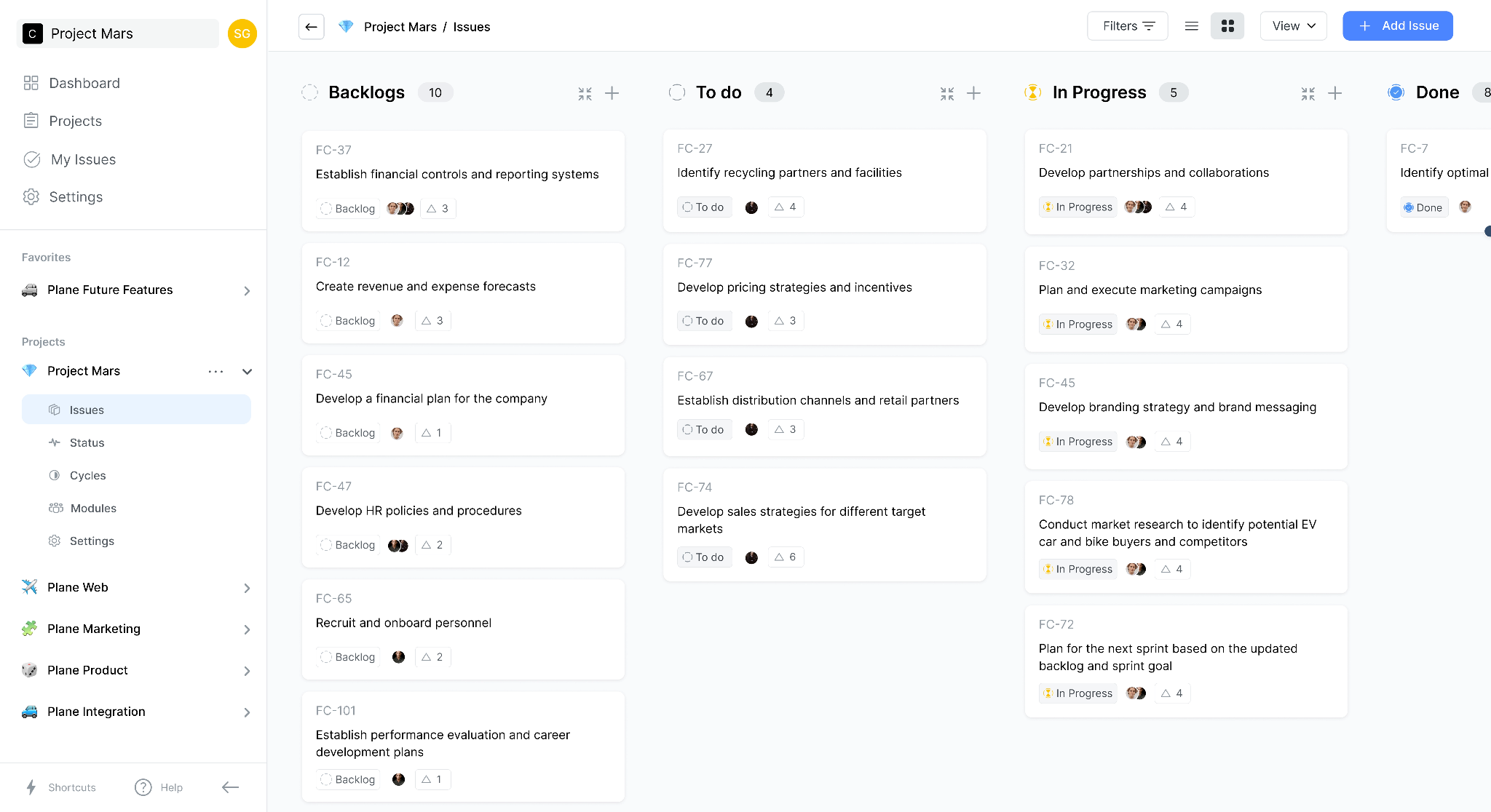
[10. CI/CD testing pipeline 28](#_2zyk9nfgrt9t)

[Links 28](#_1xp3o275gz1b)

### Brief Description of SUT

Plane is an open-source software development tool to manage issues, sprints, and product roadmaps. It is positioned as a Jira alternative.

Product website: plane.so.



Main selling points:

* Simple UX
* Customization using Cycles and Modules
* Integrations, such as GitHub, Slack; Importers to migrate from other products
* Self-hosted and cloud versions

Main features

- Issue Planning and Tracking

- Issue Attachments

- Layouts

- Cycles

- Command Pannel

- GitHub Sync and other integrations

- AI Assistant Notepad

- Team Members

Tech stack

- Frontend: JavaScript / Vue.js / Tailwind CSS

- Backend: Python / Django/ / Model-View-Controller (MVC) pattern

- Web Server: Gunicorn - The API code uses Gunicorn as the web server. Gunicorn is a Python WSGI (Web Server Gateway Interface) HTTP server that allows the application to handle multiple concurrent requests efficiently.

- Database: PostgreSQL

- Authentication: JWT

- Integrations (cloud version only): GitHub, Slack, Jira importer (coming soon)

### Team Size and Structure

1. Core team (see below).

2. GitHub contributors: 40+.

| Role | Name | Description |
| --- | --- | --- |
| Senior Developer (Tech Lead) | Will | Location: San Francisco  Expertise, frameworks: Full-stack developer with over 8 years of experience, specializing in building scalable web applications. Proficient in Java, Spring Boot, React, Redux, and MySQL. Skilled in leading and mentoring development teams, ensuring code quality, and driving technical decisions. Holds a Computer Science degree from a top-tier university.  Soft skills: Excellent leadership and communication skills, able to collaborate effectively with cross-functional teams. Strong problem-solving abilities and a proactive approach to addressing technical challenges.  Areas of improvement: Occasionally tends to be overly critical, should focus on fostering a positive team culture and recognizing individual contributions. |
| Full-stack Developer | Michael | Location: London  Expertise, frameworks: Full-stack developer with 5 years of experience. Proficient in Python, Django, React, JavaScript, and PostgreSQL. Experienced in developing RESTful APIs, implementing front-end designs, and optimizing database performance. Holds a degree in Software Engineering.  Soft skills: Collaborative team player, good at providing constructive feedback and actively participating in code reviews. Adaptable and open to learning new technologies and frameworks.  Areas of improvement: Could improve time management skills to meet project deadlines more effectively. Should focus on enhancing communication and collaboration with team members |
| Full-stack Developer | Lea | Location: Berlin  Expertise, frameworks: Full-stack developer with 4 years of experience. Proficient in Node.js, Express.js, Angular, MongoDB, and AWS. Skilled in building scalable and maintainable web applications. Holds a degree in Computer Science.  Soft skills: Strong problem-solving skills, actively contributes ideas and suggestions during team discussions. Detail-oriented and committed to delivering high-quality code. Works well under pressure and meets project deadlines.  Areas of improvement: Could enhance knowledge in testing frameworks and methodologies. Should focus on improving documentation skills for better code maintainability. |
| Front-end developer | Avinash | Location: Sydney  Expertise, frameworks: Front-end developer with 3 years of experience. Proficient in HTML, CSS, JavaScript, React, and Vue.js. Skilled in creating responsive and intuitive user interfaces. Holds a degree in Web Development.  Soft skills: Collaborative team player, actively participates in design discussions and provides valuable input. Detail-oriented and focused on delivering pixel-perfect designs. Adaptable to changing project requirements.  Areas of improvement: Could expand knowledge in UI/UX best practices. Should work on improving cross-browser compatibility and accessibility aspects. |
| Tester | Graham | Location: Toronto  Expertise, frameworks: Junior/Mid-level tester with 3 years of experience. Skilled in manual testing methodologies, test case creation, and execution. Proficient in bug tracking tools such as Jira and test management tools like TestRail. Has a solid understanding of software development lifecycle (SDLC) and testing processes. Holds a degree in Business Administration.  Soft skills: Detail-oriented and thorough in test case execution and defect reporting. Strong analytical skills and ability to think critically. Collaborative team player, actively participates in team discussions and contributes ideas.  Areas of improvement: Eager to learn test automation methodologies and tools to enhance efficiency. Should focus on gaining more knowledge in scripting languages like Python or Java for writing automated test scripts. Seeks mentorship and opportunities to gain hands-on experience in test automation. |
| UX/UI Designer | Dria | Location: New York  Expertise, frameworks: UX/UI designer with 5 years of experience. Proficient in user research, wireframing, prototyping, and visual design tools such as Sketch and Figma. Skilled in creating intuitive and visually appealing user interfaces. Holds a degree in Interaction Design.  Soft skills: Excellent communication and collaboration skills, actively engages with stakeholders to understand user requirements. Detail-oriented and focused on delivering user-centric designs. Open to feedback and iterations.  Areas of improvement: Could enhance knowledge in front-end development to better understand technical feasibility. Should focus on continuously updating design skills to stay abreast of industry trends. |
| Product Manager | Jesse | Location: Seattle  Expertise, frameworks: Product manager with 6 years of experience in software development. Skilled in product strategy, roadmap planning, and agile methodologies. Experienced in gathering user feedback, conducting market research, and defining product requirements. Holds an MBA degree.  Soft skills: Strong leadership and decision-making skills, capable of prioritizing and managing competing demands. Excellent communication and stakeholder management abilities. Collaborative and supportive of the development team's needs.  Areas of improvement: Could enhance technical knowledge to better understand development complexities. Should focus on balancing user needs with business objectives to drive product success. |

### Current Status of QA Process (QA Audit)

Maturity stages:

1. Not established
2. Start-up
3. Growth
4. Expansion
5. Mature

| Aspect / area / milestone | Tools | Maturity stage |
| --- | --- | --- |
| Test strategy | - | Start-up |
| Test plan | - | Not established |
| Defect management | GitHub Issues, Plane App |  |
| Test management | - |  |
| Manual testing | - | Start-up |
| End-to-End UI,  Integration Level testing | - | Not established |
| API Testing | - | Not established |
| Unit Testing | Python, unittest | Start-up |
| Automated regression testing | - | Not established |
| Automated E2E Testing | - | Not established |
| Acceptance Testing | - | Not established |
| Automated Production Validation Testing  (PVT Smoke Test) | - | Not established |
| Static Analysis |  | Not established |
| Exploratory Testing | - | Not established |
| Usability testing | - | Not established |
| Automated compatibility testing | - | Not established |
| PR reviews | GitHub | start-up |
| Coding Guidelines | GitHub | start-up |
| CI/CD pipeline | GitHub Actions | start-up |
| Production monitoring and alerts | - | Not established |
| Uptime monitoring | - | Start-up |

### Testing Approach

This part describes the types of tests that will be used during product testing, the testing pyramid, and its stack. Also, it prioritizes such types of tests as compatibility testing, installation testing, etc. For convenience, we will break it into subsections.

### Testing Levels

### 

| Testing type | Tools | Manual / Automated |
| --- | --- | --- |
| Manual testing | Azure DevOps, Azure Test Plans | Manual |
| End-to-End UI, Integration Level | Playwright, TypeScript, Azure Pipelines | Automated |
| API Testing | Postman, Newman, Javascript, GitHub Actions | Automated |
| Unit Testing | Python, pytest, GitHub Actions | Automated |

### Compatibility Testing Priorities

How we can identify priority?

* Do we have usage data to identify which devices, OS, screens are most popular among users? (NO, but in real project we could employ tools like Azure App Insights to collect data)
* Can we use publicly available data reflecting most popular OS, Browsers, and screens? (Yes)
* Typical audience: Software development teams and organisations
* [Computer operating systems market share 2012-2023 | Statista](https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/)
* [Stack Overflow Developer Survey 2023](https://survey.stackoverflow.co/2023/#section-most-popular-technologies-operating-system)
* [The State of Developer Ecosystem in 2022 Infographic | JetBrains: Developer Tools for Professionals and Teams](<https://www.jetbrains.com/lp/devecosystem-2022/>)
* [Browser Market Share Worldwide | Statcounter Global Stats](https://gs.statcounter.com/browser-market-share)

Top 3 OS (desktop)

1. Windows (~70%)
2. OSX (~17)
3. Linux (~3%)

Top 3 browsers (desktop)

1. Chrome (~65%)
2. Safari (~11%)
3. Edge (~10%)

Top 3 screen resolutions (desktop)

1. 1920x1080 (~22%)
2. 1366x768 (~15%)
3. 1533x864 (~10%)

| **OS** | **Windows** | **Windows** | **Windows** | **MacOS** | **MacOS** |
| --- | --- | --- | --- | --- | --- |
| **Browser** | Chrome | Edge | Chrome | Chrome | Safari |
| **Screen resolution (resolution, screen size, display ratio)** | 1920x1080 | 1920x1080 | 1366x768 | 1920x1080 | 1920x1080 |
| **Priority** | High | High | Medium | Medium | Low |

### 

### 

### Testing Types

### 

| test type | test objective(s) | process details | acceptance criteria |
| --- | --- | --- | --- |
| Feature Testing |  |  |  |
| API Testing | test objective(s) | process details | acceptance criteria |
| System Testing | Verify whether the functional and non-functional behaviors of the system are as designed and specified. Validate that the system is complete and will work as expected |  |  |
| Regression Testing | Verify that a code change in the software does not impact the existing functionality of the product. |  |  |
| Integration Testing | Verify interactions between components |  |  |
| Unit Testing | Ensure that the individual parts of a program work properly on their own, speeding up testing strategies and reducing wasted tests |  |  |
| E2E Testing | Tests the application’s workflow from beginning to end to make sure everything functions as expected |  |  |
| Acceptance Testing | Validate that the system is complete and will work as expected. Focuses on the behavior and capabilities of a whole system. Verify that functional and non-functional behaviors of the system are as specified. |  |  |
| Production Validation Testing (PVT Smoke Test) | Validate the stability of a software application after deploying new release to the production environment. |  |  |
| Static Analysis |  |  |  |
| Exploratory Testing | Explore the application or particular area to learn about it and look for defects. |  |  |
| Usability testing |  |  |  |

Priority

- High - testing must be carried out in full.

- Medium - testing can be carried out partially.

- Low - testing will be performed if there is time left.

### Impediment mitigation

Impediment mitigation describes a list of potential problems a product may have with its quality, as well as the types of testing that aim to reduce these risks and their priority.

Ways to identify product risks

1. Learn the product. E.g. what are the main features, tech stack, target audience?
2. Learn the product domain.
3. Talk with users, clients.
4. Analyse issues reported by users.
5. Analyse both functional and non-functional potential issues.
6. Check examples of product risk analysis examples.

* (F) Functional areas
* Non-functional
  + (S) Security
  + (UI) Usability
  + (P) Performance
  + (C) Compatibility
  + (I) Installability
  + (R) Recoverability

| Testing Type | Priority |
| --- | --- |
| Feature testing | High |
| API Testing | High |
| System Testing |  |
| Regression Testing | High |
| Integration Testing |  |
| Unit Testing | High |
| E2E Testing | High |
| Acceptance Testing | Medium |
| Production Validation Testing  (PVT Smoke Test) | Medium |
| Static Analysis | High |
| Exploratory Testing | High |
| Usability testing | Medium |
| Performance testing | Medium |
| Compatibility testing | Low |
| Security testing | High |
| Engaging with the community on the Discord channel, reviewing GitHub issues | Medium |
| Engaging with the community Twitter, Linkedin, YouTube | Low |
| Reviewing and updating documentation before each release | Low |

Based on this table, we can break down our test types into priorities:

| Area | Testing Type | Impact | Mitigation |
| --- | --- | --- | --- |
|  | Failure to sign up | High | Feature testing  Regression testing  Exploratory Testing  Integration Testing  E2E Testing  Unit Testing  API Testing |
|  | Failure to login | High |
|  | Failure to create project | High |
|  | Failure to create issue | High |
|  | Failure to update issue | High |
|  | Failure to create Cycle | Medium |
|  | Failure to invite new member | High |
|  | Failure to set member role | Medium |
|  | Failure to update project settings | Medium |
|  | Unable to view notifications | Low |
| F | Failure to display information on Dashboard correctly | High |
| Failure to display information on Analytics page correctly | Medium |
| Failure to create Module | Medium |
| Failure to create View | Medium |
| Command menu failure | Medium |
| AI-assistant failure | Low |
| Integraition failure: GitHub, ChatGPT, Slack, GitHub Importer | Medium | Feature testing  Regression testing  Exploratory Testing  Integration Testing  E2E Testing |
| No easy way to migrate from other products | Low | Acceptance testing  Exploratory Testing |
| Poor or missing documentation | Low | Providing the ability to submit GitHub issues  Discord channel  Reviewing and updating documentation before each release |
| S | Login security | High | Security testing  Exploratory testing |
| Incorrect handling of user permissions, roles, access to resources (cross-organisations, cross-workspaces, cross-projects) | Medium | Security testing  Exploratory testing |
| U | Poor usability | Medium | Usability testing  Acceptance testing  Engaging with the community on the Discord channel, Twitter, Linkedin, YouTube |
| P | Slow response from the app.  Performance issues: data (volume)  Performance issues: number of users (scalability)  Performance issues: number of transactions | Medium | Volume testing  Load testing  Stress testing |
| C | Compatibility issues (browser, device, screen) | Low | Compatibility testing  Reviewing issues reported by users  Engaging with community in Discord channel |
| I | Failure to run local version using Docker or any issues with self-hosted version | High | Reviewing issues reported by users  Engaging with community in Discord channel  Compatibility testing |
| R | For cloud version, failure to handle properly interrupted network connection (data upload when creating, updating issue, integration) | Medium | Recoverability testing  Exploratory testing |

### Testing phases

Testing activities that take place during the following phases

| Testing phases | Activities | Metrics | Entry criteria | Exit criteria |
| --- | --- | --- | --- | --- |
| Plan | 1. Collaboration between developers, PO/PM, testers, BAs. 2. 3 Amigos session if required 3. Defining testing approach, scope, testing types, what should be automated | Stakeholder feedback  Story/task quality (poor/  acceptable, good, excellent ) |  | All questions are answered   1. Acceptance criteria are clear and testable 2. Both negative and positive parts are covered (testing scenarios or acceptance criteria) 3. Extra tests or monitoring required - yes/no. 4. Is this user story releasable and testable on its own? yes/no 5. Feature flagging is reasonable? yes/no 6. Are there dependencies on other teams? 7. Design mocks required/available? yes/no 8. Do we know the motivation for building this story? Is the main persona clear? |
| Branch |  |  |  |  |
| Code | 1. New unit tests are written (Dev) 2. Integration tests are written 3. Exploratory testing is completed (Dev+QA) | 1. Unit test coverage 2. Integration test coverage 3. Exploratory testing reports |  |  |
| Merge | 1. PR review |  |  |  |
| Build | 1. CI pipeline build 2. Static code analysis (A) 3. Unit test execution (A) 4. Integration test: critical path (A) | 1. Build status 2. Failure rate 3. Test execution results |  |  |
| Test | 1. Feature testing, exploratory testing (MQA) 2. System testing (AQA) 3. Regression testing (AQA) | 1. List of defects (exploratory) 2. Failure rate 3. Automation test execution results |  |  |
| Release | 1. Review test results from previous stage, defect count, severity 2. Check against acceptance criteria 3. Check against release criteria | 1. Defect count and severity, 2. Release decision (go/no-go) status |  |  |
| Deploy | 1. Automated backup 2. Production validation testing (automated smoke test) 3. Manual PVT if required | 1. Automation test execution results 2. Manual test execution results 3. Defect count and severity |  |  |
| Operate | 1. Engaging with the community on the Discord channel, reviewing GitHub issues (QA, Devs) 2. Handle user inquiries, feedback, and issues, and escalate any critical incidents to the operations team | 1. Defects count: GitHub, Internal tracker 2. Defect severity 3. List of feature requests, questions (can use tags, upvotes) |  |  |
| Monitor | 1. Set up monitoring tools 2. Define alert thresholds for new features | 1. Server response times 2. Error rates, 3. Feature usage statistics |  |  |

### Reporting

Metrics and dashboards that are available and updated **automatically**

| **Phases** | **Metrics, reports** | **Ownership** |
| --- | --- | --- |
| Code | 1. Unit test execution results and test coverage 2. Integration test execution results and coverage | Unit: Dev  Integration: Dev / test engineer |
| Build | 1. Build status 2. Failure rate 3. Test execution results | Unit: Dev  Integration: Dev / test engineer |
| Test | 1. List of defects (exploratory testing) 2. Failure rate (regression, system) 3. Automation test execution results (regression, system testing) | Exploratory: QA  System, regression: Dev / test engineer |
| Release | 1. Defect count and severity (user release or sprint tags), 2. Release decision (go/no-go) status | 1. Dev team, QA 2. PO, dev team, QA |
| Deploy | 1. Automation test execution results 2. Manual test execution results | 1. Dev / test engineer 2. QA |
| Operate | 1. Defects count: GitHub, Internal tracker 2. Defect severity 3. List of feature requests, questions (can use tags, upvotes) |  |
| Monitor | 1. Server response times 2. Error rates 3. Feature usage statistics | Dev / test engineer |

Reports prepared **manually**

| **Phases** | **Metrics, reports** | **Ownership** |
| --- | --- | --- |
| Plan | 1. Stakeholder feedback 2. Story/task quality (poor/acceptable, good, excellent ) | QA |
| Code | 1. Exploratory testing results: critical issues identified by devs arefixed | Dev |
| Release | 1. Defect count and severity, 2. Release decision (go/no-go) status | QA: qualitative metrics report  Conversation between PO, dev team, QA, other stakeholders |
| Deploy | 1. Defect count and severity | QA, Dev |
| Operate | 1. Defects count: GitHub, Internal tracker 2. Defect severity 3. Prioritized list of feature requests, questions (can use tags, upvotes) | 1. QA, Dev 2. QA, Dev 3. PO |

### Hotfix

### CI/CD testing pipeline

### Links

- [How To Craft a Test Strategy Document [Sample Test Strategy Inside]](https://tech-stack.com/blog/test-strategy/)

- [How to Write a Test Strategy Document for Software Testing - NearForm](https://www.nearform.com/blog/how-to-write-software-test-strategy-document/)

- [Plane Documentation](https://docs.plane.so/)

- [Plane - The open source project management tool](https://plane.so/)

- [[testing in devops stages and collaboration with team explained]]