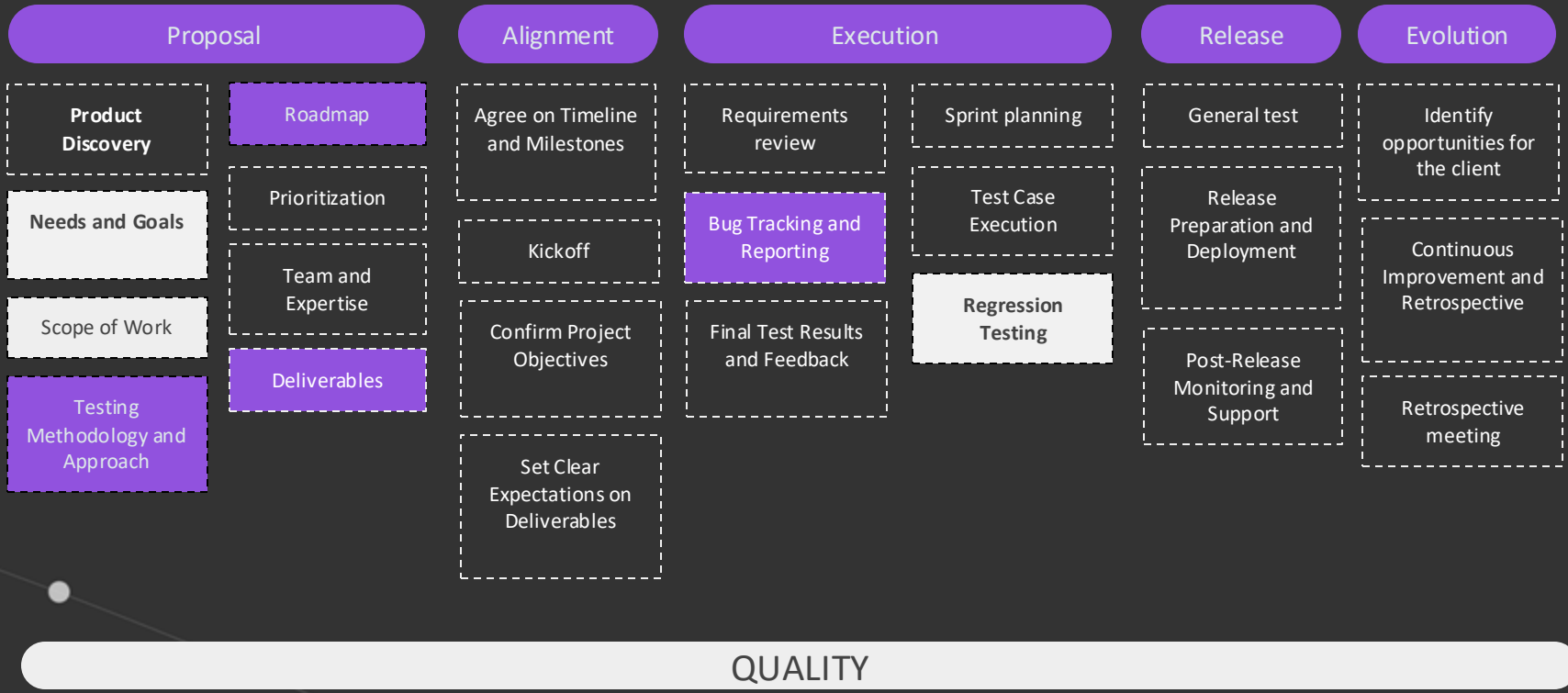
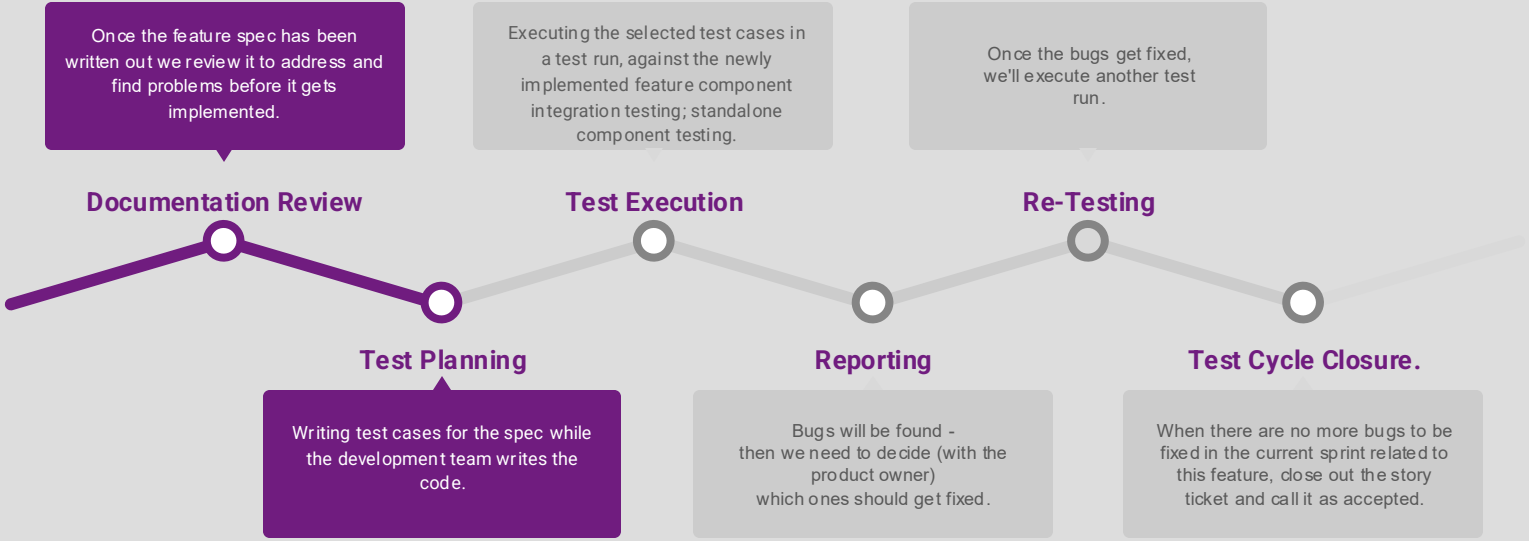
Project Phases

Agile Team Journey



How we work


Once the feature spec has been written out we review it to address and find problems before it gets implemented.
Documentation Review
Test Planning
Writing test cases for the spec while the development team writes the code.
Executing the selected test cases in a test run, against the newly implemented feature component integration testing; standalone component testing.
Test Execution
Reporting
Bugs will be found - then we need to decide (with the product owner) which ones should get fixed.
Once the bugs get fixed, we'll execute another test run.
Re-Testing
Test Cycle Closure.
When there are no more bugs to be fixed in the current sprint related to this feature, close out the story ticket and call it as accepted.

Project Phases

Phase 1
QA team induction

- Kick-off meeting
- Acquire necessary access and resources
- Establish a testing environment
- Understand the product

Phase 2
Set objectives and suggest improvement

- Define objectives and KPIs
- Identify the areas where QA team can add value and suggest improvements

Phase 3.1
Set up functional testing

- Review and test new features
- Test case creation and management
- Defect tracking and reporting
- Regression testing
- API testing

Phase 3.2
Set up non-functional testing

- Accessibility and usability testing
- Understand demographics and prioritize testing environments
- Cross-browser, cross-device, and cross-platform testing
- Continuous Integration and Continuous Delivery (CI/CD) pipeline integration
- Security testing
- Performance testing
- Bandwidth connectivity testing

Phase 4
Set up post-release monitoring

- Post-release monitoring and support
- Reporting and communication

Phase 1 - QA team induction

01	**Kick-off meeting**	• **Introduce** the QA team to the rest of the SDLC team. • **Present** an overview of the QA process and procedures • **Discuss** expectations and requirements for the project • **Determine** project timelines and milestones • **Identify** stakeholders and their roles
02	**Understand the product**	• **Review** available documentation and requirements to gain a deeper understanding of the product • **Collaborate** with the development team to understand in-progress work and any upcoming features • **Perform** ad hoc testing on the most important parts of the product, as specified by the customer
03	**Acquire necessary access and resources**	• **Request** access to communication channels (e.g., Slack) • **Obtain** product access and any required credentials • **Request** a testing environment and information about product versioning
04	**Establish a testing environment**	• **Set up** TestRail for test case management • **Integrate** JIRA for ticketing and Confluence for documentation review • **Prepare** Cypress for web automation, Espresso or Swift for mobile automation, and BrowserStack as a cloud device farm • **Ensure** that the testing environment is properly set up with all the necessary configurations • **Ensure** that you have test accounts and test data that support testing scenarios

Phase 2: Set objectives and suggest improvements

01 **Define objectives and KPIs**

- **Define** the scope and objectives of the QA team: Determine the responsibilities of the new QA team and what the team should aim to achieve
- **Identify** key performance indicators (KPIs) that will be used to measure success

02 **Add value and suggest improvements**

- **Introduce** a formal testing process that includes test cases, test plans, and test scripts to ensure consistency
- **Introduce** a defect management process to track and manage defects found during testing
- **Ensure** that testing is carried out early throughout the development lifecycle and not just at the end
- **Support** regular retrospectives to continuously improve the software development process

Phase 3.1: Set up functional testing

01 **Review and test new features**

- **Review** the documentation or requirements before the feature gets implemented
- **Examine** newly implemented features before they are released
- **Work** closely with the development team to provide feedback and identify issues
- **Create** and execute test plans for each feature, reporting progress and results

02 **Test case creation and management**

- **Create** detailed test cases and test scenarios based on product requirements and specifications
- **Organize** test cases into logical groups and maintain them in TestRail
- **Update** test cases as needed to reflect changes in product requirements or design
- **Do we** run test runs from which we can extract a test report for the builds you are testing?
- **When** we run a test run, do we add the version of the build on which the test run was done?
- **When** we fail a test case, do we mention the ticket for which the TC failed?

Phase 3.1: Set up functional testing

03 Regression testing

- **After** bug fixes and new features implementation, perform regression testing to ensure that previously tested functionality still works as intended
- **Continuously** update the regression test suite to cover new features and changes
- **Automate** regression testing, where possible, to improve efficiency and reduce manual effort

04 API testing

- **Design** and create test cases for each API function, focusing on input validation, authentication, authorization, error handling, and response consistency
- **Use** API testing tools, like Postman or SoapUI, to execute tests and validate responses against the expected results
- **Test** the API for performance, load, and stress to ensure it can handle the expected number of requests and respond within acceptable timeframes
- **Collaborate** with the development team to address and resolve any issues found during API testing

05 Defect tracking and reporting

- **Log** discovered defects and issues in the ticketing system (e.g., JIRA)
- **Categorize** and prioritize issues based on severity and impact
- **Collaborate** with the development team to ensure timely resolution and retesting of issues
- **Are** newly found defects added to the existing test cases
- **Are** we making sure that the needed bugs that need to get fixed are getting fixed before a release of that feature is happening?
- **When** adding new bugs, do we also add screenshots or videos as proof? Do we add the version of the product on which the problem occurred?
- **When** we close a bug, do we add a screenshot or video proof?
- **Do** we mention the version of the build with which you closed the ticket?
- **When** we add bugs related to a story, do we link those bugs to the story and track to see which bugs need to get fixed in order for us to close the story?

Phase 3.2: Set up non-functional testing

01	**Accessibility and usability testing**	• **Review** the product for compliance with accessibility guidelines and standards, such as the Web Content Accessibility Guidelines (WCAG) for web applications • **Perform** usability testing to evaluate how user-friendly the product is and identify any issues that may hinder user experience • **Collaborate** with the development and design teams to address and resolve any identified accessibility or usability issues
02	**Understand demographics and prioritize testing environments**	• **Gather** information on the target audience and their device preferences • **Analyze** usage statistics for different devices, browsers, and operating system versions • **Based** on demographics and usage data, prioritize testing on the most commonly used devices, browsers, and operating systems
03	**Cross-browser, cross-device, and cross-platform testing**	• **Perform** testing across various browsers (for web applications) and devices (for mobile applications) to ensure compatibility and consistent user experience • **Use** cloud device farms like BrowserStack to test on various devices, browsers, and operating system versions • **Address** any compatibility issues found during testing and collaborate with the development team to resolve them
04	**Continuous Integration and Continuous Delivery (CI/CD) pipeline integration**	• **Integrate** test automation suites (Cypress for web automation, Espresso or Swift for mobile automation) with the CI/CD pipeline • **Ensure** that tests are executed automatically whenever new code changes are pushed to the code repository • **Monitor** test results and collaborate with the development team to address any issues during the automated test execution

Phase 3.2: Set up non-functional testing

05

Security testing

- **Input** Validation Testing: Validate user input to prevent attacks like cross-site scripting (XSS), SQL injection, and other injection attacks. Test the system with malicious inputs to see if it can handle them securely
- **Authentication** and Authorization Testing: Verify that the authentication and authorization mechanisms are robust and can't be bypassed or exploited. Test for weak passwords, improper session handling, and unauthorized access to sensitive resources
- **Configuration** and Deployment Testing: Assess the security of the product's configuration and deployment settings. Check for default credentials, insecure configuration files, and unnecessary open ports
- **Sensitive** Data Exposure: Test the system to ensure that sensitive data is protected, both in transit and at rest. Check for data leakage through logs, error messages, or insecure storage
- **Web** Services and API Security Testing: Evaluate the security of any web services or APIs used in the product. Test for authentication, authorization, and input validation vulnerabilities
- **Business** Logic Testing: Identify vulnerabilities in the product's business logic that may allow attackers to manipulate or bypass intended workflows

Phase 3.2: Set up non-functional testing

06	**Performance testing**	• **Conduct** performance testing using tools like JMeter to evaluate the product's performance under various load conditions • **Identify** and address any bottlenecks or performance issues impacting the end-user experience • **Collaborate** with the development team to optimize the product's performance based on the testing results
07	**Bandwidth connectivity testing**	• **Test** the product under various network conditions (e.g., slow or unstable connections) to ensure it functions properly even with limited or fluctuating bandwidth • **Identify** and address any issues that may arise due to poor network connectivity • **Work** with the development team to optimize the product's performance in low-bandwidth environments

Phase 4: Set up post-release monitoring

01	**Post-release monitoring and support**	• **Monitor** the product's performance and user feedback after release to identify any issues that may not have been uncovered during testing • **Collaborate** with the development team to address any post-release issues and deploy fixes in a timely manner • **Continuously** refine the testing process and test suite to ensure ongoing product quality and stability
02	**Reporting and communication**	• **Regularly** update the project stakeholders on testing progress, results, and any encountered issues • **Collaborate** with the development team to ensure efficient communication and knowledge sharing • **Use** Confluence to document testing processes, test results, and any related information that needs to be shared with the team

Conclusion

This comprehensive QA process covers various aspects of testing, including functional, compatibility, accessibility, usability, security, performance, and connectivity testing. By following these procedures, the QA team can ensure that the product meets the highest quality standards and provides an exceptional user experience. Ongoing collaboration, communication, and continuous improvement will help the team adapt to changing project requirements and maintain a high level of quality assurance throughout the product's lifecycle.


Thank you for your time!
BQA
BETTER QUALITY ASSURANCE