Industry Anticipations from an Entry-Level Software Testing

WORLDLINE

Phase	Description	Activities	Purpose
Test Planning	Defining the testing scope, objectives, and resources required for the testing process.	- Identifying testing goals and objectives. - Defining testing scope and coverage. - Allocating resources. - Creating test plans and schedules.	Establish a clear plan to guide the testing process and ensure proper resource allocation.
Test Design	Creating detailed test cases and test scripts based on requirements and design specifications.	- Identifying test scenarios and cases. - Designing test data and environment. - Creating test scripts.	Develop a comprehensive set of test cases that cover various scenarios to ensure thorough testing.
Test Environment Setup	Preparing the testing environment with necessary hardware, software, and data.	- Setting up required hardware and software. - Loading test data into the environment. - Configuring test environments.	Ensure a stable and controlled environment to conduct testing without interference.
Test Execution	Running test cases, scripts, and scenarios to identify defects and ensure the software's functionality.	- Executing test cases and scripts. - Recording test results and defects. - Analyzing test outcomes.	Identify defects, validate system functionality, and collect data on the software's performance and behavior.
Defect Reporting	Documenting and reporting defects discovered during testing.	- Logging defects with detailed information. - Prioritizing defects based on severity. - Communicating defects to stakeholders.	Ensure that defects are accurately documented, reported, and communicated to facilitate resolution.
Defect Resolution	Developers address reported defects, fix issues, and retest the modified software.	- Developers analyze defects and make fixes. - Re-testing modified code to verify defect resolution.	Correct identified defects, ensuring that the software meets the required quality standards.
Retesting	Re-executing the previously failed test cases after defect resolution.	- Running the same test cases that previously failed. - Verifying whether the defects have been fixed.	Ensure that the reported defects have been successfully resolved and do not reintroduce new issues.
Regression Testing	Re-running selected test cases to ensure new code changes have not adversely affected existing functionality.	- Selecting and running a subset of test cases from the entire test suite. - Verifying that new changes have not impacted existing functionality.	Prevent unintended side effects by validating that new code changes do not disrupt previously working features.
Test Closure	Summarizing testing activities and evaluating if exit criteria are met.	- Comparing actual results against expected results. - Analyzing the achieved coverage. - Generating test summary reports.	Determine whether testing goals are met, provide a summary of testing outcomes, and decide if the software is ready for release.

Testing Type	Description	Scope	Purpose
Unit Testing	Testing individual components or functions in isolation to ensure their correctness.	Focuses on a single unit/module.	Verify that each unit works as intended, catch bugs early, and facilitate code maintenance.
Integration Testing	Testing interactions between multiple components or modules to detect defects in their integration.	Focuses on the interfaces between units/modules.	Identify issues arising from the combination of components and ensure proper communication.
System Testing	Testing the entire software system as a whole to verify its compliance with specified requirements.	Encompasses the entire application.	Validate the system against functional and non-functional requirements before release.
Acceptance Testing	Testing performed to ensure that the system meets the business requirements and is acceptable.	Represents end-users' perspective.	Confirm that the system satisfies user needs, often the final testing phase before release.

Aspect	Black Box Testing	White Box Testing
Perspective	External perspective; treats software as a "black box."	Internal perspective; examines code, logic, and structure.
Knowledge	Testers have no knowledge of the internal code structure.	Testers have access to the internal code and logic.
Focus	Functionality and user requirements.	Code structure, logic, and adherence to programming rules.
Test Case Design	Based on specifications and requirements.	Based on code paths, branches, and logic.
Code Coverage	Typically lower code coverage.	Aims for high code coverage by testing various paths.
Detection of Errors	Focuses on incorrect functionality and usability issues.	Detects logic errors, coding defects, and integration issues.
Independence	Can be conducted independently of developers.	Might require collaboration with developers for insights.
Usability Testing	May uncover user-related issues and usability problems.	Not typically designed for comprehensive usability testing.
Integration Testing	Focuses on testing integrated components from a user's view.	Can test interactions between components and code paths.
Skill Requirements	Does not require deep coding knowledge.	Requires programming expertise to design effective tests.

Testing Methodology	Description	Key Characteristics	Advantages	Disadvantages
Waterfall	A linear, sequential approach where each phase must be completed before the next one begins.	Structured, clear requirements, well-defined process.	Predictable, suitable for well-defined projects.	Limited flexibility, late feedback.
Agile	An iterative and collaborative approach that focuses on delivering small increments of working software.	Flexibility, adaptability, customer collaboration.	Quick iterations, continuous improvement.	Requires active client involvement.
Scrum	An Agile framework that divides work into time-bound iterations (sprints) and emphasizes collaboration.	Time-boxed sprints, defined roles, regular reviews.	High adaptability, teamwork, customer involvement.	Can be challenging for complex projects.
Kanban	An Agile approach that visualizes workflow on a board, aiming for continuous delivery and reduced work-in-progress.	Visual representation, flow optimization, flexibility.	Reduced bottlenecks, adaptive, efficient.	May lack structured planning, can be complex.
DevOps	A methodology that emphasizes collaboration between development and operations teams for continuous delivery.	Continuous integration, continuous delivery, automation.	Faster deployment, higher quality, collaboration.	Requires cultural and organizational changes.

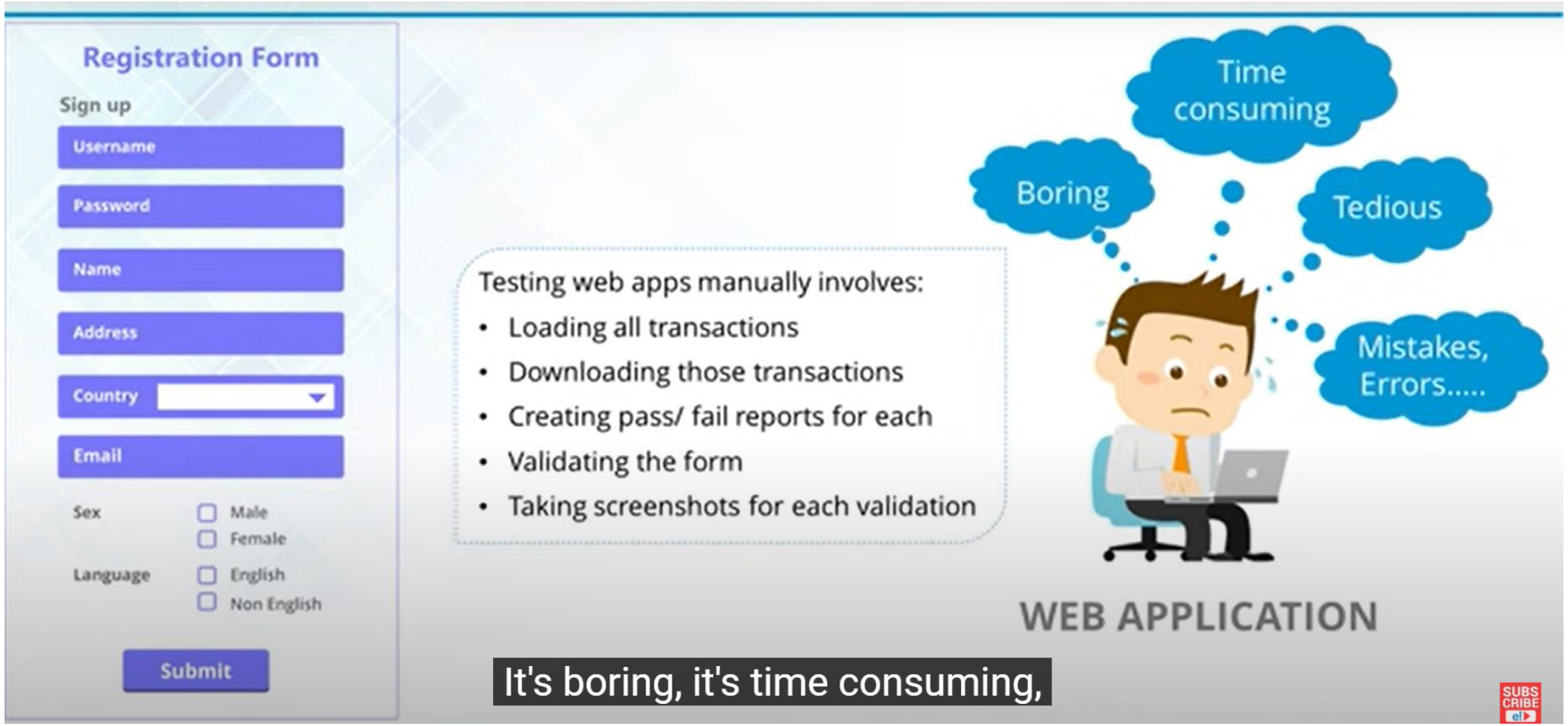
Aspect	Test-Driven Development (TDD)	Behavior-Driven Development (BDD)
Focus	Primarily on the code and its functionality.	Focuses on behavior, collaboration, and communication.
Test Creation Order	Tests are written before the actual code implementation.	Tests are written based on user behaviors and requirements.
Test Scope	Focused on individual units or components.	Encompasses user scenarios and system behaviors.
Test Granularity	Tests are often fine-grained, targeting specific code features.	Tests are more user-centric, covering end-to-end behaviors.
Test Design	Tests are designed to validate specific code behavior.	Tests are designed to verify system behaviors from a user's perspective.
Collaboration	Collaboration between developers and testers is essential.	Encourages collaboration between developers, testers, and non-technical stakeholders.
Language	Generally uses programming languages like Java, C#, etc.	Often employs plain language and user-focused terminology.
Automation	TDD encourages automated testing to validate code frequently.	BDD promotes automated acceptance testing for user behaviors.
Test Maintenance	Tests need to be updated as code changes.	Tests might need less frequent updates due to focus on behavior.
Documentation	TDD documentation is closely tied to code behavior and logic.	BDD documentation is more user-focused, aiding in understanding system behaviors.
Process Integration	Often integrated into the development workflow, with short cycles.	Can be integrated into Agile processes to enhance collaboration.
Common Tools/Frameworks	JUnit, NUnit, pytest, etc.	Cucumber, SpecFlow, Behave, etc.

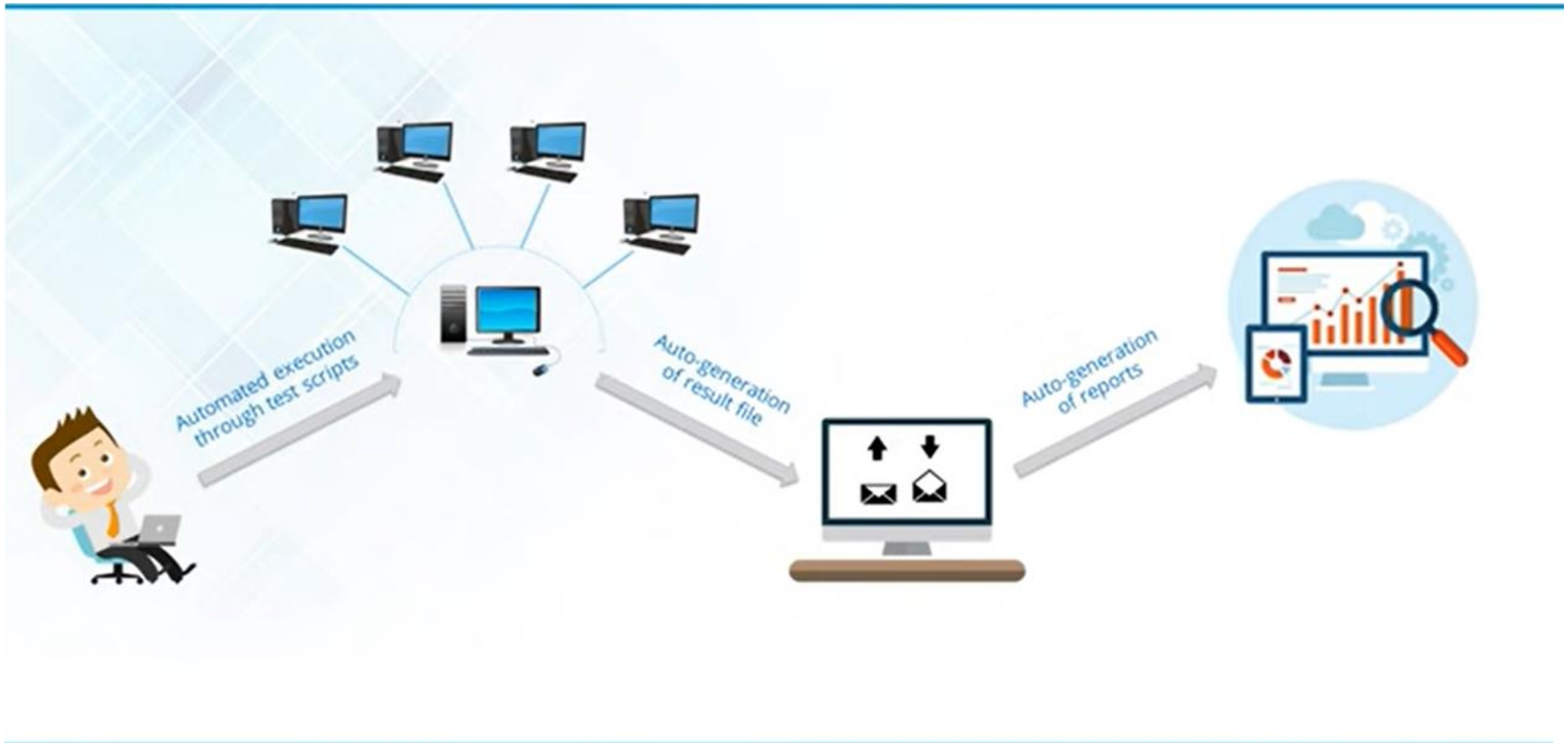
Manual Testing

Testing web apps manually involves:

- Loading all transactions
- Downloading those transactions
- Creating pass/ fail reports for each
- Validating the form
- Taking screenshots for each validation

WEB APPLICATION


Registration Form
Sign up
Username
Password
Name
Address
Country
Email
Sex
Male
Female
Language
English
Non English
Submit
Testing web apps manually involves:
• Loading all transactions
• Downloading those transactions
• Creating pass/ fail reports for each
• Validating the form
• Taking screenshots for each validation
Time consuming
Boring
Tedious
Mistakes, Errors.....
WEB APPLICATION
It's boring, it's time consuming,


Automated execution through test scripts
Auto-generation of result file
Auto-generation of reports

WORLDLINE

Features of Automation Testing

Faster Execution

More Accurate

Lesser Investment In Human Resource

Supports Regression Testing

Frequent Executions

Supports Lights Out Execution

Selenium is a suite of software tools to automate web browsers.

It is open source and mainly used for functional testing and regression testing.

→ Supports different PL → Java, Python, C#, PHP, Ruby, Perl, JavaScript

→ Supports different OS → Windows, Mac, Linux, iOS, Android

→ Supports different Browsers → IE, Firefox, Chrome, Safari, Opera


DISADVANTAGES
Supports only web applications
Only user communities available
Difficult to setup and use
No reporting facility
Limited support for image based testing
WWW
ADVANTAGES
WWW
Open source software
Supports multiple programming languages
Supports almost every OS
Supports multiple browsers
Supports parallel test execution
Provides support for frameworks: TestNG, JUnit & NUnit

Question 1: What is the primary focus of black box testing?
A) Internal code structure B) Code logic and behavior C) Code coverageD) User functionality and requirements Answer: D) User functionality and requirements

Question 2: Which testing methodology involves writing tests before implementing the code?
A) Behavior-Driven Development (BDD) B) Test-Driven Development (TDD) C) Waterfall D) Agile Answer: B) Test-Driven Development (TDD)

Question 3: Which testing methodology emphasizes collaboration between development and operations teams?
A) Kanban B) Waterfall C) DevOps D) Scrum Answer: C) DevOps

Question 4: In the testing life cycle, which phase focuses on identifying defects and ensuring the software's functionality?
A) Test Design B) Test Execution C) Defect Resolution D) Regression Testing Answer: B) Test Execution

Question 5: What is the primary goal of white box testing?
A) Validating user requirements B) Ensuring high code coverage C) Identifying defects in the functionality D) Testing external interfaces Answer: B) Ensuring high code coverage

Question 6: Which testing methodology involves testing interactions between different components or modules?
A) System Testing B) Unit Testing C) Integration Testing D) Acceptance Testing Answer: C) Integration Testing

Question 7: What is the key advantage of Behavior-Driven Development (BDD)?
A) Focus on code coverage B) Automated testing of user scenarios C) Independent testing from developers D) Thorough exploration of code paths Answer: B) Automated testing of user scenarios

Question 8: Which testing phase focuses on summarizing testing activities and evaluating if exit criteria are met?
A) Defect Resolution B) Retesting C) Test Closure D) Test Execution Answer: C) Test Closure

Question 9: Which Agile methodology involves dividing work into time-bound iterations known as sprints?
A) Kanban B) Scrum C) DevOps D) Waterfall Answer: B) Scrum

Question 10: Which testing type aims to validate the entire software system against specified requirements?
A) Unit Testing B) Integration Testing C) System Testing D) Acceptance Testing Answer: C) System Testing

Thank you!