# UI Testing

## Notes

**Elements of App :**

1. Database Layer ( API )
2. Business Layer ( API )
3. Presentation Layer ( GUI )

**Test Scenario :** how the service will be used in real life. ( +/- )

**Automated Testing** : Essential to ensure shorter release cycles and minimum baseline of quality

**Manual Testing** : Better option for UI Testing, since there’s a need to review the look and get a feel of the app design

**UI Testing :** Testing for features of any software that a user will interact with. Usually, it involves testing visual components (radio buttons, text boxes, toolbars etc) to ensure they meet the outlined requirements - both functionality( how the application handles user actions carried out using input devices AND if visual elements are displayed and working properly ) and performance. GUI ( Graphical User Interface )Testing is a subset of UI Testing.

Verifies that the app not only behaves as intended but empowers the users to make the most with least effort.

**Main aspects of UI Testing :**

1. Visual design
2. Functionality
3. Usability
4. Performance
5. Compliance

**Essential UI Test Cases :**

* Data type errors ( only valid data, including spelling errors )
* Field widths
* Navigational elements ( redirecting the user to the right page )
* Functional Validation and Interactive elements
* Adaptability ( zoom in/out nad different window sizes )
* Session expiry time
* Scalability ( many users )
* Overlapping of fields
* Onscreen instructions
* GUI Testing
  + Acceptance Errors ( for GUI elements )
  + Readability of font ( size and color )
  + Image quality and alignment
  + Positioning of elements
  + Text alignment
* Progress bar
* Type-ahead
* Table-scrolling ( scrolling should ensure proper scrolling and keep headers intact )
* Error Logging ( In case the system experiences a fatal error, make sure that it writes a detailed log file for review )
* Menu items
* Working Shortcuts
* Non-editable items
* Confirm action buttons
* Consistency in terms of visual elements
* End to End Process Test ( Ad;lldress the company’s pain point : Selling games. Millions of views on this page )

**Benefits of UI Testing :**

1. It faces the users!!
2. Seamless performance
3. Improved coverage
4. Detecting behavior differences
5. Supporting cross browser/platform testing
6. Easily integrate with bug-tracking tools
7. Helps to quickly generate reports

**Challenges in UI Testing :**

1. Constantly changing UI
2. Increasing testing complexity ( complicated features = complicated testing )
3. Time consuming
4. Maintaining UI Test scripts
5. Handling multiple errors
6. Image comparison ( handling different resolution of images )
7. Computing ROI ( Return of investment ) for UI Test automation

**Ways of Overcoming challenges of UI Testing :**

1. Maintaining standard best test practises
2. Prefer codeless automation testing tools
3. Choose common UI elements
4. Leverage object repository ( share repos which can help code maintenance )

**Responsive Testing**

**Cross Browser Testing**

## Component Testing

**Grid Control**

* Verify default values ( + editing default values )
* Verify labeling of components
* Verify alignment of the grid elements
* Tab sequence selection
* Verify events related to double clicking
* Verify that disabled elements are not clickable
* Keyboard press or right press shouldn’t be enabled for grid items
* Verify scrolling function of arrow buttons on page, wheel of the mouse, touch slider of keyboard and arrow keys from keyboard

**Hyperlink**

* Hyperlink should have a standard color to identify
* Verify that the hyperlink opens the correct page
* Single/Double or Enter or spacebar click of the mouse to open hyperlinks
* Ensure tab sequence counts all hyperlinks
* Try opening hyperlinks using
  + Ctrl + Enter
  + Ctrl + mouse click

**Image**

* Image quality (no blurry images)
* Verify if zoom in/out options on the image
* Verify rotation of image

**General Testing**

* Failure of validation should result in appropriate error messages on both client/use side
* Invalid messages for invalid actions
* Check for mandatory/default fields
* Short cuts ?
* Verify tab order
* After the error message, does the cursor return to the field in error that the user clicked?
* Consequence of Alt + Tab?
* Is data saved when the window is closed?
* Check for maximum field lengths?
* Verify if the menu has a hot-key sequence which will invoke it when needed?
* Verify consistency of mouse actions throughout the webpage
* Ctrl + F6 opening next tab within tabbed window
* Shift + Ctrl + F6 opens previous tab within tabbed window
* Ctrl + F4 closes the tabbed window and return to main screen

States of UI Components

1. UNFILLED
2. FILLED / ON
3. NORMAL / DEFAULT
4. HOVER
5. DISABLED
6. MASKED

# Front-end Testing

Web Applications consists of three tiers :

* Server
* Client
* Resources/info system

## Types

**Unit Testing**

* It’s the lowest level of testing
* Testing individual components for its functionality

**Acceptance Testing**

* Whether the system matches the business requirements and is it deliverable
* Ensures proper functioning of the user flows, user inputs, and designated actions

**Visual regression Testing**

* It involves capturing screenshots from the live site ( baseline )
* Using image comparison tools, we try to detect differences between two screenshots
* Usually at the end

**Accessibility Testing**

* Verify that everyone can access an application including senior users, users with hearing/ visual disabilities
* Usually involves compatibility checks with a screen reader

**Performance Testing**

* Verify stability, responsiveness and speed
* How does the site react in certain conditions?

**End-to-End Testing**

* Ensures smoothness of the application, maintains data integrity between systems and components.
* This test helps identify system dependency and set-up related issues

**Integration Testing**

* Testing the interaction of units within the system
* Exposes bugs during the merge of the units
* Test stubs/logs and test drivers enable easy integration testing

**Cross-Browser Testing**

* Ensure that the user has the same kind of experience on different browsers, OSs, devices and browser combinations
* Very important

**Need for front-end Testing**

1. Detecting client-side performance issues
2. Validating application behavior on different browsers
3. Improves user experience
4. Ensure seamless integration of 3rd party services

80-20 Rule of Web performance : 80 % front-end and the rest of backend. Optimization made on the client-end will return the greatest performance improvement

## Front-End performance Optimization

Determining how quickly the web page loads. With Web2.0, web apps have become more dynamic and stateless, so the client-side has become more of a performance hog.

* Directly affects the users
* Enhancing front-end performance has a huge impact on overall performance

Tools for performance optimization testing : YSlow and Pagespeed

Effective Optimization Rules:

* Reduce the number of HTTP requests
* CSS performance optimization : development teams must eliminate bloated CSS.

1. One trick is to use external CSS rather than inline/embedded CSS. This reduces the size of the HTML pages, thus reducing the bandwidth + increasing load speed
2. Use CSS Sprite to combine multiple images into a single background file. This reduce total image size and load time
3. Combine multiple properties using CSS Shorthand

* Minification
* Image optimization

1. Image compressions
2. Avoid scaling images in HTML

* Use CDN ( Content Delivery Network )
* Implementing GZip file
* Putting scripts at the bottom of the page
* Using external script files
* Eliminate external scripts
* Reduce DNS lookups
* Avoiding redirects if possible
* Minimizing cookie size
* Ensure application loads asynchronous

**Where does the error lie ? Front-end or Back-end**

Calculate TTFB ( Time to First Byte ) : Time between a client request and the arrival of the first byte on the browser. TTFB is an effective benchmark for separating server response time and connectivity speed from the front-end issues. If TTFB is low, but there’s a delay in the interactive content loading, then we can surely say that there is an issue in the front-end.

**Choosing a good front-end testing tool**

* Easy creation and maintenance of test scripts
* Insight into performance metrics
* Screen capturing/image comparison capabilities
* Decent reporting abilities
* Seamless integration into CI/CD pipelines
* Using headless browsers so that tests are faster
* Cut down DOM rendering

# Slider Test Cases

Slider

Carousel for displaying well-organized pieces of info one by one in a cycle with components including:

1. Container
2. Slide ( the space where the contents sits )
3. Navigation
4. Pagination

* Sliders are great at displaying lots if info within small space
* Display new additional features/ populer offers
* Creates a promotional landing page
* Show of products in different angles, highlighting main content
* Build trust and credibility in online audience
* Make good impression in the hero area
* Not the best user experience and not a good marketing strategy
* Could destroy the SEO ( Search Engine Optimization ) score
* Slow down website
* Stats say that sliders don’t help increase the conversion rate
* If old languages are used,then the slider may break the entire design and structure
* Many sliders don’t usually meet accessibility requirements

Tips for a good slider

1. Navigation must be smooth : Make sure that the user is in charge. Also ensure that it meets touch screen standards
2. Set delay time properly
3. Make sure that the first slide is an important one. Even if the user happens to not interact with slider in real time, they must have definitely seen the first image
4. Don’t overdo the effects
5. Make sliders accessible
6. Ensure that the various size/browser/device compatibility

([StackOverflow](https://sqa.stackexchange.com/questions/8846/functional-and-ui-test-cases-for-a-slider))

1. Ensure cross-browser compatibility
2. Ensure smoothness and automatic loading of the slides
3. Ensure the loading of slides
4. Ensure order of the slider images
5. Ensure whether the slides can be clicked and thumbnail displayed on mouse hovering
6. Ensure buttons corresponding to each slider image is displayed below the slider
7. Ensure whether clicking each button, the corresponding slideshow is displayed
8. Ensure whether button gets highlighted when corresponding slideshow loads
9. Ensure time interval between each slideshow is the same
10. Ensure whether on clicking an image on the slider, the corresponding page is displayed
11. Ensure whether administrator is able to change the image and the URL of the slides
12. Ensure that the image displayed should not get cropped
13. Ensure the corresponding slider image and title displayed
14. Ensure slider width and height

Grid Test Cases

Verifying Game Test Case

Opening Required URL Test Case

# Writing UI Test plans

* Documentation
  + Skills
  + Time
  + Requirements
* Prioritize
* Begin with basic functionality testing
* Then check if the UI matches the requirements or not
* Sanity testing after smoke tests
* Traceability matrix of the test scenarios and verify that all requirements are satisfied
* Make sections
  + Functional
  + Non-functional
  + Structural Testing
  + End-to-End Testing
  + Regression testing ( re-running previously successful tests after code changes to confirm no additional defects have been introduced )
* Include
  + Response testing
  + Cross-browser testing ( any device, size etc )
  + Integration Testing
  + System Testing

Test Cases

* Include negative and positive
* Consider corner cases
* Try to avoid dependent cases

# Common Mistakes in UI Testing

1. Subtle sale
2. Color scheme
3. Non-responsive web design
4. Non-cross browser compatible
5. Font size consistency
6. Complicated design pattern/navigation
7. Badly designed CTA
8. Text, typography and presentation
9. Having large media elements
10. Not including usability testing and accessibility testing

# Resources

1. [Checklist for perfect Dashboard test](https://uxplanet.org/checklist-for-perfect-dashboard-design-771a6f630acd)
2. [Dashboard Testing Fundamentals](https://blogs.sap.com/2016/09/03/let-s-do-qa-dashboard-testing-fundamentals/)
3. [UI Testing Guide](https://www.perfecto.io/blog/ui-testing-comprehensive-guide)
4. [Beginner’s Guide](https://www.testim.io/blog/ui-testing-beginners-guide/)
5. [Eclipsys PDF](https://www.stickyminds.com/sites/default/files/article/file/2014/Graphical%20UI%20Testing%20Checklist.pdf)
6. [IEEE Paper on UI Testing PDF](https://www.cs.umd.edu/~atif/papers/MemonIEEEComputer2002.pdf)
7. [Tutorials Point PDF](https://www.tutorialspoint.com/software_testing/software_testing_tutorial.pdf)
8. [Responsive and Cross-browser Testing](https://www.lambdatest.com/blog/understanding-the-difference-between-cross-browser-testing-responsive-testing/?utm_source=Dzone&utm_medium=blog&utm_campaign=HR-100620-1&utm_term=harish)

Front-end Testing

1. [TestIM](https://www.testim.io/blog/front-end-testing-complete-overview/)
2. [Perfectio.io](https://www.perfecto.io/blog/comprehensive-guide-front-end-testing#:~:text=Front%20end%20testing%20is%20performed%20to%20check%20the,verify%20whether%20input%20fields%20accept%20the%20specified%20characters.)c

Sliders

1. [Designmodo](https://designmodo.com/examples-sliders-web-design/)