Module 4 : System Integration

What is System Integration

- Connecting different sub-systems (components) into a single larger system that functions as one.

- The process of uniting all virtual and physical components into a single cohesive infrastructure to ensure that all the individual pieces of an organization work as a whole.

Why Integrate?

- To improve productivity and quality of organizations operations.

Types of Integration

• Types of Integration

⬡ Legacy system integration

- Integrating outdated legacy systems with newer technology solutions

⬡ Data Integration

- Combining data from different sources into a single, unified view.

⬡ Enterprise application integration (EAI)

- Linking EA applications to simplify and automate business processes without applying excessive application or data structure changes.

⬡ Business-to-business integration

- The automation of business processes and communication between two or more organizations.

⬡ Electronic document interchange

- EDI can be described as the standard electronic format that

companies use to replace their paper-based documents such as invoices or purchase orders.

System Integration Methods

System Integration Methods

POINT-TO-POINT INTEGRATION

- point-to-point integration handles only one function and does not involve any complex business logic.

VERTICAL INTEGRATION

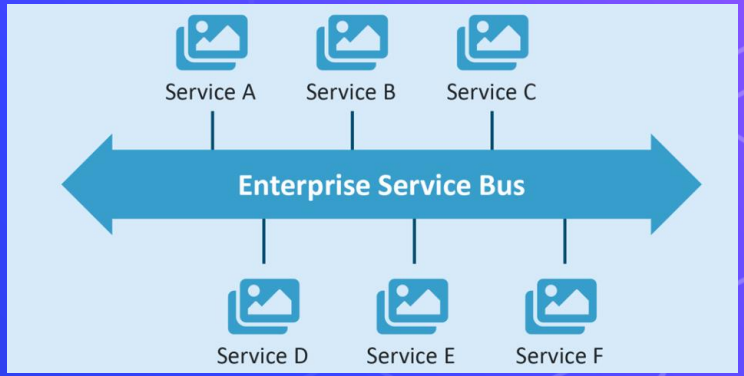
- In vertical integration method, the system components (sub-systems) are integrated by creating functional "silos", beginning with the basic bottom function upward.

STAR INTEGRATION

- Star integration means that a system where each sub-system is connected with other sub-systems using point-to-point connections.

Horizontal Integration

In horizontal integration, a separate sub-system is used as a common interface layer between all sub-systems. This method allows each sub-system to have just one single interface to communicate with all the other sub-systems connected to the common interface layer.



Pros :

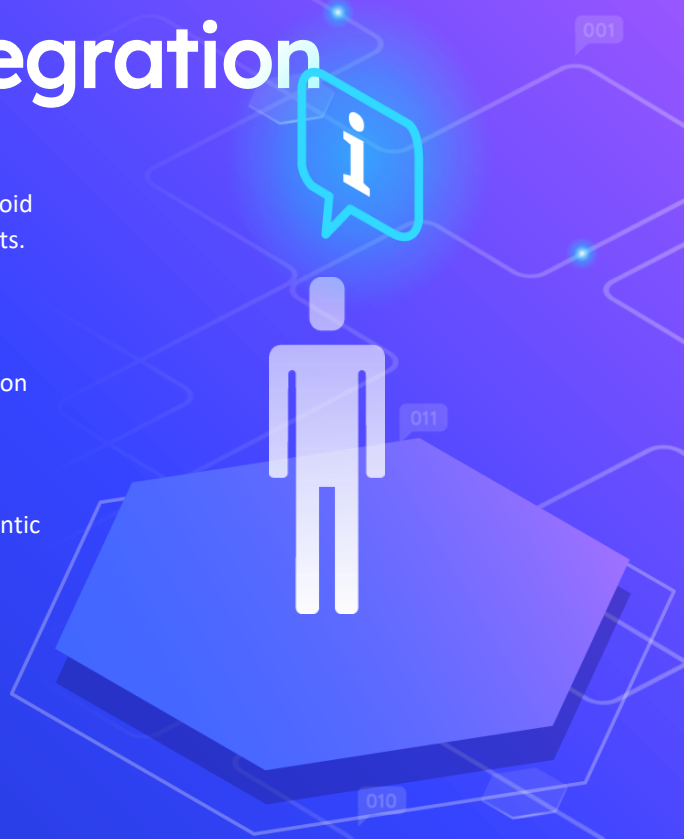
- More Flexible compared to vertical integration
- Fewer changes
- Subsystem Communication

Cons:

- High risk of system error
- Lower communication speed

Common Data Format Integration

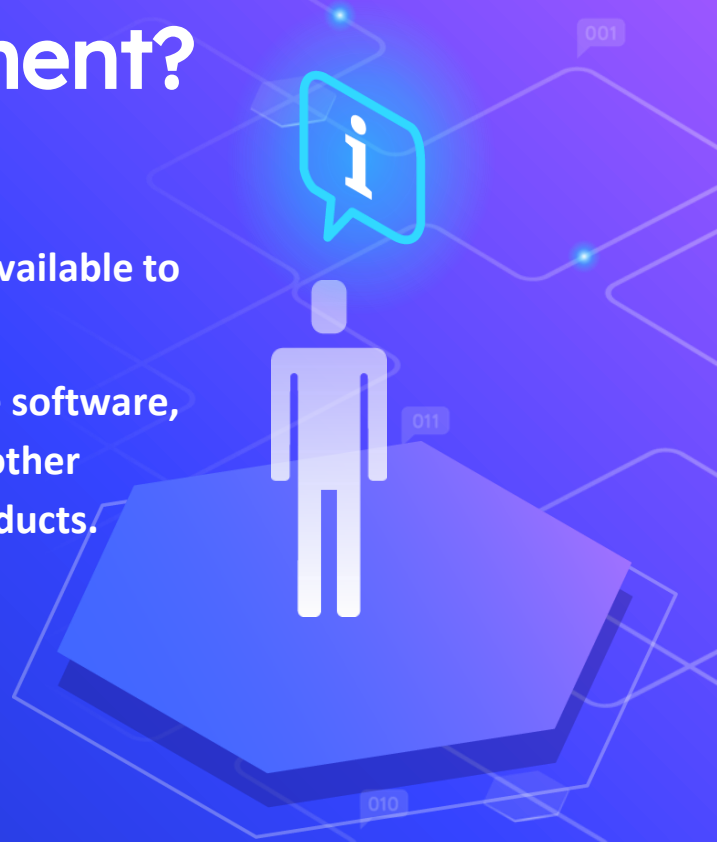
- The simplest explanation is that common data format integration is a method created to avoid having an adapter for every process of converting data to or from other applications' formats.
- Because of that EAI (Enterprise Application Integration) systems usually, contain a data transformation service that enables the conversion between specific and common application formats.
- First, the adapter converts data from the application's format to common and applies semantic transformations to this.
- Moreover, the common data format method allows only one data conversion from native to/from the common format.
- The greatest advantages of this method are automation and seamless data translation.



Lesson 2: Software Deployment

What is Software deployment?

- Software deployment is the process of making software available to be used on a system by users and other programs.
- You might deploy software to create a backup copy of the software, to move the software to another system, or to create another SMP/E-serviceable copy for installing service or other products.

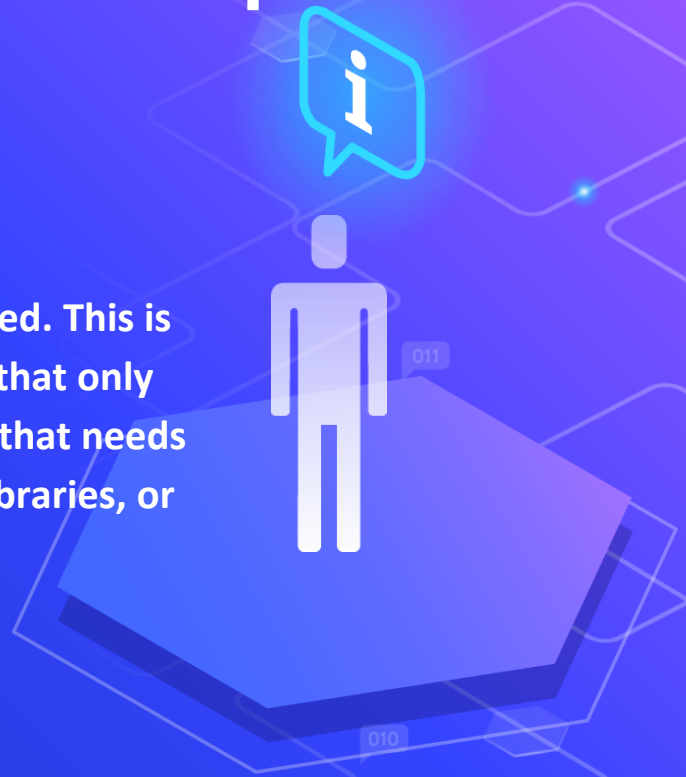


Why is software deployment important?

- Software deployment is one of the most important aspects of the software development process.
- Deployment is the mechanism through which applications, modules, updates, and patches are delivered from developers to users.

Building a software deployment process

- Preparation
  - Begin by gathering the code that needs to be deployed. This is an important step in the process because it ensures that only the correct code is deployed. At this point, the code that needs to be deployed, along with any configuration files, libraries, or resources, can be packaged as a single resource.



Building a software deployment process

- Testing
  - No new software should be deployed unless it has been properly tested. The software must first be deployed to a staging environment where all required tests can be run against a pre-configured set of tests. Developers must then go over the results and fix any bugs that have been discovered.

Building a software deployment process

- **Deployment**
  - This is the stage at which the new code is introduced into the production environment. This procedure is followed exactly as it is with the test environment, so there should be little to no problems at this point.

Software Deployment VS. Software Release

Software Deployment VS Software Release

- Deployment and release are two terms in software engineering that are often used interchangeably. However, they are different! Deployment is a shift of software from one controlled environment to another. On the other hand, releases are a collection of changes for users to experience.

Software Deployment VS Software Release

Release

- A software release is a set of changes to be delivered in the production environment.
- There are frequent releases for updating the changes in production deployments.
- Has higher risks of exposing the users to buggy versions, errors, and issues in the software.
- The release code may not be production-ready
- Software releases are visible to users.

Deployment

- Deployment is a shift of code built from one controlled environment to another
- It is the last phase of SDLC and is executed across domains
- The risk of exposing users to error-prone builds is lower than releases as deployment occurs in a controlled environment.
- Deployment codes are production-ready

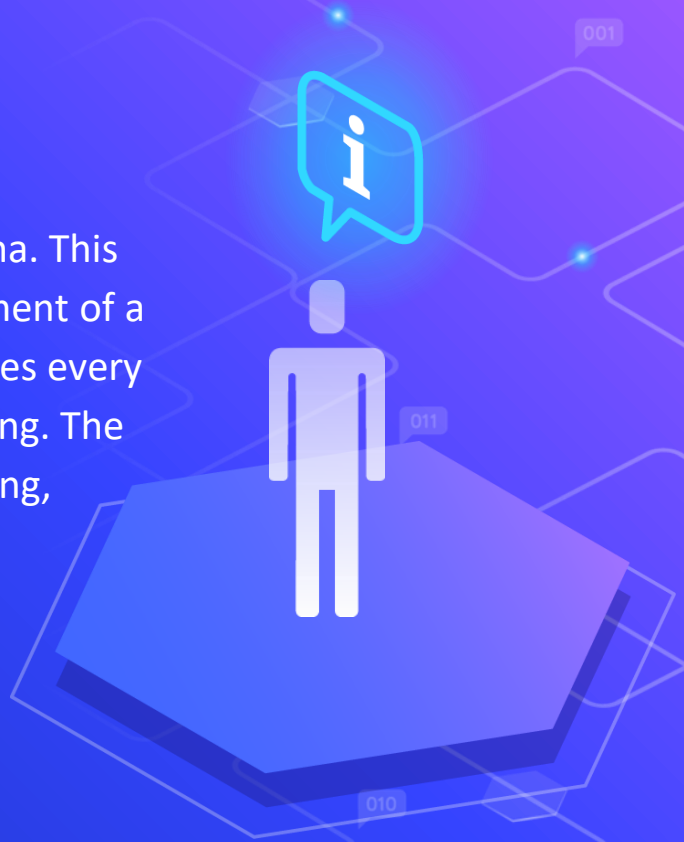
Software Release Process

- Software Release Process/Software Release Cycle is a set of milestones that describe the various stages in a piece of software's life cycle or sequential release timeline, from its conception to its eventual fully-baked release. The length of this life cycle varies based on a variety of factors, such as the type of product, its intended use, and industry security, compliance, and general standards.

The 6 Stages Of Software Release Life Cycle

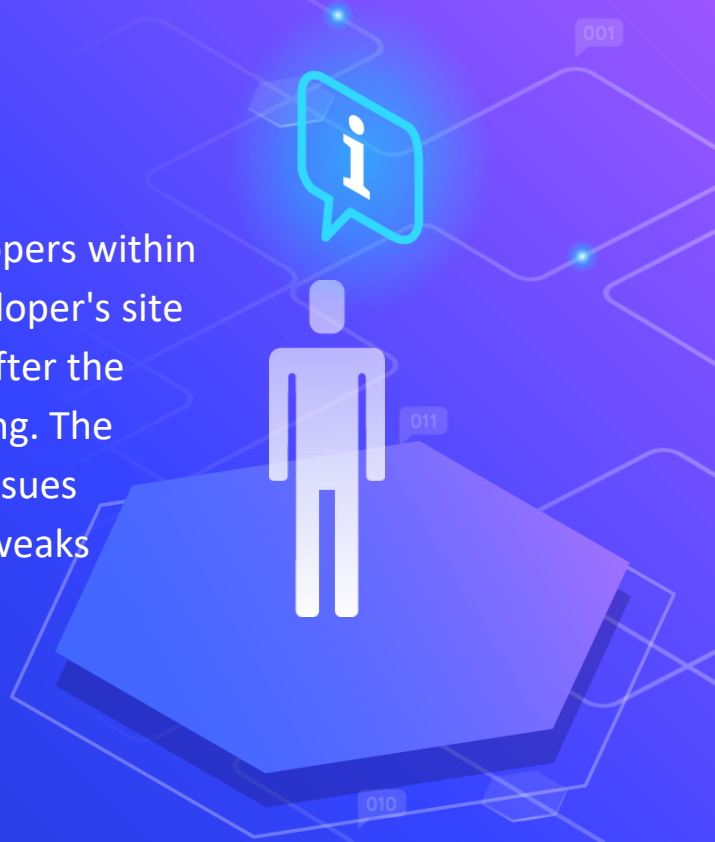
1 . Pre – Alpha Version

- The first stage in the Software Release Life Cycle is Pre-Alpha. This stage of the process is completely focused on the development of a product rather than its marketing and public release. It states every action executed during the initial development before testing. The most common phases of the pre-alpha are analysis, designing, development, and unit testing.



2. Alpha Version

- Alpha testing is performed by internal employees or developers within the organization. This type of test is performed at the developer's site but not at the customer's site. Alpha testing is conducted after the completion of the system testing and before the beta testing. The focus of alpha testing is to improve a product by catching issues before it goes to beta testing and adding any last-minute tweaks suggested from alpha feedback.

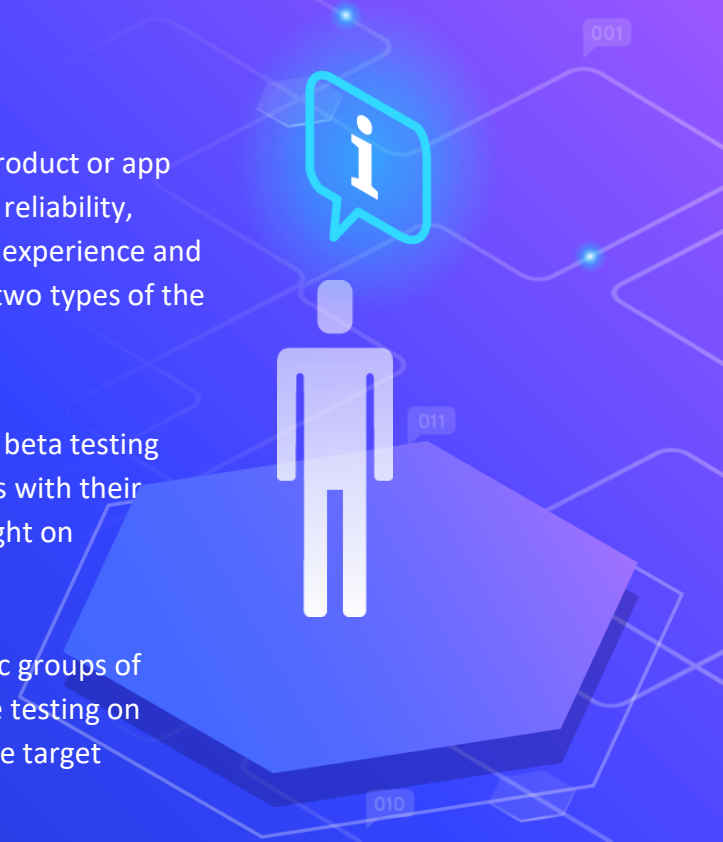


3. Beta Version

In this type of test, customers provide valuable feedback about whether the product or app meets their expectations with respect to functionality, usability, performance, reliability, scalability, and so on. The feedback provided by end-users helps improve user experience and correct operational issues prior to product release into production. There are two types of the beta phrase:

- **Open Beta:** During this phase, anyone who wishes to participate in the beta testing process to do so. This can help developers identify and handle bug fixes with their product quickly and easily, as feedback from multiple users can shed light on problems.

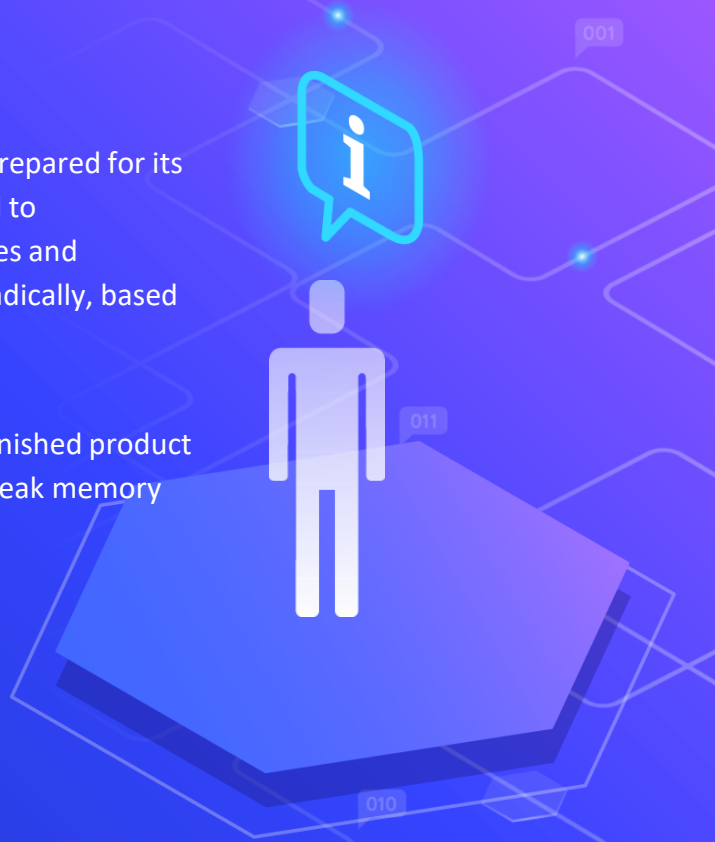
- **Closed Beta:** In this phase, there is a specific target market with specific groups of people that are the testers. The target market helps focus the software testing on specific areas of concern and helps ensure that everything works for the target consumers' needs.



4. Release Candidate

- A release candidate (RC) is a pre-release version of the software that is being prepared for its final product release (in the RC phase) to the public. This is sometimes referred to as ***"Controlled Availability"***. Although it may include all of the intended features and functionality and work as expected, it is still subject to change, possibly even radically, based on any feedback received.

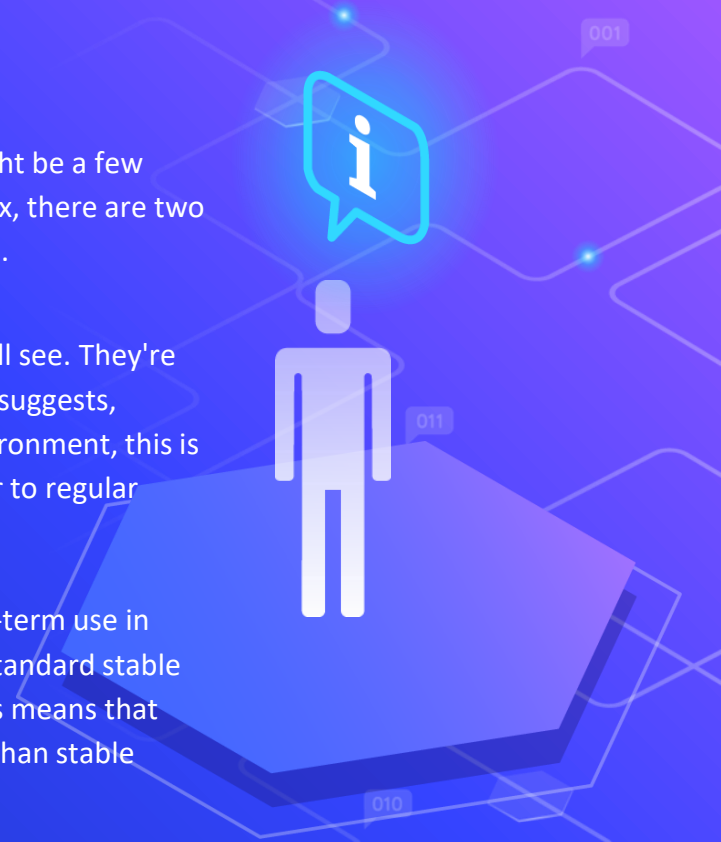
  - Developers may issue several release candidates before releasing the finished product to ensure that the program does not crash under heavy load, does not leak memory significantly, etc.



5. General Availability

- General availability (GA) signifies that a product or service has been made available for purchase by most customers (typically globally), usually via commercial channels. In software engineering, this phrase typically refers to a web application or app that is available for all intended users. During this phase, any updates or further developmental work on the product is aimed at enhancing its features and performance in order to make it more marketable to customers.

6. Production Release



This is considered a complete product by most standards, though there might be a few minor issues with it that are considered acceptable. In some cases, like Linux, there are two types of stable releases: LTS (long-term support) and regular stable releases.

- **Regular Stable releases** - are the most common type of release you'll see. They're easy to install (when we talk about OS software) and, as their name suggests, they're stable. If you're looking to test software in a production environment, this is the type of release you usually use. Web applications often will refer to regular stable releases as “Major” releases.

- **LTS (long-term support) releases** - are specifically designed for long-term use in production settings. These releases have a longer touch-time than standard stable releases (the average time between LTS releases is three years). This means that they have been extensively tested and are considered more secure than stable releases.

Understanding the Deployment Process

Understanding the Deployment Process

- Analyzing business requirements
  - Business goal, consider any business constraints that might impact the ability to achieve the business goal.
- Analyzing technical requirements
  - Design the deployment architecture
- Designing the logical architecture
  - The services required to implement the deployment.
- Designing the deployment architecture
  - Map the logical components specified in the logical architecture to physical components in a deployment architecture.
- Implementing the deployment
  - You work from design documents created during deployment design to build out the deployment architecture and implement the deployment.
- Operating the deployment
  - Continue to monitor, test, and tune the deployment to ensure that it fulfills the business goals

Levels of Software Testing

Four Levels of Software System

- **Unit Testing**
  - Basically done by the developers to make sure that their code is working fine and meet the user specifications.

- **Integration Testing**
  - Combine all of the units within a program and test them as a group.

- **System Testing**
  - System testing is the first level in which the complete application is tested as a whole.

Four Levels of Software System

- **Acceptance Testing**
- **Conducted to determine whether the system is ready for release**

When a program is more thoroughly tested, a greater number of bugs will be detected; this ultimately results in higher quality software.

Usability Testing and Heuristic Evaluation

Usability Testing and Heuristic Evaluation

- **Usability Testing**
  - Usability testing is a method used to evaluate how easy a website is to use. The tests take place with real users to measure how ‘usable’ or ‘intuitive’ a website is and how easy it is for users to reach their goals.

- **Heuristic Evaluation**
  - Heuristic evaluation is a design assessment performed by a UX expert to evaluate a site or product’s design against industry-standard usability principles. These principles were originally established by Jakob Nielsen in his book “Usability Engineering,” and provide a baseline of acceptable standards in human-centered design.

Usability Testing and Heuristic Evaluation

- Neilsen’s Heuristics for Usability

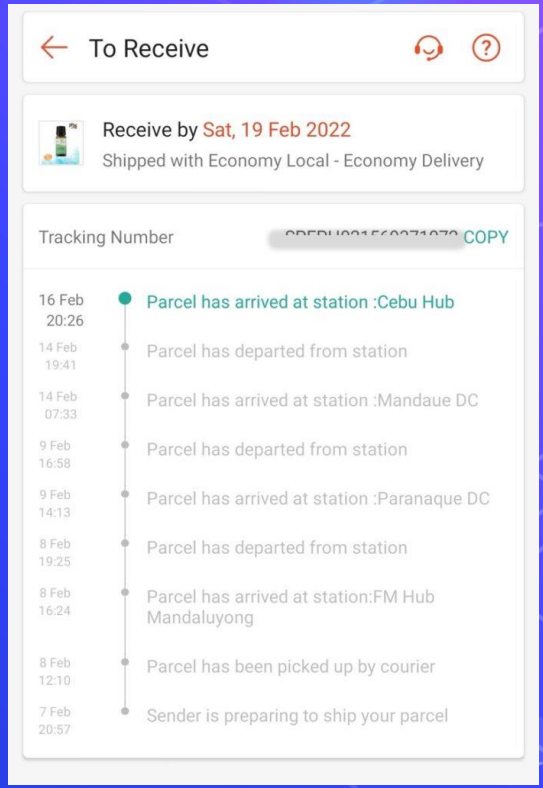
Some of the main distinctions between the two include:

- **Usability testing**
  - Performed by non-professional users with limited experience
  - Has specific tasks to perform
  - Looks at the system screen-by-screen
  - Finds real problems

- **Heuristic evaluation**
  - Performed by UX professionals with experience
  - Has set list of what to look for
  - Looks at the system holistically
  - Finds potential problems

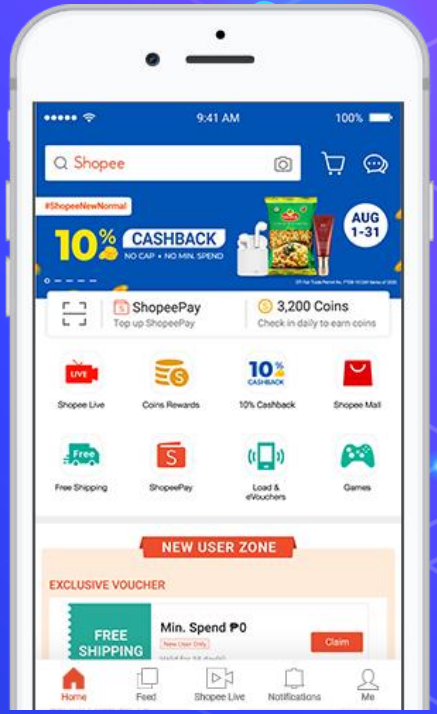
Nielsen's Heuristics for Usability

- **Visibility of System Status**
  - The system should always show the status of an on-going operation to the users until it is done. So the user will get a clear understanding of the progress of that particular process/activity.



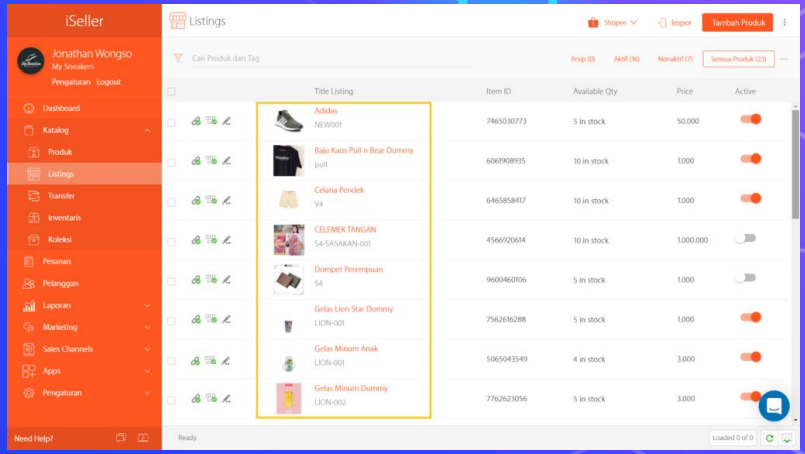
Nielsen’s Heuristics for Usability

- **Match Between System and Real World**
  - Interaction with the user is a key point in product success. To make interaction easier between users and product, try using the components that are familiar to them.



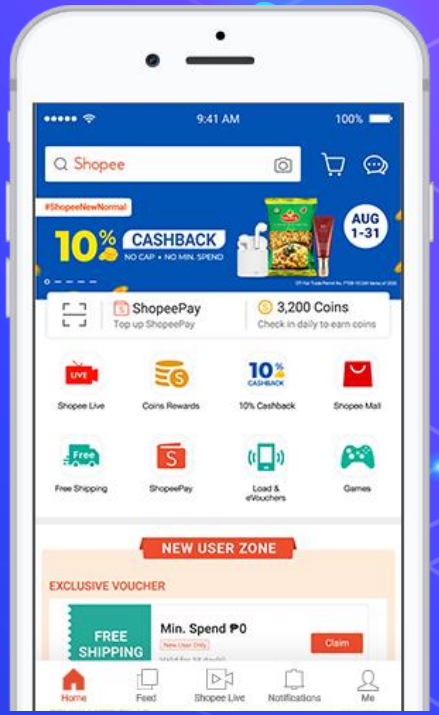
Nielsen’s Heuristics for Usability

- **User Control and freedom**
  - User needs complete control and freedom over the entire system. The system should help them to undo an action that happened by mistake.



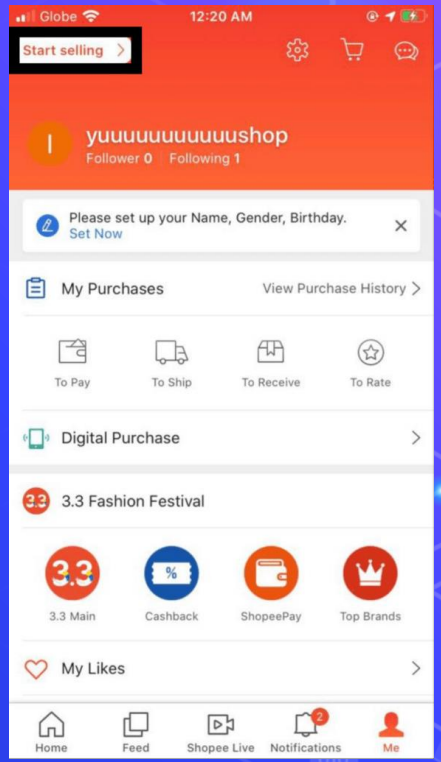
Nielsen’s Heuristics for Usability

- **Consistency and Standard**
  - The basic law of designers’ life. We should follow the consistency and standard throughout product design. Consistency is not only for colors and button styles, but it is also for the overall experience.



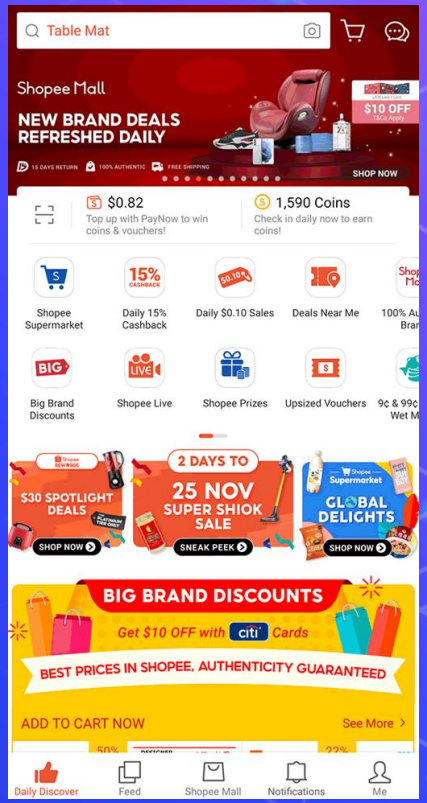
Nielsen's Heuristics for Usability

- **Error Prevention**
  - Users are humans. Humans tend to do mistakes. So, we should always keep an eye to avoid those possible mistakes by giving appropriate suggestions and notifications when needed.



Nielsen’s Heuristics for Usability

- **Recognition Rather than Recall**
  - Try to minimize the use of the user’s memory. Suggest them the options that they might need. Or remind them to complete a certain task that needs to be done soon.



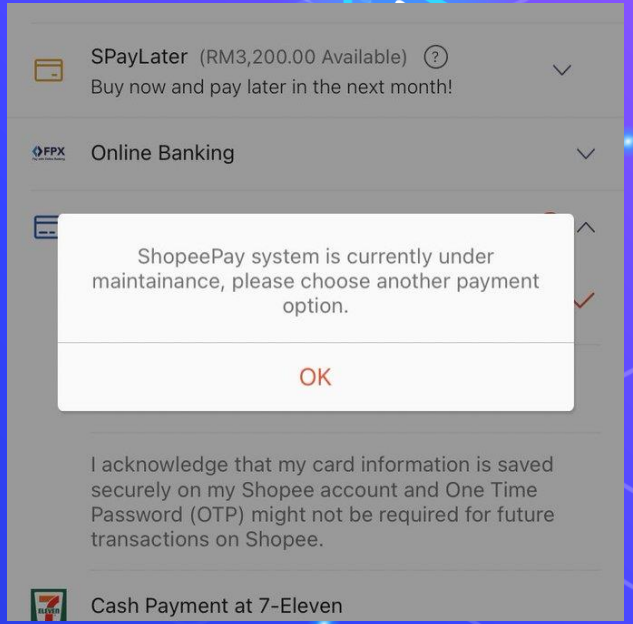
Nielsen’s Heuristics for Usability

- **Flexibility and efficiency of use**
  - The design should be easier to use for all the user groups. Even though we have only one set of a user group, then there would be novice users and experienced ones. We need to satisfy both those categories.

- **Aesthetic and Minimalist design**
  - This sounds familiar as we are following this as a routine. Aesthetic and Minimalist design is not about adding white space. It’s all about giving relevant data and removing all the unwanted things.

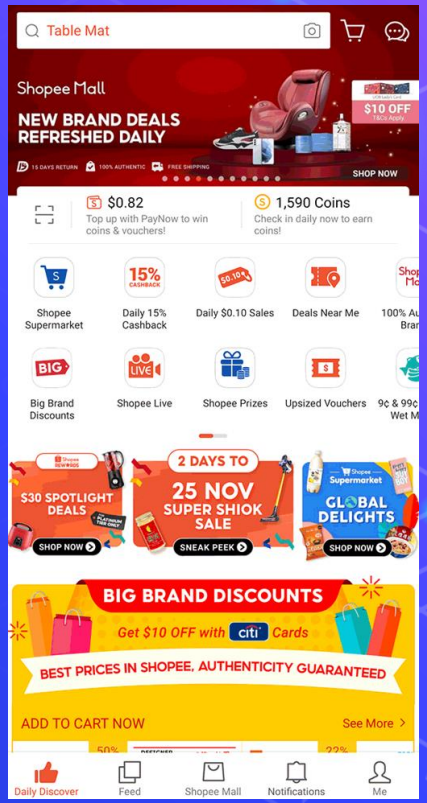
Nielsen’s Heuristics for Usability

- **Help users recognize, diagnose, and recover from errors**
  - Help the users to identify what is the exact error and suggest a way to get rid of that.



Nielsen’s Heuristics for Usability

- **Help and Documentation**
  - The presence of a user in the help page indicates that our product is not that intuitive (in most cases). But if we still think that our design is perfect, then we need to put more attention towards those kinds of users.



Any Questions?