Automated Testing & Testing Tools

- **Automation Testing or Test Automation** is a software testing technique that performs using special automated testing software tools to execute a test case suite.
- Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

- Why Test Automation?

Test Automation is the best way to increase the effectiveness, test coverage, and execution speed in software testing.

Automated software testing is important due to the following reasons:

1. Manual Testing Time and money consuming
2. difficult to test for multilingual sites manually
3. Test Automation does not require Human intervention.
4. Test Automation increases the speed of test execution
5. Automation helps increase Test Coverage
6. Manual Testing can become boring and hence error-prone.

- Benefits of Automation and Tools: The principal attributes of tools and automation are as follows

1. **Speed**
2. **Efficiency**
3. **Accuracy and Precision**
4. **Relentlessness**

- Software test tools aren’t a substitute for software testers—they just help software testers perform their jobs better.

- Test Tools

1. Quick Test Professional- Functional automation
2. Rational Robot- Functional automation
3. Coded UI- Functional automation
4. Selenium- Functional automation
5. Auto IT- Functional automation
6. Load Runner-Non Functional automation
7. Jmeter-Non Functional automation
8. Burp Suite-Non Functional automation

- Which Test Cases to Automate:-Test cases to be automated can be selected using the following criterion.

1. High Risk - Business Critical test cases
2. Test cases that are repeatedly executed
3. Test Cases that are very tedious or difficult to perform manually
4. Test Cases which are time-consuming

- The following category of test cases are not suitable for automation:

1. Test Cases that are newly designed and not executed manually at least once
2. Test Cases for which the requirements are frequently changing
3. Test cases which are executed on an ad-hoc basis.

- Following are benefits of automated testing:

1. 70% faster than the manual testing
2. Wider test coverage of application features
3. Reliable in results
4. Ensure Consistency
5. Saves Time and Cost
6. Improves accuracy
7. Human Intervention is not required while execution
8. Increases Efficiency
9. Better speed in executing tests
10. Re-usable test scripts
11. Test Frequently and thoroughly
12. More cycle of execution can be achieved through automation
13. Early time to market

- How to Choose an Automation Tool?

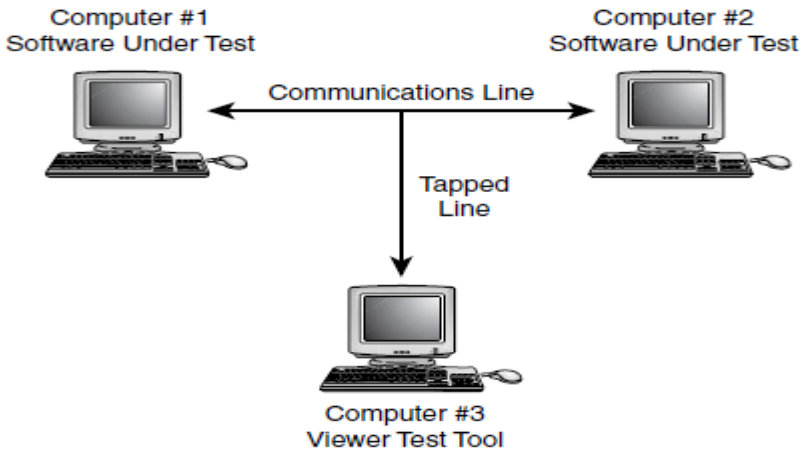
1. Environment Support
2. Ease of use
3. Testing of Database
4. Object identification
5. Image Testing
6. Error Recovery Testing
7. Scripting Language Used
8. Support for multiple testing frameworks
9. Easy to debug the automation software scripts
10. Ability to recognize objects in any environment
11. Extensive test reports and results
12. Minimize training cost of selected tools

- Test Tools
- Two types of tools—*noninvasive* and *invasive*.

1. If a tool is used only to monitor and examine the software without modifying it, it's considered non-invasive.
2. Tool modifies the program code or manipulates the operating environment in any way, it's invasive.

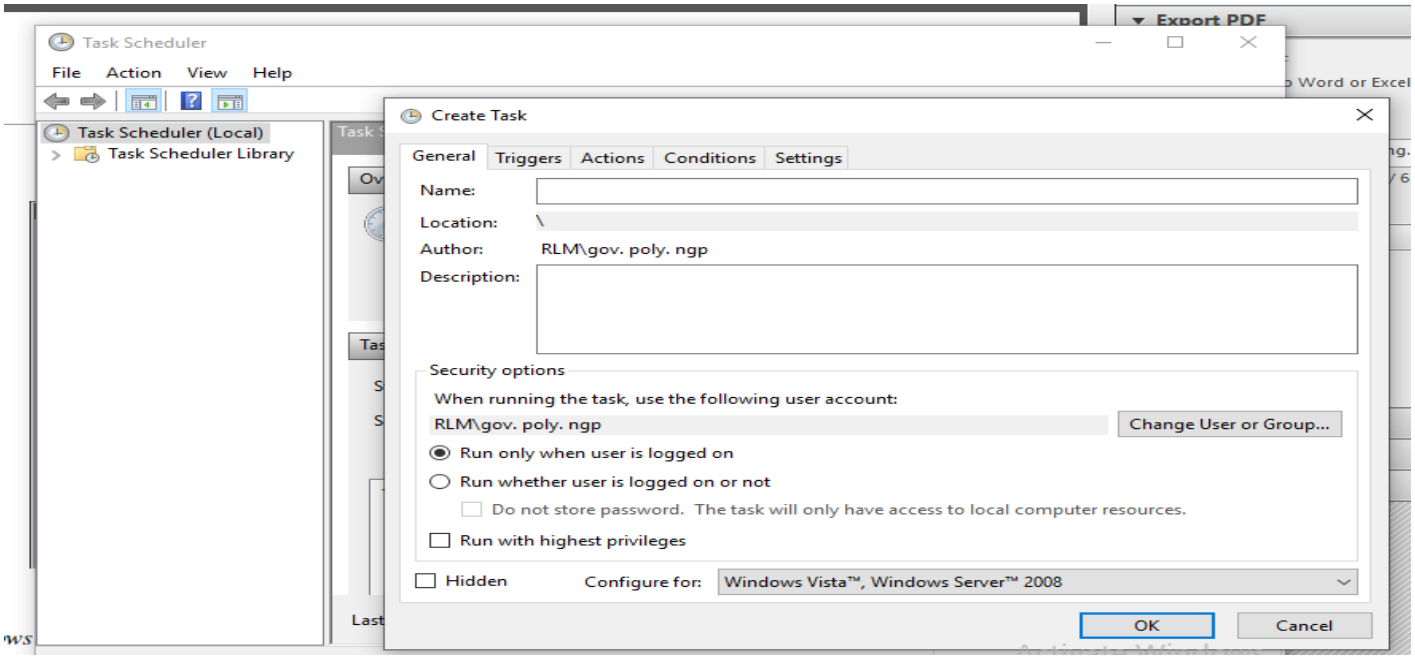
- Viewers and Monitors: A *viewer* or *monitor* test tool allows you to see details of the software's operation that you wouldn't normally be able to see.

1. A code coverage analyzer is an example of a viewing tool.
2. A *communications analyzer:* This tool allows you to see the raw protocol data moving across a network or other communications cable.
3. The code debuggers that come with most compilers



- Drivers: *Drivers* are tools used to control and operate the software being tested.

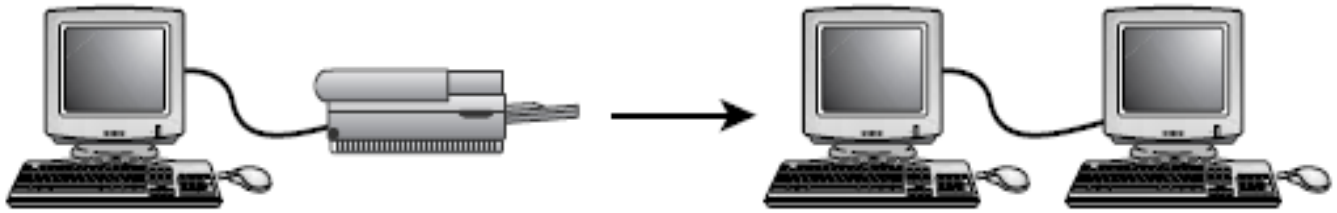
1. One of the simplest examples of a driver is a *batch file*, a simple list of programs or commands that are executed sequentially.
2. Windows task scheduler



- Stubs:

1. Stubs are essentially the opposite of drivers in that they don't control or operate the software being tested; they instead receive or respond to data that the software sends.

Stubs are frequently used when software needs to communicate with external devices.



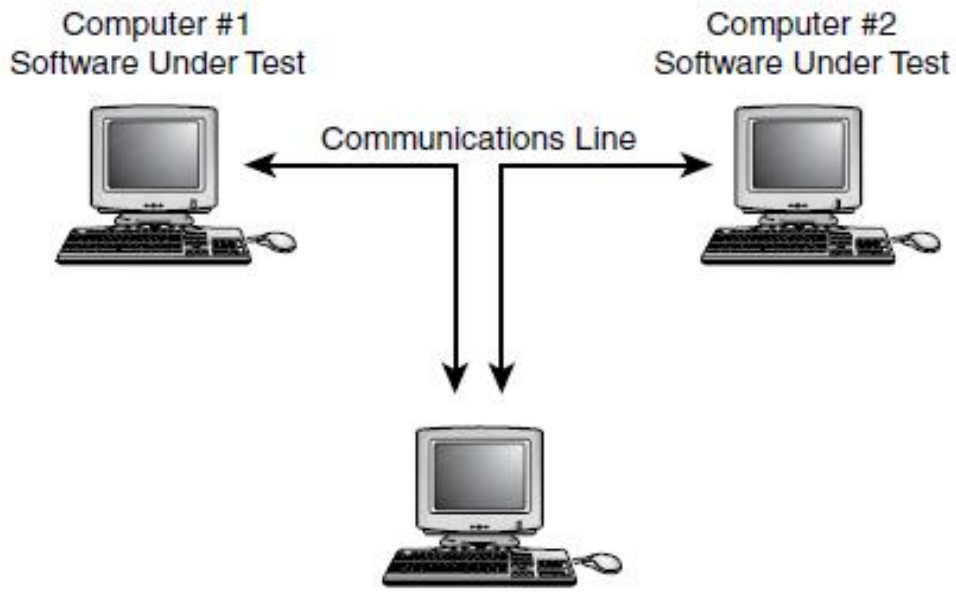
Normal System Configuration

Test Stub Configuration

A computer can act as a stub, replacing a printer and allowing more efficient analysis of the test output.

- Stress and Load Tools: *Stress* and *load* tools induce stresses and loads to the software being tested.
- Microsoft Stress utility: Set the amounts of memory, disk space, files, and other resources available to the software running on the machine.
- *The Microsoft Stress utility allows you to set the system resources available to the software you're testing.*
- Load tools are similar to stress tools in that they create situations for your software that might otherwise be difficult to create.

- Interference Injectors and Noise Generators: *An interference injector hooked into a communications line could test that the software handles error conditions due to noise.*



- Analysis Tools:

1. Word processing software
2. Spreadsheet software
3. Database software
4. File comparison software
5. Screen capture and comparison software
6. Debugger
7. Binary-hex calculator
8. Stopwatch
9. VCR or camera

- **Software Test Automation:**

1. run your test cases
2. look for bugs
3. Analyze what they see
4. log the results

- **Macro Recording and Playback:** *The Macro Setup Wizard allows you to configure how your recorded macros are triggered and played back.*

1. **Name**
2. **Repetitions**
3. **Triggers**
4. **What's captured**
5. **Playback speed**
6. **Playback position**

- **Programmed Macros:**

A Simple Macro That Performs a Test on the Windows Calculator

```
1: Calculator Test #2
2: <<EXECUTE:C:\WINDOWS\Calc.exe~~~~>>
3: <<LOOKFOR:Calculator~~SECS:5~~>>
4: 123-100=
5: <<PROMPT:The answer should be 23>>
6: <<CLOSE:Calculator>>
```

- **Fully Programmable Automated Testing Tools: T**he power of a full-fledged programming language, coupled with macro commands that can drive the software being tested, with the additional capacity to perform verification?

```
FOR i=1 TO 10000
PLAY “Hello World!”
NEXT I
```

- ***Visual Test,*** *originally developed by Microsoft and now sold by Rational Software, is an example of a tool that provides a programming environment, macro commands, and verification capabilities in a single package.*

- **Random Testing:** Monkeys and Gorillas
- Its goal is to simulate what your users might do. That type of automation tool is called a *test monkey*.

- **Dumb Monkeys:** The easiest and most straightforward type of test monkey is a *dumb monkey*. A dumb monkey doesn't know anything about the software being tested; it just clicks or types randomly.

- **Semi-Smart Monkeys:** If you started your monkey running for the night and it found a bug as soon as you walked out the door, you'd lose many hours of valuable test time.

- If you can add programming to your monkey to recognize that a crash has occurred, restart the computer, and start running again, you could potentially find several bugs each night.

- **Smart Monkeys:** A true smart monkey knows

Where he is

What he can do there

Where he can go

Where he's been

If what he's seeing is correct

- examine data as it goes, checking the results of its actions and looking for differences from what it expects. If you programmed in your test cases, the smart monkey could randomly execute them, look for bugs, and log the results.

- Realities of Using Test Tools and Automation:

1. The software changes. Specifications are never fixed. New features are added late.
2. There's no substitute for the human eye and intuition.
3. Verification is hard to do.
4. It's not reliable to rely on automation too much
5. Don't spend so much time working on tools and automation that you fail to test the software.
6. If you're writing macros, developing a tool, or programming a monkey, you're doing development work. You should follow the same standards and guidelines that you ask of your programmers.
7. Some tools are invasive and can cause the software being tested to improperly fail.

- **Bug Bashes:** A **bug bash** is a procedure where all the developers, testers, program managers, usability researchers, designers, documentation folks, and even sometimes marketing people, put aside their regular day-to-day duties and "pound on the product“

- That is, each exercises the product in every way they can think of. Because each person will use the product in slightly different (or very different) ways, and the product is getting a great deal of use in a short amount of time, this approach may reveal bugs relatively quickly.

- You're likely under a tight schedule, you find as many bugs as possible in the time you have, but someone else can come in, test the same code, and find additional bugs.

1. Having another set of eyes look at the software helps break the pesticide paradox.
2. Similarly, people don't just see differently from each other, they go about their testing differently too.
3. Having someone assist you in your testing helps eliminate boredom.
4. Watching how someone else approaches a problem is a great way to learn new testing techniques.

- **Test Sharing:** One common approach is to simply swap test responsibilities with another tester for a few hours or a few days.

- Think of it as “You run my tests and I’ll run yours.”

- You’ll both gain an independent look at the software while still having the basic testing tasks completed.

- A fun way to share the testing tasks is to schedule a *bug bash*.

- **Beta Testing** :Beta testing is the term used to describe the external testing process in which the software is sent out to a select group of potential customers who use it in a real-world environment.

- It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer's environment

- Another common method for having others verify and validate the software is a process called *beta testing*.

- Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

- Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

- Several things to think about when planning for or relying on a beta test

1. Who are the beta testers?
2. Similarly, how will you know if the beta testers even use the software?
3. Beta tests can be a good way to find compatibility and configuration bugs.
4. Usability testing is another area that beta testing can contribute to if the participants are well chosen—a good mix of experienced and inexperienced users.
5. Besides configuration, compatibility, and usability, beta tests are surprisingly poor ways to find bugs.
6. beta test program can take up a lot of a tester's time.

- Outsourcing Your Testing: A common practice in many corporations is to outsource or subcontract a portion of the test work to other companies that specialize in various aspects of software testing.

- Configuration and compatibility testing are typically good choices for outsourcing. They usually require a large test lab containing many different hardware and software combinations and a staff of several people to manage it.

- Small software companies can’t afford the overhead and expense for maintaining these test labs, so it makes more sense for them to outsource this testing to companies who make it their business to perform configuration and compatibility tests.

1. What exactly are the testing tasks that the testing company is to perform?
2. What schedule will they follow? Who will set the schedule?
3. What deliverables are you to provide to the testing company?
4. What deliverables are they to provide to you?
5. How will you communicate with them?
6. How will you know if the testing company is meeting your expectations?

- **Types of test Tools:** Tools from a software testing context can be defined as a product that supports one or more test activities right from planning, requirements, creating a build, test execution, defect logging and test analysis.

- Classification of Tools

Tools can be classified based on several parameters. They include:

1. The purpose of the tool
2. The Activities that are supported within the tool
3. The Type/level of testing it supports
4. The Kind of licensing (open source, freeware, commercial)
5. The technology used

1. Test Management Tool: Test Managing, scheduling, defect logging, tracking and analysis.

- Used by Tester

2. Configuration management tool:For Implementation, execution, tracking changes.

- Used by- All Team members

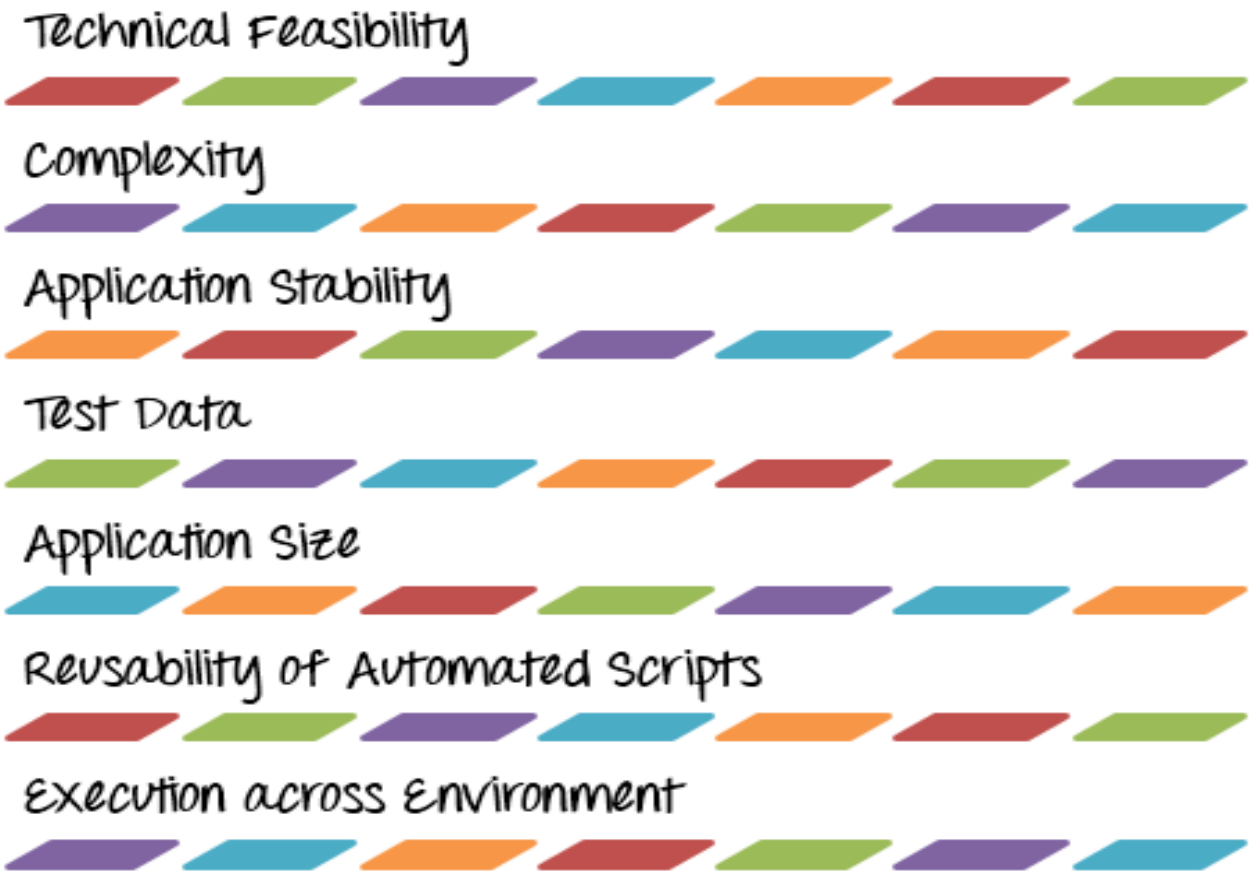
3. Static Analysis Tools: Static Testing

- Used by Developers

4. Test data Preparation Tools: Analysis and Design, Test data generation

- Used by Testers

- **Selection and Introduction of Test Tools:**



- **Tool selection process:**

1. **Identify the requirement for tools**
2. **Evaluate the tools and vendors**
3. **Estimate cost and benefit**
4. **Make the final decision**

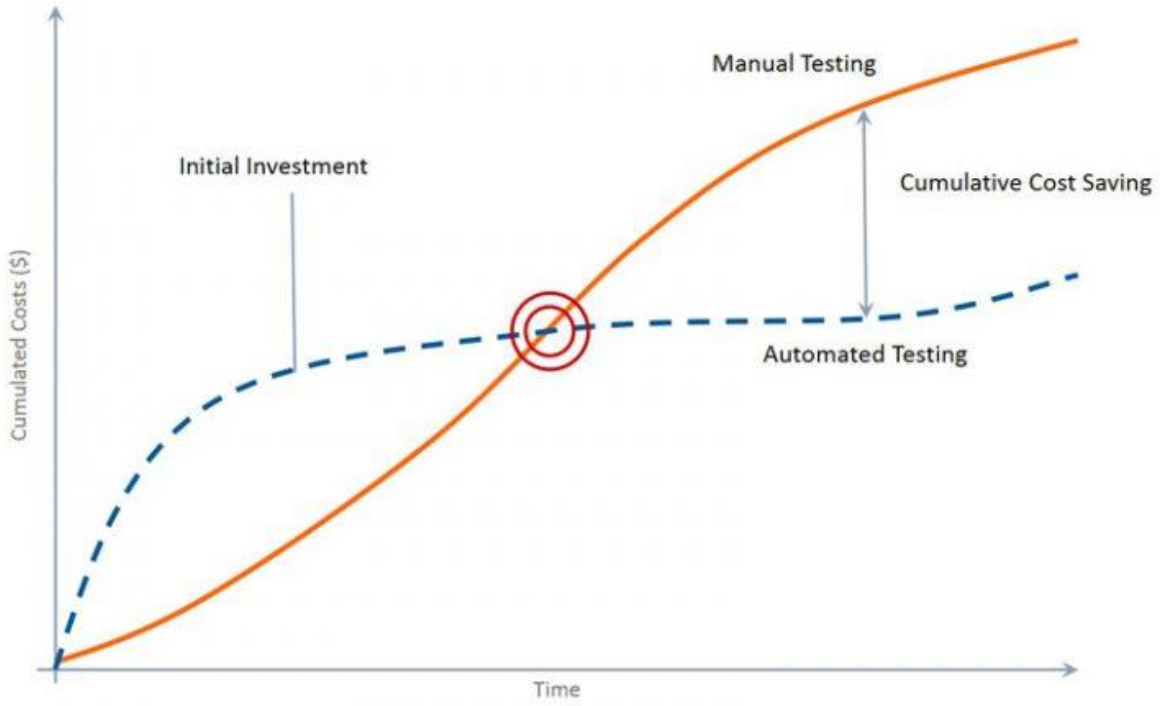
- **Test Automation Tool Evaluation Criteria:**

1. Do you have the necessary skilled resource to allocate for automation tasks?
2. What is your budget?
3. Does the tool satisfy your testing needs?
4. Does the tool provide you the free trial version so that you can evaluate it before making a decision?
5. is the current tool version stable?
6. Do you want automation tool for only your project needs or you are looking for a common tool for all projects in your company?
7. Which testing types does it support?
8. Does the tool support easy interface to create and maintain test scripts?
9. Does it provide simple interface yet powerful features to accomplish complex tasks?
10. how easy is it to provide input test data for complex or load tests?
11. Does it provide the powerful reporting with graphical interface?
12. Is the vendor providing initial training?

- The importance of cost-effective software testing is as follows.

1. For Validation and Verification
2. Prove Operability and Usability
3. Improve software quality
4. Avoid operational problems
5. Maintain a good customer image
6. Avoid legal problems
7. Decrease cost of fixing bugs by 5x

Manual vs Automated Testing



Not everything can be automated, and there's no need to try to replace all manual testing because there are things that just can't be automated and are not worth automating. Testing a new functionality manually allows you to quickly know more about the application at a low cost. As knowledge accumulates, the inventory of tests increases, and consequently, the cost also increases of manual testing. On the other hand, automation has a higher initial cost which decreases as it progresses.