

TESTING ENGINEERING FUNDAMENTALS

Software Engineering Seminar

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Full-time Adjunct Professor
Computer Engineering Program
School of Engineering
Universidad Distrital Francisco José de Caldas

2025-II



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline

1 Basic Concepts

2 Testing Levels



Outline

1 Basic Concepts

2 Testing Levels



Software Engineering

- **Software engineering** is the application of engineering principles to the design, development, and maintenance of software systems.
- It involves the use of systematic methods and tools to ensure that software is of high quality, reliable, and meets the needs of its users.
- The main goal of software engineering is to produce software that is cost-effective, efficient, and maintainable.
- It encompasses a wide range of activities, including requirements analysis, design, implementation, testing and maintenance.
- Software engineering is a collaborative discipline that involves collaboration between developers.



Software Testing

- **Software testing** is the process of **evaluating** a software **system** or its **components** to determine whether it **satisfies** the specified **requirements** and to **identify any defects**.

[Functional

non-functional \Rightarrow quality

Fix before user detect them



Software Testing

- **Software testing** is the process of **evaluating** a software system or its components to determine whether it **satisfies** the specified **requirements** and to **identify any defects**.
- It involves **executing the software under controlled conditions** and evaluating the **results against expected outcomes**.
- The *main goal of software testing* is to ensure that the software is of high quality, reliable, and meets the needs of its users.
- It is an **essential** part of the **software development process** and is typically performed by independent testers or quality assurance teams.

test
design

Users
Infrastructure
resources



Software Testing

- **Software testing** is the process of **evaluating** a software system or its components to determine whether it **satisfies** the specified **requirements** and to **identify any defects**.
- It involves executing the software under **controlled conditions** and evaluating the **results against expected outcomes**.
- The main goal of software testing is to ensure that the software is of high quality, reliable, and meets the needs of its users. Q.A.
- It is an essential part of the software development process and is typically performed by independent testers or quality assurance teams.

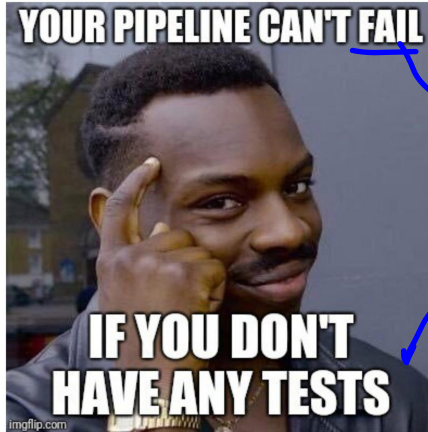
Software developers
↓
Unit test



Classical Philosophy

"If you don't have time to do it right, when will you have time to do it over?"

↳ project management



Software Quality

- **Software quality** is the degree to which a software system meets the specified requirements and satisfies the needs of its users.
- It is a critical aspect of software development and is typically measured by a set of **quality attributes**, such as **reliability**, **performance**, **usability** and **maintainability**.
- **Software quality** is typically assessed through a combination of **static analysis**, **dynamic analysis**, and **testing** techniques.



Quality is a Key Factor

**“No pasa
nada, así
mándalo a
producción”
by
Crowdstrike**

Figure: Microsoft Style!



Testing Engineering

- **Testing engineering** is the application of engineering principles to the **design**, **development**, and **execution** of **tests** for software systems.
- It involves the use of **systematic methods** and **tools** to ensure that software is of **high quality**, **reliable**, and **meets the needs of its users**.
- The main goal of testing engineering is to produce software that is **cost-effective**, **efficient**, and **maintainable** through the use of **automated testing** and **test-driven development** (TDD) practices.
- It encompasses a wide range of activities, including **test design**, **test execution**, and **test analysis**.
- **Testing engineering** is a *collaborative* discipline that involves *collaboration* between **developers** and **testers** to ensure that software is of **high quality** and **meets the needs of its users**.



Testing Engineering

- **Testing engineering** is the application of engineering principles to the **design**, **development**, and **execution** of **tests** for software systems.
- It involves the use of **systematic methods** and **tools** to ensure that software is of **high quality**, **reliable**, and **meets the needs of its users**.
- The main goal of testing engineering is to produce software that is **cost-effective**, **efficient**, and **maintainable** through the use of **automated testing** and **test-driven development** (TDD) practices.
- It encompasses a wide range of activities, including **test design**, **test execution**, and **test analysis**.
- **Testing engineering** is a *collaborative* discipline that involves *collaboration* between **developers** and **testers** to ensure that software is of **high quality** and **meets the needs of its users**.



Testing Engineering

- **Testing engineering** is the application of engineering principles to the **design**, **development**, and **execution** of **tests** for software systems.
- It involves the use of **systematic methods** and **tools** to ensure that software is of **high quality**, **reliable**, and **meets the needs of its users**.
- The main goal of testing engineering is to produce software that is **cost-effective**, **efficient**, and **maintainable** through the use of **automated testing** and **test-driven development** (TDD) practices.
- It encompasses a wide range of activities, including **test design**, **test execution**, and **test analysis**.
- **Testing engineering** is a *collaborative* discipline that involves *collaboration* between **developers** and **testers** to ensure that software is of **high quality** and **meets the needs of its users**.



Test-Driven Development (TDD)

- **Test-Driven Development (TDD)** is a *software development methodology* that emphasizes the use of **automated tests** to drive the design and development of software.
- It involves **writing a test** for a specific piece of functionality **before** writing the code to implement that functionality.
- The main goal of TDD is to produce software that is of **high quality**, **reliable**, and **meets the needs of its users** through the use of **automated testing** and **test-driven development** practices.
- TDD is typically used in *conjunction* with other software development methodologies, such as *Agile* or *Scrum*, to ensure that software is developed in a **collaborative** and **iterative** manner.



Test-Driven Development (TDD)

- **Test-Driven Development** (TDD) is a *software development methodology* that emphasizes the use of **automated tests** to drive the design and development of software.
- It involves **writing a test** for a specific piece of functionality **before** writing the code to implement that functionality.
- The main goal of TDD is to produce software that is of **high quality**, **reliable**, and **meets the needs of its users** through the use of **automated testing** and **test-driven development** practices.
- TDD is typically used in *conjunction* with other software development methodologies, such as *Agile* or *Scrum*, to ensure that software is developed in a **collaborative** and **iterative** manner.



Quality Guidelines

- **Quality guidelines** are *principles* that *guide* the **design** of a **system** to *ensure* that it **meets** the **needs** of its **users**.
- They include **reliability**, **scalability**, **maintainability**, and **usability guidelines**.
- They are *important* for *ensuring* that a **system** is **robust**, **efficient**, and **effective**.



Reliability Guidelines

- **Reliability guidelines** are *principles* that *guide* the **design** of a **system** to *ensure* that it is **reliable** and **dependable**.
- They include **fault-tolerance**, **redundancy**, and **error-handling guidelines**.
- They are *important* for *ensuring* that a **system** is **robust** and **resilient** to **failures**.



Scalability Guidelines

- **Scalability guidelines** are *principles* that *guide* the **design** of a **system** to *ensure* that it is **scalable** and **flexible**.
- They include **modularity**, **extensibility**, and **performance guidelines**.
- They are *important* for *ensuring* that a **system** can **grow** and **adapt** to **changing requirements**.



Maintainability Guidelines

- **Maintainability guidelines** are *principles* that *guide* the **design** of a **system** to *ensure* that it is **easy** to **maintain** and **update**.
- They include **modularity**, **documentation**, and **versioning guidelines**.
- They are *important* for *ensuring* that a **system** can be **easily maintained** and **updated** by its **developers**.



Quality Standards

- **Quality standards** are **benchmarks** that *define* the **level** of **quality** that a **system** must **meet**.
- They include **ISO 9000**, **CMMI**, and **Six Sigma standards**.



ISO 9000

- **ISO 9000** is a quality standard that *defines* the **requirements** for a **quality management system**.
- It is *designed* to help **organizations** ensure that they **meet** the **needs** of their **customers** and **stakeholders**.
- It is *based* on a number of quality management principles, including **customer focus**, **leadership**, and **continuous improvement**.



ISO 27001

- **ISO 27001** is a quality standard that *defines* the **requirements** for an **information security management system**.
- It is *designed* to help **organizations** protect their **information** and **ensure** that it is **secure** and **confidential**.
- It is *based* on a number of information security management principles, including **risk assessment**, **security policies**, and **incident response**.



CMMI

- **CMMI** is a quality standard that *defines* the **requirements** for a mature software development process.
- It is *designed* to help organizations improve their software development processes and deliver high-quality products to their customers.
- It is *based* on a number of best practices for software development, including requirements management, project planning, and process monitoring.



Six Sigma

- **Six Sigma** is a quality standard that *defines* the **requirements** for a **process** that is *capable* of **producing high-quality products**.
- It is *designed* to help **organizations** improve their **processes** and **reduce defects** in their **products** and **services**.
- It is *based* on a number of quality management principles, including **data-driven decision-making**, **process improvement**, and **customer focus**.



Outline

1 Basic Concepts

2 Testing Levels



Testing Levels

- **Testing levels** refer to the **different stages** of testing that a software system goes through during its **development lifecycle**.
- **Each level** of testing has its own set of **objectives**, **techniques**, and **tools**.
- The **main goal** of testing levels is to **ensure** that the software is of high quality, reliable, and meets the needs of its users.
- The most **common testing levels** include **unit testing**, **integration testing**, **system testing**, and **acceptance testing**.



Testing Levels

- **Testing levels** refer to the **different stages** of testing that a software system goes through during its **development lifecycle**.
- **Each level** of testing has its own set of **objectives**, **techniques**, and **tools**.
- The **main goal** of testing levels is to **ensure** that the software is of high quality, reliable, and meets the needs of its users.
- The most **common testing levels** include **unit testing**, **integration testing**, **system testing**, and **acceptance testing**.



Base Code Quality & Good Practices

- **Base code quality** refers to the **fundamental principles** and **practices** that ensure that the **code** is of high quality, maintainable, and efficient.
- It includes **practices** such as **code reviews**, **code refactoring**, and **code documentation**.
- **Good practices** in software development include following **coding standards**, using **version control systems**, and writing **unit tests**. These practices help to ensure that the **code** is of high quality, maintainable, and efficient, and that it meets the needs of its users.

Demo Time!



Base Code Quality & Good Practices

- **Base code quality** refers to the **fundamental principles** and **practices** that ensure that the **code** is of high quality, maintainable, and efficient.
- It includes **practices** such as **code reviews**, **code refactoring**, and **code documentation**.
- **Good practices** in software development include following **coding standards**, using **version control systems**, and writing **unit tests**. These practices help to ensure that the **code** is of high quality, maintainable, and efficient, and that it meets the needs of its users.

Demo Time!



Static Analysis

- **Static analysis** is a *software testing technique* that involves **analyzing the source code** of a software system **without executing it**.
- The **main goal** of static analysis is to identify **potential defects**, **vulnerabilities**, and **code quality issues** in the source code.
- **Static analysis tools** can automatically analyze the source code and provide feedback on potential issues, such as **coding standards violations**, **security vulnerabilities**, and **performance issues**.
- *Static analysis* is typically performed **early** in the software development process.

Demo Time!



Static Analysis

- **Static analysis** is a *software testing technique* that involves **analyzing the source code** of a software system **without executing it**.
- The **main goal** of static analysis is to identify **potential defects**, **vulnerabilities**, and **code quality issues** in the source code.
- **Static analysis tools** can automatically analyze the source code and provide feedback on potential issues, such as **coding standards violations**, **security vulnerabilities**, and **performance issues**.
- *Static analysis* is typically performed **early** in the software development process.

Demo Time!



Unit Testing

- **Unit testing** is a *software testing technique* that involves **testing individual units or components** of a software system in isolation.
- The **main goal** of unit testing is to identify **defects** and **issues** in the source code at an early stage of the software development process.
- **Unit tests** are typically written by **developers** and are executed **automatically** as part of the software development process.
- **Unit testing** is typically performed **after the code has been written** and **before the code is integrated** into the larger software system.
- **Unit testing** is an **essential** part of the software development process and is typically performed using **unit testing frameworks** such as JUnit, NUnit, or pytest.

Demo Time!



Unit Testing

- **Unit testing** is a *software testing technique* that involves **testing individual units or components** of a software system in isolation.
- The **main goal** of unit testing is to identify **defects** and **issues** in the source code at an early stage of the software development process.
- **Unit tests** are typically written by **developers** and are executed **automatically** as part of the software development process.
- **Unit testing** is typically performed **after the code has been written** and **before the code is integrated** into the larger software system.
- **Unit testing** is an **essential** part of the software development process and is typically performed using **unit testing frameworks** such as JUnit, NUnit, or pytest.

Demo Time!



Unit Testing

- **Unit testing** is a *software testing technique* that involves **testing individual units or components** of a software system in isolation.
- The **main goal** of unit testing is to identify **defects** and **issues** in the source code at an early stage of the software development process.
- **Unit tests** are typically written by **developers** and are executed **automatically** as part of the software development process.
- **Unit testing** is typically performed **after the code has been written** and **before the code is integrated** into the larger software system.
- **Unit testing** is an **essential** part of the software development process and is typically performed using **unit testing frameworks** such as JUnit, NUnit, or pytest.

Demo Time!



Integration Testing

- **Integration testing** is a *software testing technique* that involves testing the interactions between different components or units of a software system.
- The **main goal** of integration testing is to identify defects and issues in the interactions between different components or units of the software system.
- **Integration testing** is typically performed after unit testing and before system testing.
- It can be performed using a variety of techniques, including top-down, bottom-up, and big bang integration testing.
- **Integration testing** is an essential part of the software development process and is typically performed using integration testing frameworks such as TestNG, Mocha, or Cucumber.



Integration Testing

- **Integration testing** is a *software testing technique* that involves testing the interactions between different components or units of a software system.
- The **main goal** of integration testing is to identify defects and issues in the interactions between different components or units of the software system.
- **Integration testing** is typically performed after unit testing and before system testing.
- It can be performed using a variety of techniques, including top-down, bottom-up, and big bang integration testing.
- Integration testing is an essential part of the software development process and is typically performed using integration testing frameworks such as TestNG, Mocha, or Cucumber.



Integration Testing

- **Integration testing** is a *software testing technique* that involves testing the interactions between different components or units of a software system.
- The **main goal** of integration testing is to identify defects and issues in the interactions between different components or units of the software system.
- **Integration testing** is typically performed after unit testing and before system testing.
- It can be performed using a variety of techniques, including top-down, bottom-up, and big bang integration testing.
- **Integration testing** is an essential part of the software development process and is typically performed using integration testing frameworks such as TestNG, Mocha, or Cucumber.



Integration Testing Techniques

- **Top-down integration testing** is a technique that involves testing the **higher-level components** of a software system first, and then gradually integrating and testing the **lower-level components**.
- **Bottom-up integration testing** is a technique that involves testing the **lower-level components** of a software system first, and then gradually integrating and testing the **higher-level components**.
- **Big bang integration testing** is a technique that involves integrating and testing **all components** of a software system at once, without any prior integration testing.



Integration Testing Techniques

- **Top-down integration testing** is a technique that involves testing the **higher-level components** of a software system first, and then gradually integrating and testing the **lower-level components**.
- **Bottom-up integration testing** is a technique that involves testing the **lower-level components** of a software system first, and then gradually integrating and testing the **higher-level components**.
- **Big bang integration testing** is a technique that involves integrating and testing **all components** of a software system at once, without any prior integration testing.



Integration Testing Techniques

- **Top-down integration testing** is a technique that involves testing the **higher-level components** of a software system first, and then gradually integrating and testing the **lower-level components**.
- **Bottom-up integration testing** is a technique that involves testing the **lower-level components** of a software system first, and then gradually integrating and testing the **higher-level components**.
- **Big bang integration testing** is a technique that involves integrating and testing **all components** of a software **system at once**, without any prior integration testing.



API Testing

- **API testing** is a *software testing technique* that involves **testing the application programming interfaces (APIs)** of a software system.
- The **main goal** of API testing is to identify **defects** and **issues** in the APIs of the software system, including their **functionality**, **performance**, and **usability**.
- **API testing** is typically performed after **integration testing** and before **system testing**.
- It can be performed using a variety of **techniques**, including **functional**, **non-functional**, and **regression** testing.
- **API testing** is an **essential** part of the software development process and is typically performed using **API testing frameworks** such as **Postman**, **SoapUI**, or **RestAssured**.

Demo Time!



API Testing

- **API testing** is a *software testing technique* that involves **testing the application programming interfaces (APIs)** of a software system.
- The **main goal** of API testing is to identify **defects** and **issues** in the APIs of the software system, including their **functionality**, **performance**, and **usability**.
- **API testing** is typically performed after **integration testing** and before **system testing**.
- It can be performed using a variety of **techniques**, including **functional**, **non-functional**, and **regression** testing.
- **API testing** is an **essential** part of the software development process and is typically performed using **API testing frameworks** such as **Postman**, **SoapUI**, or **RestAssured**.

Demo Time!



API Testing

- **API testing** is a *software testing technique* that involves **testing the application programming interfaces (APIs)** of a software system.
- The **main goal** of API testing is to identify **defects** and **issues** in the APIs of the software system, including their **functionality**, **performance**, and **usability**.
- **API testing** is typically performed after **integration testing** and before **system testing**.
- It can be performed using a variety of **techniques**, including **functional**, **non-functional**, and **regression** testing.
- **API testing** is an **essential** part of the software development process and is typically performed using **API testing frameworks** such as **Postman**, **SoapUI**, or **RestAssured**.

Demo Time!



System Testing

- **System testing** is a *software testing technique* that involves **testing the entire software system as a whole**.
- The **main goal** of system testing is to identify **defects** and **issues** in the software system as a whole, including its **functionality**, **performance**, and **usability**.
- **System testing** is typically performed after **integration testing** and before **acceptance testing**.
- It can be performed using a variety of **techniques**, including **functional**, **non-functional**, and **regression** testing.
- **System testing** is an **essential** part of the software development process and is typically performed using **system testing frameworks** such as **Selenium**, **Appium**, or **Robot Framework**.



System Testing

- **System testing** is a *software testing technique* that involves testing the entire software system as a whole.
- The **main goal** of system testing is to identify defects and issues in the software system as a whole, including its functionality, performance, and usability.
- **System testing** is typically performed after integration testing and before acceptance testing.
- It can be performed using a variety of **techniques**, including functional, non-functional, and regression testing.
- **System testing** is an essential part of the software development process and is typically performed using system testing frameworks such as Selenium, Appium, or Robot Framework.



System Testing

- **System testing** is a *software testing technique* that involves **testing the entire software system as a whole**.
- The **main goal** of system testing is to identify **defects** and **issues** in the software system as a whole, including its **functionality**, **performance**, and **usability**.
- **System testing** is typically performed after **integration testing** and before **acceptance testing**.
- It can be performed using a variety of **techniques**, including **functional**, **non-functional**, and **regression** testing.
- **System testing** is an **essential** part of the software development process and is typically performed using **system testing frameworks** such as **Selenium**, **Appium**, or **Robot Framework**.



System Testing Techniques

- **Functional testing** is a technique that involves testing the **functionality** of a software system to ensure that it meets the **specified requirements**.
- **Non-functional testing** is a technique that involves testing the **non-functional aspects** of a software system, such as its **performance**, **usability**, and **security**.
- **Regression testing** is a technique that involves **retesting** a software system **after changes** have been made to ensure that the changes have not introduced any new defects.



System Testing Techniques

- **Functional testing** is a technique that involves testing the **functionality** of a software system to ensure that it meets the **specified requirements**.
- **Non-functional testing** is a technique that involves testing the **non-functional aspects** of a software system, such as its **performance**, **usability**, and **security**.
- **Regression testing** is a technique that involves **retesting** a software system **after changes** have been made to ensure that the changes have not introduced any new defects.



System Testing Techniques

- **Functional testing** is a technique that involves testing the **functionality** of a software system to ensure that it meets the **specified requirements**.
- **Non-functional testing** is a technique that involves testing the **non-functional aspects** of a software system, such as its **performance**, **usability**, and **security**.
- **Regression testing** is a technique that involves **retesting** a software system **after changes** have been made to ensure that the changes have not introduced any new defects.



Stress Testing

- **Stress testing** is a *software testing technique* that involves **testing the software system under extreme conditions** to determine its **stability** and **performance**.
- The **main goal** of stress testing is to identify **defects** and **issues** in the software system when it is subjected to **high loads** or **stressful conditions**.
- **Stress testing** is typically performed after **system testing** and before **acceptance testing**.
- It can be performed using a variety of **techniques**, including **load testing**, **performance testing**, and **scalability testing**.
- **Stress testing** is an **essential** part of the software development process and is typically performed using **stress testing frameworks** such as **JMeter**, **Gatling**, or **Locust**.



Demo Time!



Stress Testing

- **Stress testing** is a *software testing technique* that involves **testing the software system under extreme conditions** to determine its **stability** and **performance**.
- The **main goal** of stress testing is to identify **defects** and **issues** in the software system when it is subjected to **high loads** or **stressful conditions**.
- **Stress testing** is typically performed after **system testing** and before **acceptance testing**.
- It can be performed using a variety of **techniques**, including **load testing**, **performance testing**, and **scalability testing**.
- **Stress testing** is an **essential** part of the software development process and is typically performed using **stress testing frameworks** such as **JMeter**, **Gatling**, or **Locust**.



Demo Time!



Stress Testing

- **Stress testing** is a *software testing technique* that involves **testing the software system under extreme conditions** to determine its **stability** and **performance**.
- The **main goal** of stress testing is to identify **defects** and **issues** in the software system when it is subjected to **high loads** or **stressful conditions**.
- **Stress testing** is typically performed after **system testing** and before **acceptance testing**.
- It can be performed using a variety of **techniques**, including **load testing**, **performance testing**, and **scalability testing**.
- **Stress testing** is an **essential** part of the software development process and is typically performed using **stress testing frameworks** such as **JMeter**, **Gatling**, or **Locust**.



Demo Time!



Acceptance Testing

- **Acceptance testing** is a *software testing technique* that involves testing the software system to ensure that it meets the needs of its users.
- The **main goal** of acceptance testing is to identify **defects** and **issues** in the software system from the perspective of the end user.
- **Acceptance testing** is typically performed after **system testing** and *before deployment*.
- It can be performed using a variety of **techniques**, including **user acceptance**, **alpha**, and **beta** testing.
- **Acceptance testing** is an **essential** part of the software development process and is typically performed using **acceptance testing frameworks** such as **Cucumber**, **FitNesse**, or **Behave**.

Demo Time!



Acceptance Testing

- **Acceptance testing** is a *software testing technique* that involves testing the software system to ensure that it meets the needs of its users.
- The **main goal** of acceptance testing is to identify **defects** and **issues** in the software system from the perspective of the end user.
- **Acceptance testing** is typically performed after **system testing** and *before deployment*.
- It can be performed using a variety of **techniques**, including **user acceptance**, **alpha**, and **beta** testing.
- **Acceptance testing** is an **essential** part of the software development process and is typically performed using **acceptance testing frameworks** such as **Cucumber**, **FitNesse**, or **Behave**.

Demo Time!



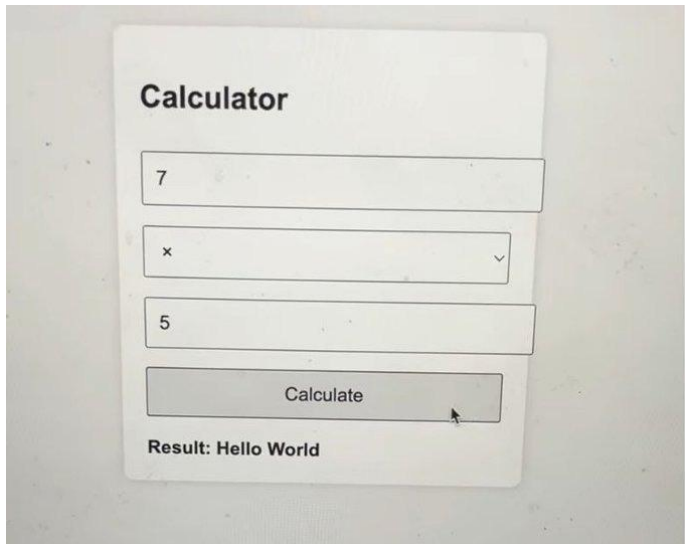
Acceptance Testing

- **Acceptance testing** is a *software testing technique* that involves testing the software system to ensure that it meets the needs of its users.
- The **main goal** of acceptance testing is to identify **defects** and **issues** in the software system from the perspective of the end user.
- **Acceptance testing** is typically performed after **system testing** and *before deployment*.
- It can be performed using a variety of **techniques**, including **user acceptance**, **alpha**, and **beta** testing.
- **Acceptance testing** is an **essential** part of the software development process and is typically performed using **acceptance testing frameworks** such as **Cucumber**, **FitNesse**, or **Behave**.

Demo Time!



Feature Error



Outline

1 Basic Concepts

2 Testing Levels



Thanks!

Questions?



Repo: www.github.com/EngAndres/ud-public/tree/main/courses/software_engineering_seminar

