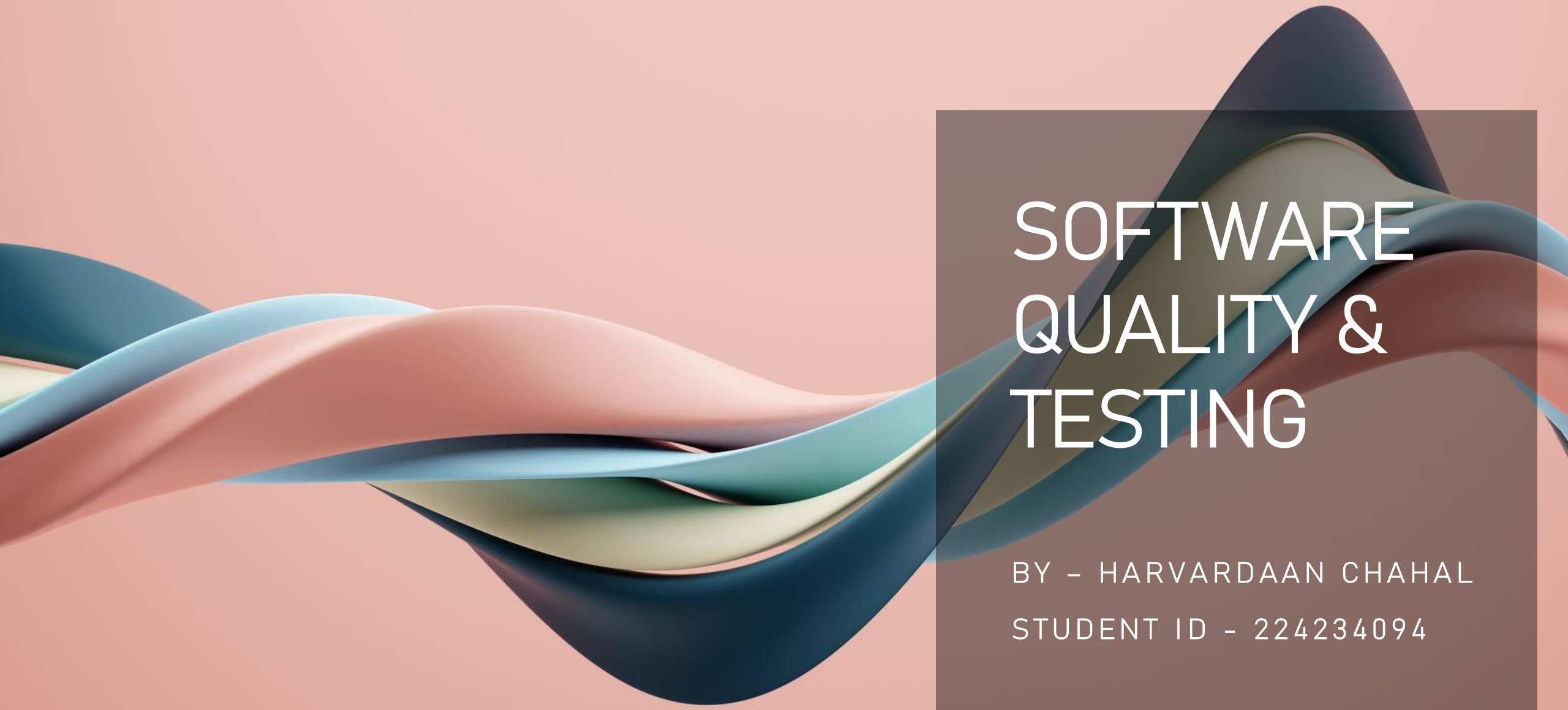


SOFTWARE QUALITY & TESTING

BY – HARVARDAAN CHAHAL

STUDENT ID - 224234094

Introduction to Software Quality and Testing

- **Definition of Software Quality:** Software quality refers to the degree to which a software product meets specified requirements and user expectations. It includes aspects such as reliability, maintainability, efficiency, and security.
- **Importance of Software Testing in Development:** Software testing is crucial to identifying and fixing defects before deployment. It helps ensure that the software functions correctly, meets user needs, and minimizes failures that could lead to financial loss or security vulnerabilities.
- **Overview of Testing Levels:** Software testing is conducted at multiple levels to ensure a thorough evaluation:
    - **Unit Testing:** Testing individual components or functions in isolation.
    - **Integration Testing:** Verifying interactions between integrated modules.
    - **System Testing:** Assessing the complete system's functionality.
    - **Acceptance Testing:** Ensuring the software meets business and user requirements.

Why is Software Testing Necessary?

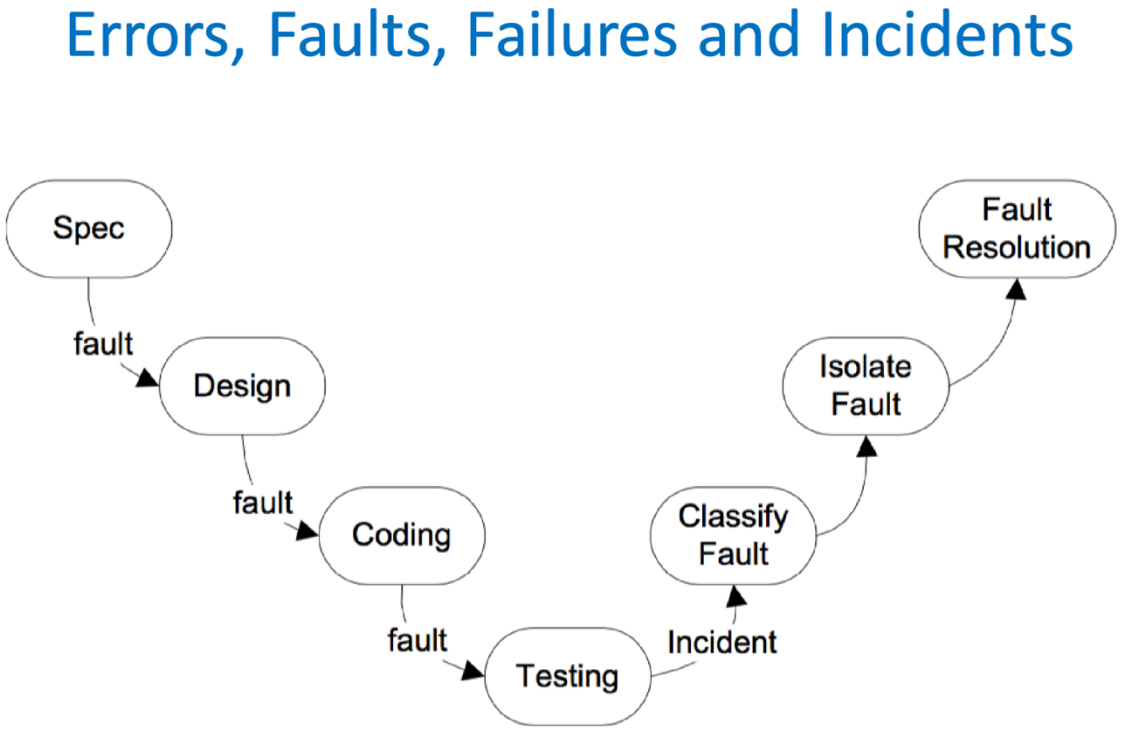
Prevents Critical Failures: Undetected software defects can lead to severe consequences, such as financial losses, security breaches, or system crashes. Testing helps identify and rectify such issues before deployment.

Ensures Software Reliability and Security: Regular testing ensures that software performs as expected in different scenarios, reducing the risk of unexpected crashes and security vulnerabilities.

Saves Cost and Time in the Long Run: Fixing bugs after release is costly and time-consuming. Detecting and resolving issues early in development minimizes expenses and prevents reputational damage.

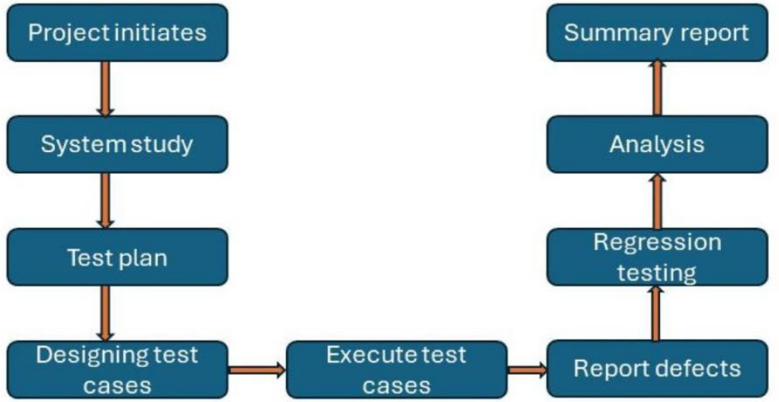
Improves User Satisfaction: Well-tested software enhances user experience by ensuring smooth functionality, reducing errors, and increasing customer trust in the product.

Examples of Software Failures



- NASA's Mars Climate Orbiter (1999) – Unit Conversion Error:
  - A mix-up between metric and imperial measurement systems caused the spacecraft to enter the Martian atmosphere at the wrong angle, leading to its destruction.
  - This highlights the importance of consistent standards and rigorous testing in mission-critical software.
- Knight Capital Trading Glitch (2012) – $440M Loss Due to Faulty Deployment:
  - A misconfigured trading algorithm triggered uncontrolled stock trades, leading to massive financial losses in just 45 minutes.
  - Demonstrates the necessity of thorough testing before deploying financial software in live environments.
- AT&T Long Distance Network Collapse (1990) – Cascading Software Failure Affecting 50M Calls:
  - A single software bug caused a series of cascading failures across 114 telephone switches, disrupting long-distance calls nationwide for 9 hours.
  - Showcases how small defects in critical software systems can have widespread consequences.

Levels of Software Testing

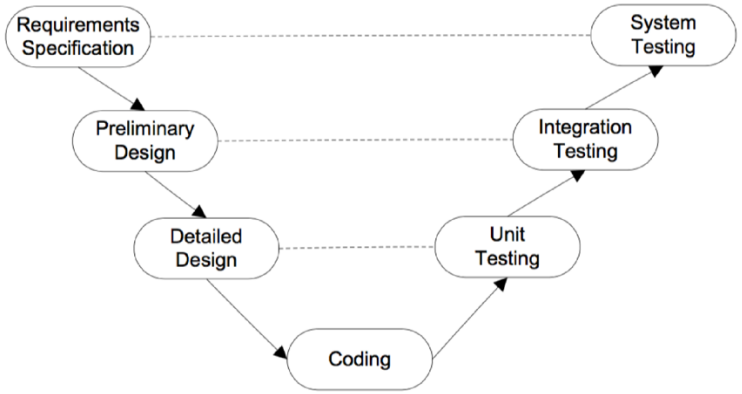


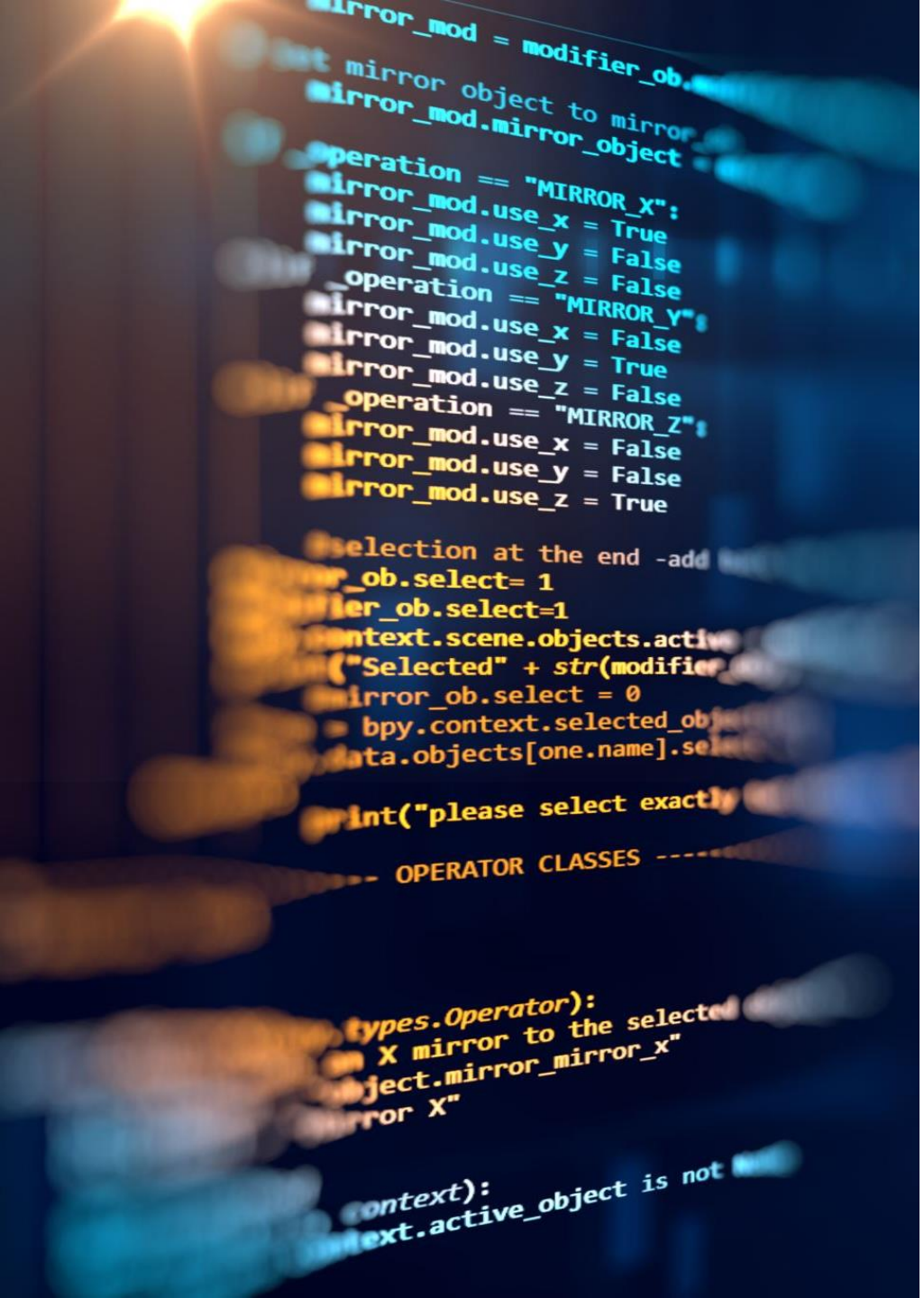
- Unit Testing:
  - Focuses on testing individual components or functions in isolation.
  - Ensures that each module performs correctly before integration.
  - Example: Using JUnit for Java to validate functions independently.
- Integration Testing:
  - Verifies the interactions between integrated modules.
  - Ensures that different parts of the system communicate correctly.
  - Example: Checking if a login module properly connects with a database.
- System Testing:
  - Validates the complete system functionality as a whole.
  - Ensures that the integrated components work together in real-world scenarios.
  - Example: Running a full test on an e-commerce platform to validate order processing.
- Acceptance Testing:
  - Verifies that the software meets business and user requirements.
  - Usually performed by end users or stakeholders before deployment.
  - Example: Conducting user acceptance testing (UAT) for a banking app before launch.

Key Testing Approaches

- Manual Testing:
  - Involves human testers executing test cases without automation.
  - Useful for exploratory testing, usability testing, and scenarios that require human intuition.
  - Example: A tester manually checking a website's navigation flow.
- Automated Testing:
  - Uses software tools to execute pre-scripted test cases.
  - Enhances efficiency, repeatability, and accuracy for regression and performance testing.
  - Example: Running Selenium scripts to verify website functionality.
- Functional Testing:
  - Ensures that the software behaves according to the specified requirements.
  - Focuses on input/output validation, user interactions, and expected functionality.
  - Example: Checking if a banking app correctly processes transactions.
- Non-functional Testing:
  - Evaluates system performance, security, and usability.
  - Ensures that the software meets industry standards and provides a seamless user experience.
  - Example: Performing load testing to see how an application handles high traffic.

Level of Testing (the V-model)





Software Quality Assurance (SQA) Principles

- Continuous Process Improvement: This principle emphasizes the importance of regular reviews and improvements of software development and testing processes. The goal is to increase efficiency and quality over time by identifying areas for improvement and implementing changes as needed.
- Code Reviews and Inspections: Code reviews involve systematically inspecting the codebase to find potential defects or issues before the software goes live. Inspections are performed by peers or senior developers to maintain coding standards and improve code quality.
- Adherence to Industry Standards: Following established software engineering standards like ISO 9001 and IEEE 829 ensures that software development processes align with best practices. These standards guide the development, testing, and maintenance of software to deliver reliable and efficient products.

Real-world Applications of Software Testing

Mobile Applications: Mobile apps need to function properly on a wide range of devices, screen sizes, and operating systems. Testing is essential to ensure compatibility and that the app performs optimally in real-world scenarios, avoiding crashes and bugs that could negatively affect user experience.

Banking and Finance Applications: Software testing is critical in the banking and finance sectors, as it ensures the accuracy, reliability, and security of financial transactions. Testing helps prevent fraud, errors, and system failures that could result in financial loss and damage to a company's reputation.

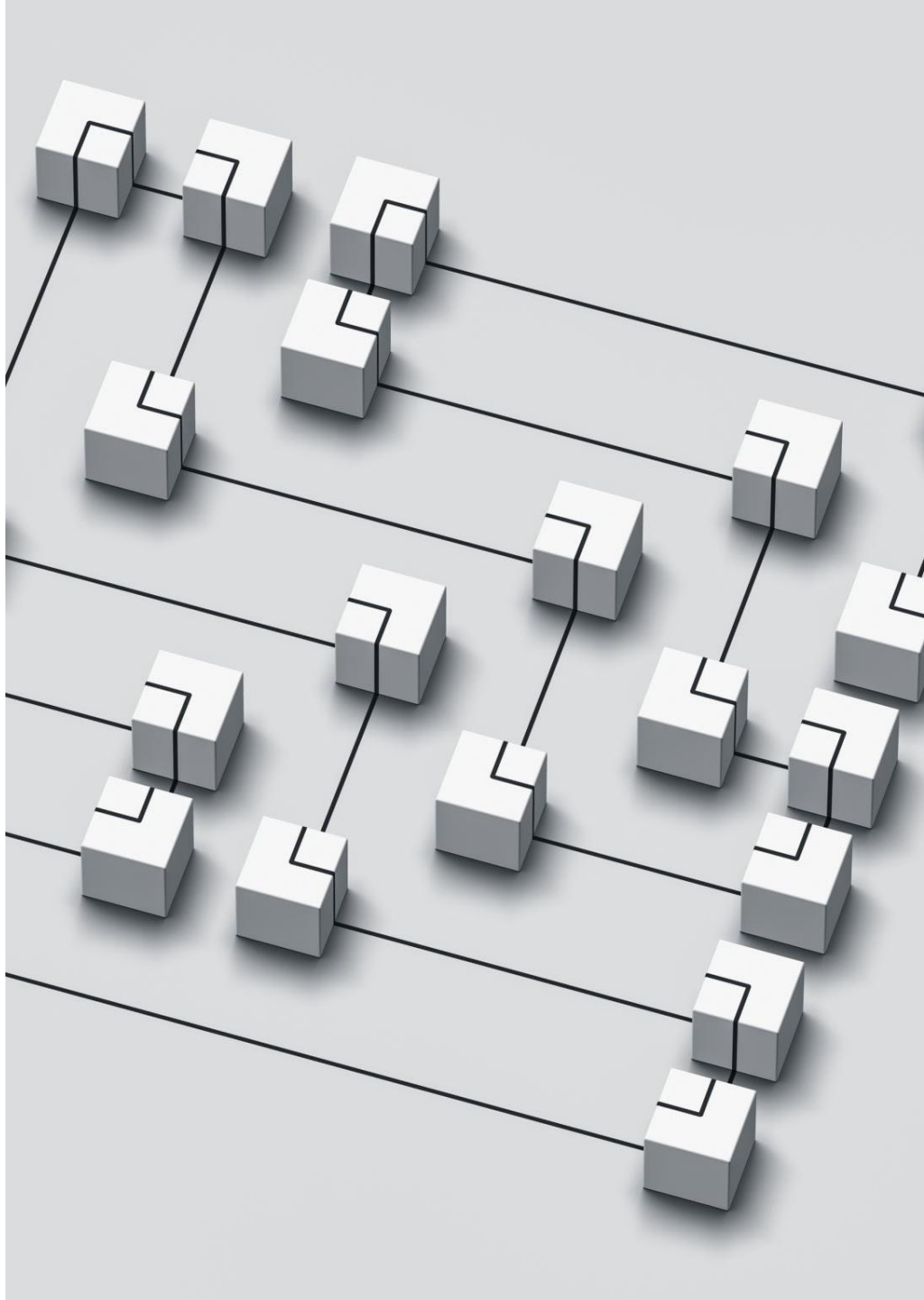
Automotive and Healthcare Systems: Embedded software in vehicles and medical devices is life-critical. Testing these systems is vital to ensure they operate correctly, meet regulatory standards, and ensure the safety of users. For example, ensuring the software in a car's braking system or a medical device like a pacemaker functions without failure is essential for preventing accidents and saving lives.

Software Failure: Therac-25

- Incident: Therac-25 was a radiation therapy machine used to treat cancer patients. Due to software errors, the machine administered lethal overdoses of radiation, leading to patient deaths and injuries.
- Causes:
  - Counter overflow and improper error handling: The software was unable to handle specific input conditions correctly, causing dangerous errors.
  - Lack of experience: The programmer responsible for the software lacked experience with real-time systems, contributing to the oversight in system design and testing.
  - Inadequate testing: There was insufficient testing and safety analysis of the software, which would have likely identified the flaws before deployment.
- Outcome: The failure resulted in the deaths of 3 patients and the severe injury of 4 others. This tragic incident highlighted the devastating consequences of poor software testing in life-critical systems and underscored the need for rigorous quality assurance in such industries.

Software Quality Standards

- ISO 9000: A set of standards focused on quality management and ensuring customer satisfaction. It provides a framework for organizations to establish processes that enhance product quality and continuous improvement.
- CMMI (Capability Maturity Model Integration): A model that provides organizations with a roadmap to improve their software development processes. It is widely used in government contracts and helps businesses assess and improve their capability in delivering high-quality software.
- ISO/IEC 15504 (SPICE): A framework for assessing and improving the software development and maintenance processes. It defines best practices for software engineering and ensures the consistent delivery of high-quality software products by evaluating software processes.



Conclusion & Summary

Software Testing Ensures Reliability: Testing helps identify defects early in the development process, ensuring that the software functions as intended and reducing the risk of failure upon deployment.

Different Testing Methods Address Various Aspects: The slide highlights the importance of different testing methods, such as:

Unit Testing: Testing individual components of the software.

Integration Testing: Ensuring components work together as intended.

System Testing: Testing the entire system as one.

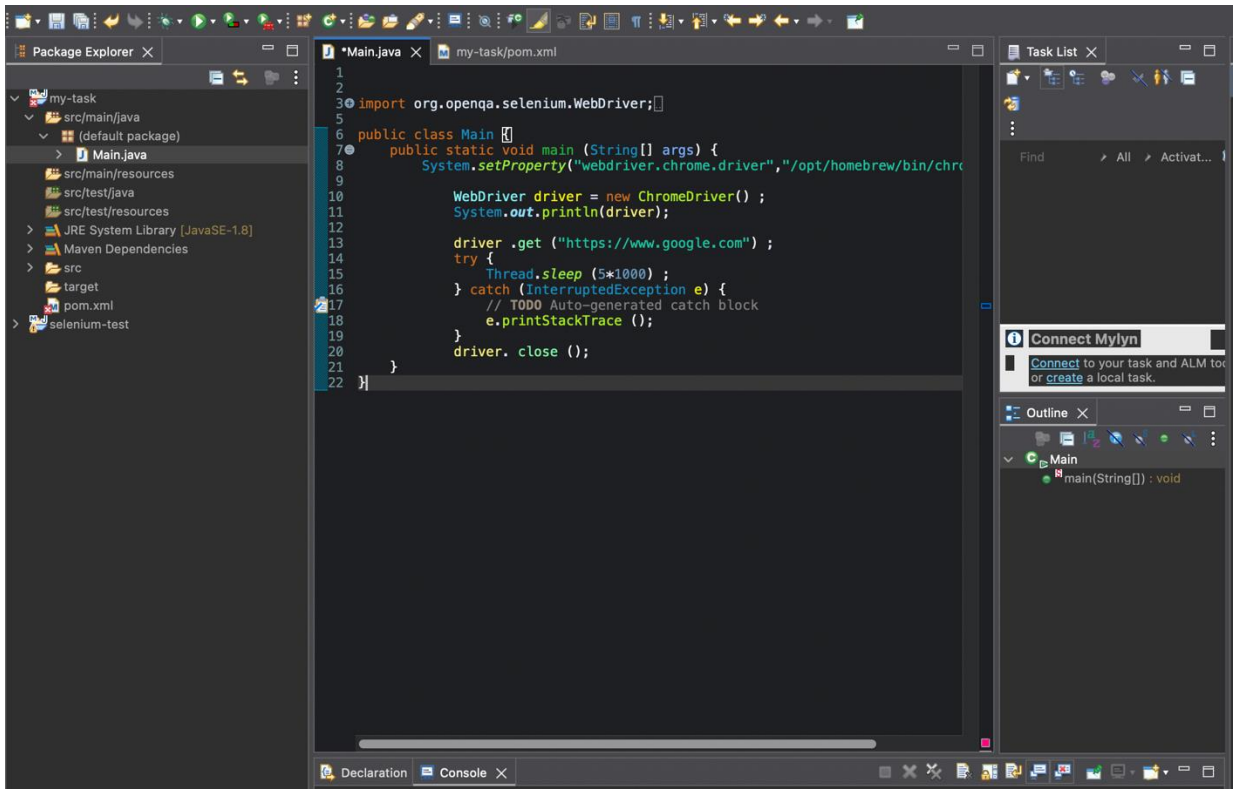
Acceptance Testing: Ensuring the software meets user needs and requirements.

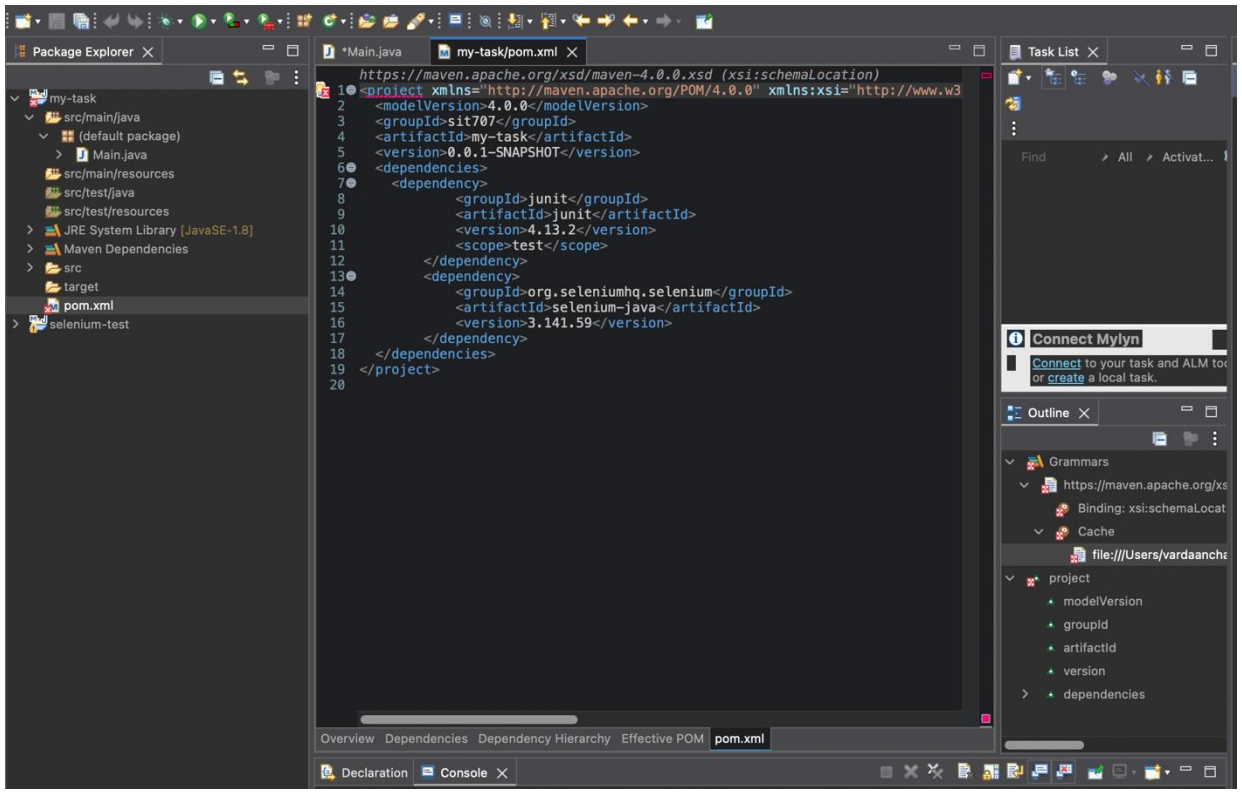
Real-world Failures Demonstrate the Need for Rigorous Testing: Examples like the Mars Climate Orbiter and Therac-25 serve as cautionary tales about the severe consequences of insufficient testing. These incidents emphasize the importance of comprehensive testing to prevent disastrous outcomes.

Following Best Practices and Industry Standards Leads to High-Quality Software: Adhering to software testing frameworks, industry standards, and best practices improves the overall quality, reliability, security, and user satisfaction of software products, reducing the risk of errors and failures in real-world applications.

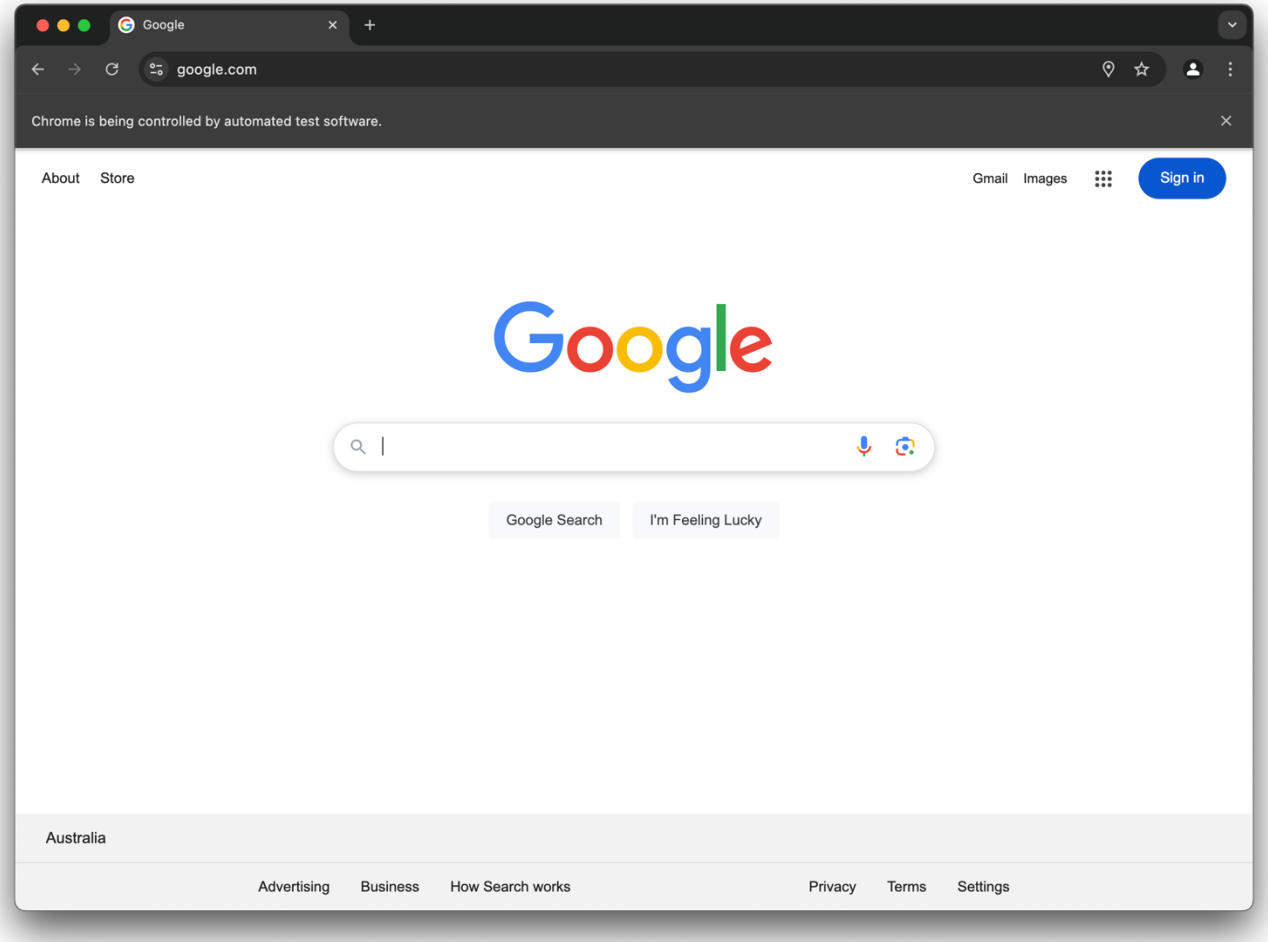
EVIDENCE

CODE





OUTPUT



GITHUB LINK :- https://github.com/hxryy7/sit333-1.1p/tree/main