We invite you to participate in our research study focused on examining the key factors that influence bug fixing during unit testing, especially when using automated unit test generators. This study is designed to identify critical factors that play a significant role in the bug-fixing process through controlled experiments.
The experiment will take up to 60 minutes, followed by a post-experiment questionnaire that should take about 10 minutes to complete. Participants are required to use screen recording software during the experiment; this footage will help us gain a deeper understanding of the factors affecting bug fixing.

**Procedure:** You will be given two assignments, each with a 30-minute time limit. Your tasks will involve identifying and fixing bugs in a test suite. The post-experiment questionnaire will ask about your experiences and perceptions regarding the key factors in bug fixing.

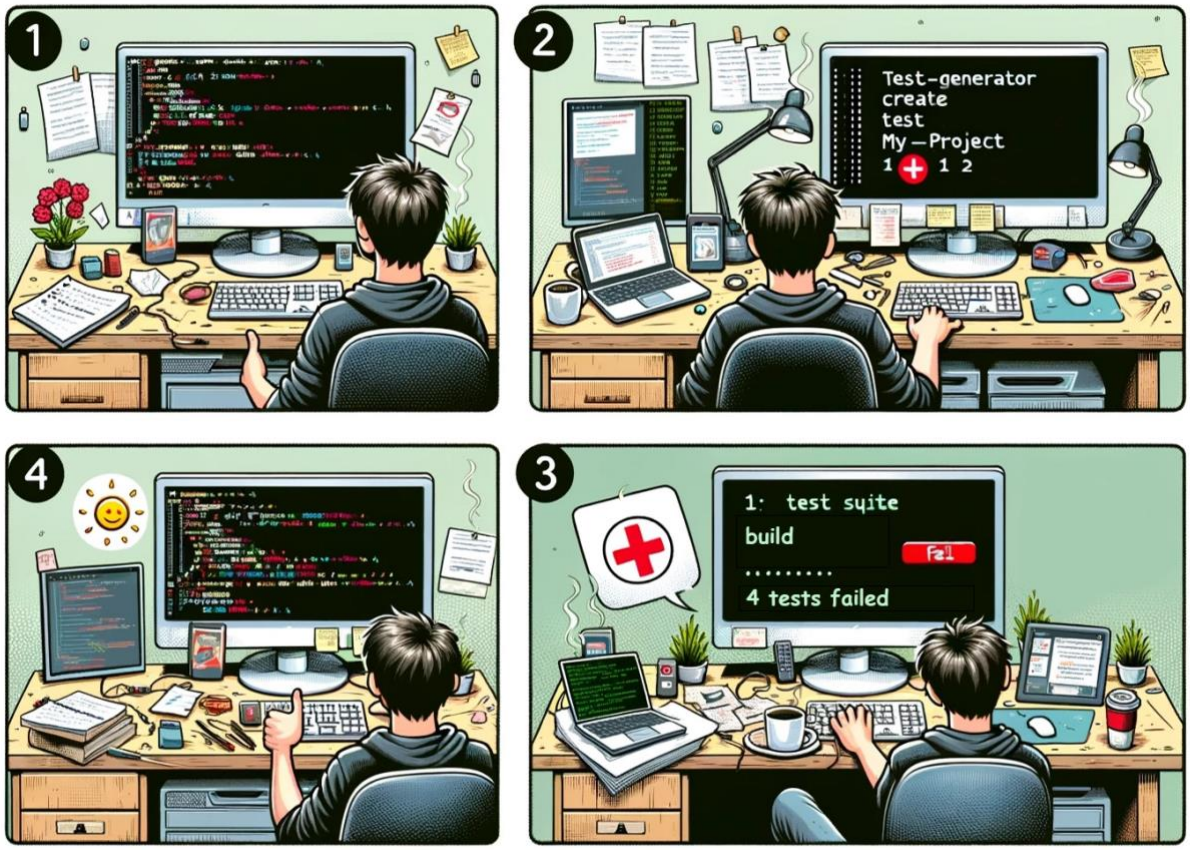
**Risks and Benefits:** The risk associated with participating in this study is minimal. Your participation will contribute to software testing research and the overall practices of software development.

**Confidentiality:** Your participation will remain confidential. We will ensure that no personal information is disclosed through the screen recordings. All data will be securely stored and accessible only to the research team, and findings will be reported in a way that does not identify individual participants.

**Voluntary Participation:** Participation is voluntary, and you are free to withdraw at any time without any penalty or loss of benefits.

**Consent:** By participating, you confirm that you have understood the information provided and agree to participate voluntarily, with the knowledge that you can withdraw at any time without any negative consequences.

Thank you for considering participation in our study.




(1) As a developer, your goal is to ensure your system is free of bugs. (2) Start by using a test generator to create test cases for your system. (3) Initially, the tests may pass, but as the code changes, some tests might fail. (4) Investigate the source code to understand the failures, fix the issues, and update the test cases as needed.

**Your main objective** is to efficiently fix as many bugs as possible in the shortest time. Use the automated test cases, adjust them to find the bugs, and then correct the source code. Each task has four bugs.

**Notes: Make sure:**

1. Java 8/11 is already installed on your computer.
2. You already have Intellij on your computer.
3. Use JaCoCo with Gradle for coverage reports, which is included in your project.
4. Feel free to adjust the assertions and test cases as needed!
5. Avoid using external sites like StackOverflow or ChatGPT! If you have questions, ask our team!
6. Start screen recording when you're ready to begin!

Avoid spending too much time on debugging. Begin your tasks after setting up your IDE.
**Tasks Description:**

# Task 1 - JSWeaponData (de.outstare.fortbattleplayer.model.impl.JSWeaponData)

JSWeaponData is designed to parse and store weapon data for a game, utilizing data from a CSV file that follows the format. It implements the WeaponData interface, defining methods to assess weapon attributes such as whether a weapon is a "Golden Gun", has specific bonuses like bayonets or various types of lubricants and oils, and other enhancements. The class has a nested class, WeaponGameData, to encapsulate the properties of each weapon, including its ID, damage range, offensive and defensive bonuses, and name. The main functionality of JSWeaponData is to load this weapon data into a map upon initialization. Additionally, it contains several methods to determine specific weapon enhancements based on predefined criteria, aiding in the dynamic assessment of weapon capabilities within the game environment.

# Task 2 - Budget (com.lts.caloriecount.data.budget)

The Budget class, which is part of a calorie counting application, is designed to manage and calculate daily calorie budgets based on user-defined time intervals and calories burned per hour. The class provides methods to set and get start and end times of the day (TimeOfDay objects), the interval between these times, and the calories burned per hour. It includes functionality to calculate the total calorie budget for a day or up to the current time, considering the specified time interval and calorie rate.