1. What is this project trying to achieve?

The project is aimed at implementing security requirements. In particular, the project aims to **automatically check for security requirements**. In this particular repository for the tutorial, it checks for incorrect password entry 5 times.

2. What methodology is used in the project?

The project is based on the **Requirements as code (RQCODE)** methodology. This is an approach in which **requirements are formalised in the form of program code**, allowing for automated verification and validation. It combines the concepts of **automated testing**, **requirement modelling** and **security compliance**.

3. Describe the tool used in the project to achieve the goal.

The project is using **RQCODE**, which describes and checks security requirements automatically. Some interfaces:

- `Requirement` – base class for requirements description.
- `Checkable` – interface to validate requirements.

4. During the presentation Ildar talked about “seamlessness”. How did he define it and why is it important for achieving the goals of the project?

By **"seamless ‘** Ildar meant **integrating security requirements into the development process without disrupting the developers’ workflow**. This means that the requirements are expressed in the same language (Java) in which the underlying code is written, eliminating the need for third-party tools or complex specifications.

5. How does the project propose to formalize security requirements?

The project assume to formalise security requirements in a form of Java classes, which derives from `Requirement` class and `Checkable` by transitivity. It allows:

- **Define requirements in the form of code.**
- **Automate verification.**
- **Integrate requirements into the software development process.**

6. What is the benefit of expressing requirements in Java before we write executable code in Java?

- **Automatic verification of code compliance with security requirements.** Code can be tested immediately after writing.
- **Single language for requirements and code.** Developers work with requirements in the same way as with regular classes, no need to learn new tools.
- **Minimising the cost of errors.** errors are identified early stages.