1. Using the Git versioning system and using a remote host such as GitHub, Bitbucket, Gitlab, Microsoft Azure, and so on. The following aspects will be checked:

* Using micro-commits (in other words, using several incremental commits during the implementation of a feature) instead of a single huge commit corresponding to the entire implementation of a feature (or even worst, corresponding to the entire application).
* Commits must contain explicit messages, including the task / story ID from Jira. Please use consistently only the story IDs or only the task IDs (as in the Git team exercise).
* Using separate branches for each new feature, BUG fix or refactoring (as in the Git team exercise).
* It is forbidden to have commits on the main branch (**mandatory named master**) without having Pull Requests. To restrict the access to the master please follow the instructions detailed [here](https://stackoverflow.com/questions/38864405/how-to-restrict-access-to-master-branch-on-git).

2. Using the Jira  Issue Tracking system. The following aspects will be checked:

* Proper split into Stories according to the Use Case diagrams and the requirements document approved by your teaching assistant.
* Detailed description of the main success scenario for each story and its extensions.
* Splitting each story into subtasks and detailing each subtask.
* Estimating each story and subtask. You can use at will the preferred estimation method (i.e., time, story points, t-shirt size).
* Assign all stories to team members
* Properly follow the states of each story (i.e., its transition through To Do, In Progress, Done).
* Integrating Jira with your Git Repository.

3. Using a build system (Maven, Gradle, npm, etc) that enables the usage of external libraries and the the automatisation of your final product build. The next aspects are very important:

* The code on **Git master**branch must be compilable.
* The building process result must be an executable **jar**(i.e., running the *java -jar your\_jar\_file.jar* must run your application).
* If using Maven, we will consider that the build steps are *mvnw clean install*(we consider you are using Maven wrapper).
* If using Gradle, we will consider that the build steps are *gradlew clean build* (we consider you are using Gradle wrapper).
* If using other build system you'll have to specify in the README.md file the build and run instructions.

4. In the design / implementation of the application the entities corresponding to the problem that is solved by your product must be represented as classes and objects (using the guidelines and heuristics provided during the lectures starting from the CRC cards).

5. Testing your application (using a corresponding testing framework). This will be detailed during the last assignment.

6. Presenting you application in week 13. You'll have:

* An online PowerPoint presentation that must contain:
  + Project name and the team members
  + Why have you chosen this project?
  + Use Case diagram
  + Story overview from Jira
  + How have you split your work (what stories were implemented by each team member).
  + An overview on your project in Git (total number of commits, total number of files, the number of commits performed by each team member). Please note that the mark is NOT proportional with the number of commits / files / modified files done by an author!
  + Some reports on tests will be added later during the last assignment.
* The presentation will be made by each team online (if we will not be able to meet in person) and it will take at most 15 minutes (10 minutes presentation + DEMO and 5 minutes questions).
* Each team member will present the work done by her/him
* The application DEMO must show at work the most important use cases of the application.