1. **Make a comparative table of the most common methodologies:**

|  | **Definition** | **Pros** | **Cons** | **Industry** |
| --- | --- | --- | --- | --- |
| **Waterfall Model** | Each phase of the model must be completed before the next phase can begin, without returning to the previous stage. | - This model is easy and clear to understand;  - Easier to manage in general, since each phase has specific deliverables  - Phases don’t overlap because each phase is processed/completed at a time. | - In the “Testing” phase it is very difficult to go back and change something that was not considered in the concept stage.  - An important defect/bug (i.e. not working software) could be built until late in the software life cycle  - Not suitable for projects where requirements are always changing  - Requires a highly qualified team (programers, designers and testers) for the planning phase | **-** Best suited for small projects with no uncertain requirements.  - Aerospace Industry (e.g. Nasa) where there is a almost unlimited of time to plan, unlimited time and budget |
| **V-Model** | The V-model is also a sequential path and each phase must be completed before the next phase begins. The testing of the product is planned in parallel with the corresponding phase of development, so testing begins at the stage of writing requirements. | - This model is also simple and easy to use;  - Some of the testings activities start before developing code, which could save time  - Defects are found at an early stage, contrary to the Waterfall model  - Also, works well for small projects where requirements are easy to understand and fixed | - Not very flexible when compared to other approaches  - Difficult to make changes later in the process  - It requires a significant amount of planning in the beginning  - Requires a highly qualified team  - Requirements are very well defined | - Aerospace and Automotive Industry, where requirements are very well defined since the beginning |
| **Incremental Model** | The incremental model is divided into various builds divided into smaller, more easily managed modules. Each build passes through design, implementation and testing phases. A working version of software is produced during the 1st build, and the process continues (adding new features) until the complete system is achieved. | - Generates a working software easily and quickly during the life cycle  - More flexible than the two mentioned above  - Less costly to change the requirements and lower initial cost to deliver  - Easier to test and debug during each iteration  - Feedback on the customer for each build | - Requires a highly qualified team to develop a good planning and design  - Needs a clear and complete definition of the whole system in the beginning | - E-commerce Industries: Where we can quickly launch an operational website and then add new features and improve it through time |
| **Iterative Model** | The interactive model we don’t need to start with the full specification of requirements. It divides the project into parts (stages and iterations) and going through the stages of the life cycle with every part. Each stage is independent, and the set of stages makes the final result. | - Build and improve the product step by step  - Track defects at an early stage  - The requirements for the final system are very well defined and understood from the beginning; | - Each phase is rigid  - Not all requirements are gathered up for the entire lifecycle | - |
| **Spiral Model** | There are successive iterations. During the first stages, product specifications are refined, and then new features and functions are added. After each iteration, there is a risk assessment to define risks and how to mitigate them. | - High amount of risk analysis, so risks are mitigated after each iteration | - Costly to implement | - Bank Industries: Softwares need to be secure and compliant with financial regulations |
| **Agile Model** | Development happens via short iterations (cycles). At the end of each of tem, the customer receives a working code or a product. | - Little planning is enough to start the project  - Higher customer satisfaction, because there is a continuous delivery of software  - Rapid delivery of a software product (MVP - Minimum viable product)  - Late requirement changes are not a problem | - Difficult to assess the total effort required at the beginning of the development cycle  - Projects can be taken off track if the final outcome of the project, in situations where the final outcome is not clear | - Web and Mobile App: Allows a quick software delivery and is oriented towards customer satisfaction and feedback |

**2. Write detailed answers (0.5-1 pages in volume) to the following two questions:**

* **In your opinion, why did the Agile manifesto appear?**

The Agile Manifesto was created as an alternative solution to the traditional and more rigid and inflexible software development approaches that existed at the time. These approaches were notoriously known to be highly driven towards documentation, and with rigid and inflexible development processes. There was an increasing need to make the software development life cycle more agile and changing towards a more flexible and adaptable process, but still follow a structure.

* **What problems did it have to solve and did it succeed?**

In my opinion, Agile helped solving problems like:

* **Inflexible and rigid software development processes:** Traditional processes are more rigid, slow and have a higher difficulty to handle changes in requirements or unpredictable challenges during the development phase.
* **Long development cycles:** We have to wait until the end of the life cycle to have a working software, and have stakeholders feedback on it.
* **Limited customer feedback:** No customer feedback or involvement until the end of the project.There is also a higher risk that software product doesn’t meet the customer expectations, making it much more costly to make changes.
* **Market Changes:** Slow when it comes to adapting and releasing new features that accompany the changing market.

**3. You are the founder of a startup planning to launch a mobile application for sharing cat photos.**

**What methodology will you choose for the development process and why? Support your answer.**

If I wanted to launch a mobile application for sharing cat photos, I believe I would choose the Agile Methodology. Agile allows me to build an initial photo sharing App with the basic features, and have a minimum viable product that I can quickly launch into the market and then refine it according to customer feedback and needs. I can also prioritize features based on how valuable they will be to my customers.

Additionally, by delivering a “final'' and usable App more quickly, I can benefit from having revenue early on which I can invest to improve additional features in my App.

Finally, I could also opt-in for a Scrum framework that would allow me, my team (development team, testers and stakeholders) to have a clear and real-time overview of the project, making it easier to track the progress of my project.