Week_1_pf
Discuss the SDLC methodology you chose for the software development project. Weigh the pros and cons and explain why you chose this method over other methods.

Let's define our way of the discussion. We start our journey from no plan/methodology to strict plan/methodology.

Say, a neighbor wants me to develop an application that monitors our area and counts homeless cats. I liked this idea and started to develop an application. During the coding I realized that I need an interface to the public cameras. Found a way to get information from the cameras, next challenge to understand how to recognise a cat. After googling I found a library of python, tensorFlow and trained a model to be able to recognize who is who. Next challenge was to count cats only once. Here I end my imaginary story. Only thing is - my neighbor could not make my cooliest application because it does not have a UI. One day a fire happened and my garage (data center) went out completely. Oops.

Opposite direction - I was talking with stakeholders, customers etc in order to gather requirements. After that, I spent months analyzing, to understand use cases, to make an application scalable, available, maintainable, fault tolerant and bug free. So far, so good. Time flies by. After confirmation from people that everything looks good, every potential problem has its solution. I started to code, then started to test. During the testing I found a bug in the design. Such a pity. Should start over.

The second example was an attempt to show how waterfall works.

Probably all these methods were correct decades ago. Today, it is so ridiculous. A couple of things we would like to keep in mind. Changes and rapid deployment. Modern software changes often and drastically. Clients always want this feature or that one. And any single change should be deployed immediately.

How any way could be on time and our customers will be satisfied.
I am a big believer in the Agile development life cycle. We are breaking a project into phases, continuously collaborating with customers and other stakeholders and improving.
Cycle is simple: planning, executing/coding/testing/drawing and evaluating. A work divided by timeline, between two weeks to a month. Ideal team amounts between 3 - 6 people. Each spring the process begins with planning for the next two weeks. Each person takes part of work and continues until it finishes and takes another task/stroy from the backlog. Ideally no changes during the sprint. Development done by TDD, test driven development. Any added business logic covered by appropriate tests. New functionality testing by AB testing. In addition, each commit retriggers deploy in the dev/qa/stage environment. If something is broken, immediate feedback is gotten and if needed fixed. Some follow the rule - once a week deployment to production. I would rephrase it - each merge to master should be redeployed on production. Customers can take part in planning and evaluating. At the end of each Sprint, a team has to show some chunk of working software, design or whatever makes sense. When Spring finishes,

another step comes, postmortem. Intention of this step is to get feedback on how the current sprint went, how it can be improved or changed.

Agile has some forms like: Scrum, Extreme programming or Kanban.
Summing up: important things are small periods of time for part of bigger work, agility, ability to absurd changes and rapid deployment. No years or months for waiting or replanning.

Such an approach makes development more manageable, collaborative and innovative. However, not everybody can work with such methodology, people should be appropriate, responsible and professional. Each one depends on others, not on task and task but by success in general.

This is the first week of my study. I followed and was reading all the instructions and material. Videos were informative.
Difficult and interesting things were discussed in the discussion forum :-) where you should imagine how you would develop a product for a bank. For me the challenge was not in the process itself but a way to explain my thoughts and approaches. Writing assignment was also challenging. But reading, looking at articles and simply blog posts gave me direction.

References:

1. Agile coach https://www.atlassian.com/agile
2. What Are The Steps of the Software Development Lifecycle http://www.kaltura.com/tiny/yex8c
3. Destin Learning. (2019, February 15). *What are the steps of the software development lifecycle?* [Video]. YouTube. https://www.youtube.com/watch?v=gNmrGZSGK1k (10:35)
4. Dmitry-Project Management Basics. (2020, March 3). *SDLC: All you need to know about software development life cycle* [Video]. YouTube. https://www.youtube.com/watch?v=cY-3wdvbz6o (15:26)
5. Jevtic, G. (2022, November 17). What is SDLC? Phases of software development, models, & best practices. PhoenixNap global IT services. https://phoenixnap.com/blog/software-development-life-cycle