**Software development methodology**

**Waterfall development methodology**

For this software design, I adopted the waterfall methodology since it is a rigid linear model that consists of sequential phases (requirements, design, implementation, verification, maintenance) focusing on distinct goals. Each phase must be 100% complete before the next phase can start. There’s usually no process for going back to modify the project or direction.

In the context of developing an application software that regulates the amount of time spent on a social media platform, here's how the waterfall methodology can be applied:

***Requirements Gathering***: Begin by gathering all the requirements for the application, including the home page, user login, and linking with social media platforms. Define the features and functionality needed for each component.

***System Design***: Once the requirements are finalized, proceed to design the overall system architecture. Create a detailed design for the home page, user registration/login system, and integration with social media APIs. Identify the necessary technologies, frameworks, and infrastructure.

***Implementation***: With the design in hand, start implementing each component of the application in a sequential manner. Begin with the home page, ensuring its layout, navigation, and basic functionality are implemented. Then proceed to develop the user registration/login system, followed by the integration with social media APIs.

***Testing***: After implementing each component, conduct thorough testing to ensure that they meet the specified requirements and functionality. Perform unit testing to validate the individual components, and integration testing to ensure seamless communication between the different modules.

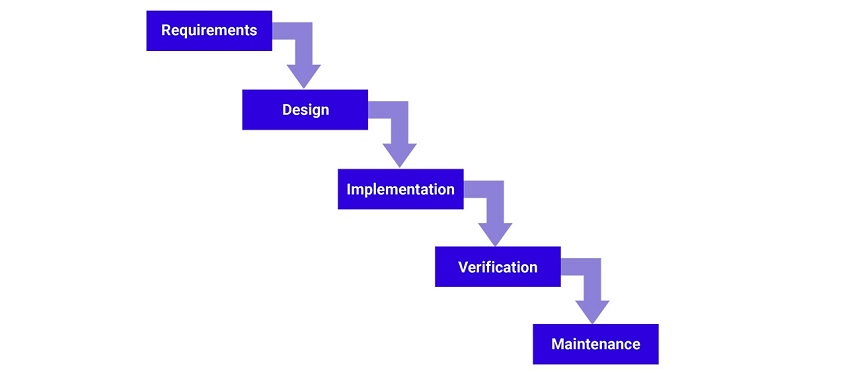
***Deployment***: Once all components have been implemented and tested, deploy the application to a production environment. Set up the necessary infrastructure and servers to host the application and make it accessible to users.

***User Acceptance Testing***: Invite users or selected beta testers to perform user acceptance testing. Allow them to use the application and provide feedback on its functionality, usability, and performance. Make any necessary refinements based on the feedback received.

***Documentation***: Create comprehensive documentation that describes the architecture, design, and functionality of the application. Document the user login process, linking with social media platforms, and any suitable configuration or setup information.

***Maintenance***: Once the application is deployed, provide ongoing maintenance and support. Address any reported issues or bugs, and make necessary updates or enhancements based on user feedback and evolving requirements.

Below is the diagram of the waterfall approach used in the development of this software.



**Pros and Cons of Waterfall design methodology**

The fact this design methodology was adopted for this software development doesn’t mean it is 100% perfect, it has its own pros and cons as we are going to explore.

***Pros of the Waterfall methodology***

* The waterfall methodology follows a sequential and well-structured approach, with clear phases and deliverables. This can provide a sense of stability and control, especially for projects with well-defined and stable requirements.
* Each phase in the waterfall model emphasizes documentation, which can be beneficial for projects requiring thorough documentation, such as those involving regulatory compliance or long-term maintenance.
* The linear nature of the waterfall model makes it relatively easy to manage and understand the project's progress. It simplifies project planning and resource allocation since each phase has specific goals and deliverables.

***Cons of the Waterfall methodology***

* The sequential nature of the waterfall model makes it less adaptable to changes in requirements or scope. Once a phase is completed, it is difficult to revisit and make significant modifications without impacting the entire project timeline.
* User feedback is typically collected in the later stages of the waterfall process, such as during user acceptance testing. This means that potential issues or required changes may not be discovered until later, which can result in costly rework.
* : Since all requirements are defined upfront, there is a higher risk of misalignment between what is specified and what the end users actually need. This can lead to wasted efforts and dissatisfaction with the final product.

Long Delivery Cycle: The linear nature of the waterfall methodology often leads to longer project timelines, as each phase must be completed before moving to the next. This can result in delayed delivery and decreased responsiveness to changing market needs.

Limited Collaboration: The waterfall model can discourage collaboration between different stakeholders, as each phase typically involves separate teams or individuals. This may hinder effective communication, feedback exchange, and knowledge sharing.