**PROJECT: METHODOLOGY**

The methodology refers to the approach in which the software was developed. In software engineering we have various methodologies for software development such as waterfall model, V-model, prototype model, spiral model and Extreme programming (XP).

In the career guidance system, agile methodology proves to be an ideal approach due to its flexibility, iterative nature, and focus on continuous improvement. Agile is fundamentally different from traditional development methods because it breaks down the development process into small, manageable cycles called **iterations** or, in the case of Extreme Programming (XP), extreme programming cycles. These cycles are typically short, usually lasting 1-2 weeks, and at the end of each cycle, a **working version of the system** is delivered, incorporating new features or improvements. This approach ensures that the system is constantly evolving and improving, rather than waiting until the end of the development process to test the product.

The key advantage of Agile is that it allows for frequent feedback after testing, which is particularly valuable in the context of a Career Guidance System, where they are different technological approaches to develop the Career Guidance System. Since career advice and recommendations need to be personalized and adaptable to rapidly changing market demands, Agile’s flexibility enables the system to respond to new insights, evolving user needs, or emerging career trends. Each iteration provides an opportunity to integrate user feedback, ensuring that the system is aligned with the goals and requirements of students using the system for planning their courses of study and careers.

The various other methodologies used in software development include the following;

**(i). Spiral model**

The Spiral Model is a software development methodology that combines iterative development with elements of the traditional waterfall approach. It is structured around repeating cycles, or "spirals," each consisting of four key phases: planning, risk analysis, engineering, and evaluation. This model is well-suited for large, complex, and high-risk projects, as it allows for ongoing risk management and frequent adjustments based on evolving requirements.

**Advantages**

1. Continuous risk assessment and mitigation help identify and address potential issues early.
2. Development progresses in cycles, allowing for incremental improvements and early detection of problems.
3. Adapts to changing requirements and feedback at each cycle.
4. Regular reviews ensure the product aligns with user expectations.
5. Continuous testing and validation maintain high product quality.  **Disadvantages**
6. It can be difficult to manage, especially for smaller projects.
7. Repetitive cycles can lead to higher costs in terms of time and resources.
8. Frequent cycles can extend project timelines.
9. Risk analysis and management require specialized skills.
10. Extensive planning and documentation for each iteration can be burdensome.

**(ii) Waterfall model**

The Waterfall Model is one of the earliest and most straightforward software development methodologies. It follows a linear and sequential approach, where the project progresses through clearly defined phases: requirements gathering, system design, implementation, integration and testing, deployment, and maintenance. Each phase must be completed before moving on to the next, and there is little to no overlap between the phases. This model is best suited for projects with well-defined requirements that are unlikely to change throughout the development process.

The Waterfall Model is often used for smaller projects with clear objectives, where the project scope and requirements are well understood from the beginning. It is ideal when a structured, systematic approach is needed, with minimal changes expected during development.

**Advantages**

1. Simple and Easy to Understand: The linear structure makes it easy to understand and manage, especially for smaller projects.
2. Clear Requirements: The Waterfall Model requires that requirements are defined upfront, which helps in creating clear, detailed documentation.
3. Structured Approach: The process is highly structured, ensuring that each phase is completed thoroughly before moving on to the next.
4. Easy to Track Progress: Since the phases are sequential, it is easy to measure progress at each stage.
5. Ideal for Small Projects: Best suited for projects with well-defined requirements and no expected changes during the development process.

**Disadvantages**

1. Once a phase is completed, it's difficult to go back and make changes. This makes it unsuitable for projects with evolving or unclear requirements.
2. Testing only begins after the development phase is completed, meaning defects may not be identified until the end of the project.
3. The model assumes that all requirements are known at the beginning, which can be unrealistic for many projects.
4. Since the model progresses linearly, there's no opportunity for customer feedback until the product is fully developed, making it harder to align the final product with user needs.
5. The lack of flexibility and late-stage testing can lead to high risks if issues are discovered too late in the process.

**(iii) The V-model**

The V-Model (Verification and Validation Model) is a software development methodology that extends the Waterfall Model by emphasizing the parallel relationship between development phases and testing phases. In the V-Model, each development phase corresponds to a testing phase, ensuring that testing is integrated throughout the development process. The process flows downwards through the stages of requirements analysis, design, and coding, and then the corresponding testing phases—unit testing, integration testing, and system testing—occur in parallel, moving upwards in a "V" shape.

The V-Model is best suited for projects with well-understood and stable requirements, and it ensures a rigorous, structured approach to both development and testing.

**Advantages**

Early Testing: Testing begins early in the process, ensuring that issues are identified and fixed at the right time, avoiding defects being found too late.

1. The model is easy to understand, with clear and sequential stages for both development and testing.
2. By linking each development phase with a corresponding testing phase, the V-Model ensures that quality is built into the product from the outset.
3. Straightforward Progress Tracking: Since the process is linear and each development phase has a corresponding testing phase, progress is easy to track and measure.
4. The V-Model's simplicity makes it a good fit for small projects or projects with clear requirements and limited scope.

**Disadvantages**

Inflexibility to Changes: Like the Waterfall model, the V-Model is inflexible, making it difficult to accommodate changes in requirements once development has started.

1. User feedback is typically not incorporated until after the development is complete, which may lead to the product not meeting expectations.
2. If issues or changes are discovered late in the process, the cost and risk of addressing them can be high, particularly since testing is conducted after each development phase.
3. The V-Model is less suitable for large, complex projects or projects with evolving requirements, where the iterative approach would be more beneficial.
4. Extensive documentation for each stage of development and testing can lead to a high administrative overhead.

**(iv) Prototype model**

The Prototype Model is a software development methodology that focuses on creating an early version (prototype) of the system, which is then tested and refined through user feedback. This prototype is built quickly with minimal features to simulate the key aspects of the system. Users interact with the prototype, provide feedback, and suggest changes or additional features. Based on this feedback, the prototype is improved and refined through several iterations until it meets the user's needs. The goal of the prototype model is to deliver a working version of the system that allows for hands-on user involvement early in the development process.

This model is particularly useful when the system requirements are not well-defined initially, or when users are unsure of what they want. It allows for quick adjustments based on user input and helps ensure that the final product aligns closely with user expectations.

**Advantages**

1. Users can interact with the prototype and provide valuable feedback at an early stage, which helps guide the development in the right direction.
2. The prototype model is flexible, allowing for changes and modifications based on feedback throughout the development process.
3. As users interact with the prototype, their feedback helps clarify and refine system requirements, which may not have been clear at the start.
4. By getting feedback early, potential risks or misunderstandings can be identified and addressed before they grow into larger issues.
5. Users are more likely to be satisfied with the final product because their needs and expectations have been considered and incorporated throughout the development process.

**Disadvantages**

1. The initial prototypes may be rudimentary and not fully represent the final product, leading to user expectations that might be too high.
2. Iterative prototyping can become time-consuming if frequent changes and refinements are needed, potentially leading to delays.
3. Since prototypes are often not the final product, there is a risk that the development process becomes focused on refining the prototype rather than building the complete system.
4. The repeated iterations of building and refining prototypes may increase development costs, especially if there is a lot of reworks required.
5. Continuous changes and user requests during the prototyping phase can lead to scope creep, where the project's requirements keep expanding, potentially leading to an over-complicated system.

In the development of career guidance system, I prefer using Agile methodology. Under Agile methodology I would prefer using Extreme programming (XP) whereby project is subdivided into iterations known as extreme programming cycles. Whereby each cycle would take for 1-2 weeks, in each cycle it involves extreme and best practices for short period of time. The contributes to the development of best quality software.

**Reasons why Agile methodology is preferred in this project**

1. It is iterative where small iterations are developed and tested before being integrated to the career guidance system.
2. It is flexible to change the approaches and technologies used in the development of career guidance system, when the initial technologies used to develop the career guidance system are not giving out the required results.
3. Production of high-quality software, because the agile methodology includes testing at every iteration, therefore this would contribute to final quality of the software.
4. Early detection of the some the challenges that might be encountered in the system. Therefore, challenges can be mitigated early enough