QUESTION 9 and 10 Responses:

9. What is a unique but essential approach you use in your development process? {Be creative and try to stand out but also remain original}

My development process has been enclosed with three key unique approaches as discussed below:

Agile Ecosystem Mapping (Agile development process)

From Agile methodology perspective, this is flexible approach to software development that emphasizes collaboration, adaptability, and customer-centricity. It originated as a response to the limitations of traditional linear approaches such as waterfall methods and aims to deliver value to customers in a more incremental and iterative manner. The creative part is when the team utilizes "Agile Ecosystem Mapping." They create an ecosystem map that visualizes the project's components as interconnected entities within a dynamic environment. This map adapts and evolves as the project progresses, allowing for flexible adjustments based on changing requirements and external factors. Here all the development units are kept intact with main objectives in sync with below stages:

1. **Feasibility study**
2. **System design**
3. **Implementations/ development**
4. **Testing (Unit testing, UAT and Pentesting)**
5. **Deployment.**
6. **Support and documentations.**

This process can be efficient enough with:

Daily Stand-up Meetings: Short, daily meetings where team members share updates on their progress, challenges, and plans for the day.

- **Product Backlog:** A prioritized list of features, enhancements, and fixes that need to be addressed throughout the project.
- **User Stories:** Short, simple descriptions of a feature from an end-user's perspective, often written on index cards or digital tools.
- **Burndown Charts:** Visual representations of work completed versus remaining work, helping teams track progress within a sprint.
- **Retrospectives:** Scheduled sessions to review the previous sprint and identify areas for improvement.
- **Continuous Integration and Continuous Delivery (CI/CD):** Practices that ensure frequent integration of code changes and automated deployment to production environments.
- **Customer Collaboration:** Close collaboration with customers, users, and stakeholders is fundamental. Regular feedback and involvement throughout the development process help ensure that the software meets their needs and expectations.
- **Face-to-Face Communication:** While remote collaboration is common, Agile values face-to-face communication whenever possible. Direct interactions help convey information more effectively and build strong team relationships.
- **Working Software as a Measure of Progress:** Agile focuses on delivering working software rather than extensive documentation. The primary measure of progress is the software's functionality and features.

Eco-User Story Cultivation

Incorporating the principles of permaculture, user stories are seen as "Eco-User Stories." These stories emphasize sustainable software features that offer long-term value, reduce technical debt, and align with users' evolving needs. Each story is designed to integrate seamlessly into the software ecosystem, much like how a permaculture garden achieves symbiotic relationships.

10. Outline the learning and research steps you follow when given a task that you do not yet have the skills/experience for:

Interesting question, I do have robust way of handling tasks. For the task that I yet to have the skill/ experience, I do take them as carrier challenges that required solutions. The best schematic approach steps are listed as below:

1. Understand the Task:
- Carefully read and comprehend the task requirements and objectives.
- Identify the specific aspects of the task that are unfamiliar to you.

2. Break Down the Task:
- Divide the task into smaller sub-tasks or components.
- Determine the dependencies between these sub-tasks.

3. Research and Learning Plan:
- List the skills, technologies, and concepts you need to learn to complete the task.
- Prioritize these items based on their importance to the task's completion.

4. Preliminary Research:
- Gather introductory materials like articles, tutorials, videos, and online courses.
- Aim to gain a high-level understanding of the technologies and concepts involved.

5. Choose Learning Resources:
- Select comprehensive resources such as reputable online courses, documentation, and books.
- Consider seeking recommendations from experienced peers or online communities.

6. Skill Acquisition:
- Engage in focused learning sessions for each skill or concept you need to acquire.
- Work through tutorials, exercises, and practice problems to reinforce your understanding.

7. Hands-On Practice:
- Set up a personal development environment to experiment and practice.
- Create small projects or prototypes related to the task to apply your new knowledge.

8. Learning by Doing:
- As you work on practical exercises, pay attention to the challenges you encounter.
- Seek solutions for specific issues as they arise.

9. Experimentation and Exploration:
- Go beyond tutorials and explore variations or advanced aspects of the technologies.
- Experiment with different configurations, tools, and approaches.

10. Seek Guidance: - If you face difficulties, don't hesitate to ask questions on online forums, developer communities, or chat platforms. - Engage with experienced developers who can provide insights and guidance.

11. Iterative Progress: - Regularly revisit the task's requirements and your understanding of them. - Update your learning plan and adjust your approach as needed.

12. Collaboration and Code Review: - Collaborate with colleagues or mentors who have experience in the relevant technologies. - Share your progress, ask for feedback, and discuss potential improvements.

13. Continuous Improvement: - Continuously refine your skills as you work on the task. - Reflect on your progress and identify areas for improvement.

14. Task Completion: - Once you feel confident in your understanding and ability to apply the required skills, start working on the task. - Apply your newfound knowledge to develop a solution.

15. Testing and Refinement: - Test your solution thoroughly to identify and address any issues. - Iterate and refine your work based on feedback and testing results.

16. Documentation: - Document your learning journey, challenges faced, solutions discovered, and the final implementation. - This documentation can be helpful for future reference and for sharing your experience with others.