Seat No.: _____ Enrolment No._____

GUJARAT TECHNOLOGICAL UNIVERSITY

BE - SEMESTER–VII (NEW) EXAMINATION – WINTER 2021

Subject Code:3171610 Date:15/12/2021

Subject Name: Agile Development and UI/UX design

Time: 10:30 AM TO 01:00 PM Total Marks: 70

MARKS

Q.1 **(a)** Define "Agile development". **03**

Agile development is a software development approach that emphasizes flexibility, rapid prototyping, and fast iteration. It is based on the Agile Manifesto, a set of values and principles that guide software development. Agile development methods are designed to be adaptable and responsive to change, allowing development teams to quickly respond to new requirements or challenges. Agile development is often used in conjunction with other methodologies, such as Scrum or Kanban, to help teams deliver high-quality software quickly and efficiently.

(b) What is meant by "UX"? Explain the importance of "Usefulness" and "Emotional impact" in UX design.

UX stands for "user experience." It refers to the overall experience of a person using a product, system, or service. This includes the practical, experiential, affective, meaningful, and valuable aspects of human-computer interaction and product ownership.

Usefulness refers to how well a product, system, or service fulfills the needs of the user. It is important in UX design because a product that is not useful to the user is unlikely to be successful.

Emotional impact refers to the emotional response of the user to the product, system, or service. It is important in UX design because products that create positive emotional responses are more likely to be successful. This is because people are more likely to use and recommend products that they have a positive emotional connection to. In addition, products with a strong emotional impact can create a sense of loyalty in users and increase their overall satisfaction with the product.

(c) Discuss the important Agile Principles that guide agile development.

The Agile Manifesto, which is a set of guiding principles for agile development, includes four core values and 12 principles.

The four values are:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation

3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

The 12 principles are:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Q.2 **(a)** Explain the following terms with respect to Extreme Programming:

(i) user stories (ii) pair programming

(i) User stories are short, simple descriptions of a feature or functionality that is needed by the user. In extreme programming (XP), user stories are used to describe the desired functionality from the perspective of the user. They are written in natural language and are typically no more than a few sentences long. User stories are used to guide the development process and are used to create the requirements for the system being developed.

(ii) Pair programming is a software development technique in which two programmers work together at one workstation. One programmer, the "driver," writes code while the other programmer, the "observer," reviews

each line of code as it is typed in. The observer checks for mistakes and suggests improvements. Pair programming is used in extreme programming as a way to improve the quality of the code and to share knowledge between team members. It is also used as a way to reduce the time it takes to complete a task, as two programmers can work on the task simultaneously.

(b) Compare Agile Model with Waterfall Model.

Agile Model	Waterfall Model
Emphasizes flexibility and adaptability	Emphasizes a strict plan and linear progression
Iterative and incremental	Sequential
Emphasizes working software over comprehensive documentation	Emphasizes comprehensive documentation over working software
Customer collaboration is emphasized	Customer requirements are fixed and defined at the beginning of the project
Projects are divided into small phases, with each phase building on the previous one	Projects are divided into distinct phases, with no overlap between phases
Suitable for complex projects with rapidly changing requirements	Suitable for projects with well-defined requirements that are unlikely to change

(c) With the help of a neat diagram, discuss the Funnel Model of Agile UX and discuss its main features.

The funnel model of agile UX is a process for designing and developing user experiences in an agile environment. It consists of five main stages:

1. Discover: In this stage, the team gathers information about the user, the business, and the problem that the product is trying to solve. This includes conducting research, such as user interviews and surveys, to understand the needs and goals of the user.
2. Define: In this stage, the team defines the problem that the product is trying to solve and develops a high-level solution. This includes creating user personas, developing a product roadmap, and defining the overall user experience.
3. Design: In this stage, the team creates detailed designs for the product, including wireframes, prototypes, and user flows. This is also when user testing is typically conducted to gather feedback on the design.
4. Develop: In this stage, the team builds the product, incorporating feedback from the design phase. This may involve multiple iterations as the team works to create the final product.

5. Deliver: In this stage, the final product is released to the user. This may involve a phased rollout or a full release, depending on the needs of the business and the user.

The main features of the funnel model of agile UX are:

- It is iterative: The process is designed to be flexible and adaptable, allowing the team to make changes and improvements as needed.
- It emphasizes customer collaboration: The team works closely with the user throughout the process to ensure that the final product meets their needs and expectations.
- It is focused on delivering value: The goal of the process is to create a product that is useful, usable, and desirable for the user.
- It is data-driven: The process is based on data gathered from research and user testing, allowing the team to make informed decisions about the product.

OR

(d) Write a detailed note on Agile Testing methods.

Agile testing is a software testing practice that follows the principles of agile software development. Agile testing methods are designed to be flexible, iterative, and incremental, allowing testing to be integrated into the development process and aligned with the needs of the user.

There are several key practices that are central to agile testing:

1. Test-driven development (TDD): In this practice, tests are written for a piece of code before the code is actually written. This helps ensure that the code is developed in a way that meets the requirements and passes the tests.
2. Continuous integration (CI): In this practice, code changes are regularly integrated into a shared codebase and automatically built and tested. This helps catch errors and regressions early in the development process.
3. Continuous delivery (CD): In this practice, code changes are automatically built, tested, and deployed to production on a regular basis. This allows the team to deliver new features and updates to the user quickly and efficiently.
4. Automated testing: Agile testing relies heavily on automated testing tools and techniques to help ensure the quality and reliability of the software. This includes unit tests, integration tests, and acceptance tests, among others.
5. Collaboration: Agile testing is collaborative in nature, with testers working closely with developers and other stakeholders to ensure that the software meets the needs of the user.

Overall, agile testing methods are designed to support the rapid development and delivery of high-quality software in an agile environment. By focusing on flexibility, collaboration, and automation, agile testing helps teams deliver value to the user quickly and efficiently.

Q.3 **(a)** What is “Refactoring”?

Refactoring is the process of changing the structure of a piece of code without changing its external behavior. It is a way to improve the design of code by making it more readable, maintainable, and flexible.

Refactoring can involve a wide range of activities, such as renaming variables or methods, extracting code into separate functions or modules, and reorganizing code to follow a particular design pattern. The goal of refactoring is to make the code easier to understand, modify, and test.

Refactoring is an important practice in software development because it helps teams maintain and improve the quality of their code over time. It is especially useful in agile development environments, where code is constantly changing and evolving as new features are added and requirements shift. By regularly refactoring code, teams can ensure that their codebase remains clean and easy to work with, which can help them deliver new features and updates to the user more quickly and efficiently.

(b) What is meant by “Scope” and “Rigor” of a project? Briefly discuss the factors that influence Rigor during agile project development.

Scope refers to the boundaries or limits of a project, including the features and functionality that are included and excluded. In agile development, the scope of a project is typically defined through user stories, which describe the desired functionality from the perspective of the user. The scope of a project can change throughout the development process as new requirements emerge or priorities shift.

Rigor refers to the level of discipline, structure, and formality that is applied to a project. In agile development, the level of rigor can vary depending on the needs and constraints of the project. For example, a project with a high level of rigor may have strict processes and strict deadlines, while a project with a lower level of rigor may have more flexibility and fewer constraints.

There are several factors that can influence the level of rigor during agile project development:

1. The complexity of the project: Projects that are more complex may require a higher level of rigor to ensure that all requirements are properly defined and accounted for.
2. The level of uncertainty: Projects with a high level of uncertainty may require a lower level of rigor to allow for flexibility and adaptability.
3. The level of risk: Projects with a high level of risk may require a higher level of rigor to mitigate potential risks and ensure the success of the project.
4. The team's experience: Teams with more experience may be able to handle a lower level of rigor, while teams with less experience may need a higher level of

rigor to ensure that they are following best practices.

5. The team's size: Larger teams may require a higher level of rigor to ensure that everyone is on the same page and working towards the same goals.

(c) Write a detailed note on "SRP: The Single-Responsibility Principle".

The Single-Responsibility Principle (SRP) is a software design principle that states that a class or module should have only one reason to change. In other words, a class or module should have a single, well-defined responsibility and should be designed in such a way that it can be changed for only one reason.

The goal of SRP is to reduce the complexity and coupling of a system by ensuring that each class or module has a clear and specific purpose. This makes it easier to understand and maintain the code, as well as to add new features or make changes without affecting other parts of the system.

There are several benefits to following the Single-Responsibility Principle:

1. It promotes modularity: By dividing a system into smaller, self-contained units, SRP makes it easier to reuse and test individual components.
2. It reduces complexity: By limiting the scope of each class or module, SRP makes it easier to understand and maintain the code.
3. It improves flexibility: By decoupling components, SRP makes it easier to add or change features without affecting other parts of the system.

To follow the Single-Responsibility Principle, it is important to carefully consider the responsibilities of each class or module and to design them in such a way that they have a single, well-defined purpose. This may involve breaking down larger classes or modules into smaller, more focused components, or combining multiple small components into a larger, more cohesive unit.

OR

Q.3 **(a)** Define a "sprint" in agile development.

In agile development, a sprint is a fixed time period (usually one to four weeks) during which a specific set of work is completed. Sprints are a central part of Scrum, an agile framework that is commonly used in software development.

During a sprint, the development team works to complete a set of user stories, or small, self-contained pieces of functionality that are valuable to the user. The team plans the work for the sprint at the beginning of the sprint and then works to complete the work during the sprint. At the end of the sprint, the team demonstrates the completed work to the rest of the organization and reviews the process to identify ways to improve in the future.

Sprints are designed to be short and focused, with a clear goal and defined scope. They allow the development team to make steady progress on a project, deliver value to the user quickly, and adapt to changing requirements or priorities. Sprints also provide a way for the team to regularly review and reflect on their work, which helps them improve their processes and deliver better results over time.

(b) What are the problems that develop in software systems over time? How does agile design overcome these problems?

Software systems can face a number of problems over time, including:

1. Complexity: As a system grows and evolves, it can become increasingly complex, which can make it difficult to understand and maintain.
2. Fragility: A system that is not designed to handle change can become fragile and prone to breaking when changes are made.
3. Lack of flexibility: A system that is not designed to be flexible may be unable to adapt to new requirements or changing priorities.
4. Lack of reuse: A system that is not designed with reuse in mind may be difficult to extend or modify, which can lead to duplication of effort and increased costs.

Agile design can help overcome these problems by emphasizing flexibility, modularity, and reuse. Agile design practices, such as iterative development, test-driven development, and continuous integration, help ensure that a system is able to adapt to change and remain maintainable over time.

In addition, agile design emphasizes the importance of collaboration, customer feedback, and rapid prototyping. This helps ensure that the system is able to meet the needs of the user and remain valuable over time. By following agile design principles, teams can create software systems that are able to evolve and adapt to changing requirements, which can help them deliver value to the user more quickly and efficiently.

(c) Write a detailed note on "OCP: The Open-Closed Principle".

The Open-Closed Principle (OCP) is a software design principle that states that a class or module should be open for extension but closed for modification. This means that a class or module should be designed in such a way that it can be extended to add new functionality without requiring changes to the existing code.

The goal of OCP is to reduce the complexity and maintenance costs of a system by ensuring that changes to the system can be made in a controlled and predictable way. This is achieved by designing the system in a modular and flexible way, with well-defined interfaces that can be extended without modifying the underlying implementation.

There are several benefits to following the Open-Closed Principle:

1. It promotes reuse: By designing classes and modules in a way that they can be

extended without modification, OCP encourages the reuse of code.

2. It reduces complexity: By minimizing the number of changes that need to be made to the system, OCP helps keep the system simple and easy to understand.
3. It improves maintainability: By limiting the number of changes that need to be made to the system, OCP helps reduce the maintenance costs of the system.

To follow the Open-Closed Principle, it is important to carefully consider the interfaces of classes and modules and to design them in a way that allows them to be extended without requiring changes to the underlying implementation. This may involve using design patterns, such as the template method pattern or the strategy pattern, to provide a flexible and modular architecture.

Q.4 **(a)** What is "acceptance testing"?

Acceptance testing is a type of software testing that is used to determine whether a product, system, or component meets the specified acceptance criteria and is ready for delivery. Acceptance testing is typically performed by the end user or the customer, and is focused on verifying that the system meets the functional and non-functional requirements that have been defined.

Acceptance testing is an important part of the software development process because it helps ensure that the system is fit for its intended purpose. It is typically conducted towards the end of the development process, after unit and integration testing have been completed.

There are several types of acceptance testing, including:

1. User acceptance testing (UAT):
2. System acceptance testing (SAT):
3. Operational acceptance testing (OAT)

(b) Discuss the significance of Prototyping as a UX lifecycle activity. What is a "clickthrough prototype"?

Prototyping is a crucial UX (user experience) lifecycle activity that is used to test and refine the design of a product or system. Prototyping allows the design team to quickly create and test different design concepts, gathering feedback and making iterative improvements as needed.

There are several types of prototypes, ranging from low-fidelity paper prototypes to high-fidelity digital prototypes. The type of prototype used depends on the needs and goals of the project, as well as the stage of the design process.

Prototyping is important because it allows the design team to test and validate their ideas with users early in the process, before investing time and resources in building the final product. This can help the team identify and address any usability issues or problems with the design before they become costly to fix.

A clickthrough prototype is a type of digital prototype that is used to simulate the user experience of navigating through a product or system. A clickthrough prototype typically consists of a series of screens or pages that are linked together, allowing the user to click through the prototype and interact with it as if it were a real product. Clickthrough prototypes are often used to test the flow and usability of a product or system, and to gather feedback from users.

Overall, prototyping is a valuable UX lifecycle activity that allows the design team to test and refine their ideas, gathering feedback and making iterative improvements as needed. It is an important tool for ensuring that the final product meets the needs and expectations of the user.

(c) Discuss and compare the Top-Down and Bottom-up approaches for design using suitable examples.

Top-Down Design	**Bottom-Up Design**
The overall system is decomposed into smaller subsystems, which are then further decomposed into smaller components until the lowest level of detail is reached	The lower-level details are designed first and then combined to create the higher-level components
Starts with a high-level design and works down to the lower-level details	Starts with the lowest-level details and works up to the higher-level system
Fast and efficient	Slower and more iterative
More difficult to make changes to the design once it is complete	Allows for more flexibility and adaptability
Example: software system development	Example: car design

Or

Top-down and bottom-up are two approaches that can be used for design.

Top-down design is a design approach in which the overall system is decomposed into smaller subsystems, which are then further decomposed into smaller components until the lowest level of detail is reached. This approach starts with a high-level design and works down to the lower-level details, breaking the system into smaller and smaller pieces as the design progresses.

An example of top-down design is the development of a software system. In this case, the system is divided into modules or components, which are further divided into classes and functions.

Bottom-up design is a design approach in which the lower-level details are designed first and then combined to create the higher-level components. This approach starts with the lowest-level details and works up to the higher-level system, building the system from the ground up.

An example of bottom-up design is the design of a car. In this case, the individual parts, such as the engine, transmission, and suspension, are designed first, and then these parts are combined to create the car.

Top-down and bottom-up approaches have different advantages and disadvantages. Top-down design is often faster and more efficient, as it allows the designer to focus on the overall system and then drill down into the details as needed. However, it can be more difficult to make changes to the design once it is complete, as it may require changes to be made at multiple levels of the design. Bottom-up design is slower and more iterative, but it allows for more flexibility and adaptability, as it is easier to make changes to the design at the lower levels.

OR

Q.4 **(a)** What is a physical mockup? How does it help in design?

A physical mockup is a physical representation of a product or system that is used to test and refine the design. Physical mockups can be made using a variety of materials, such as cardboard, foam core, or 3D printing, and are used to simulate the size, shape, and feel of the final product.

Physical mockups are often used in the design process to test and refine the ergonomics and usability of a product. They can be used to test the fit and comfort of a product, as well as to evaluate the layout and accessibility of controls and features. Physical mockups can also be used to gather feedback from users, which can help the design team identify and address any issues or problems with the design.

Overall, physical mockups are a valuable tool in the design process, as they allow the design team to test and refine the design in a realistic way. They can help ensure that the final product is user-friendly, comfortable, and functional, which can help improve the user experience.

(b) Discuss the following concepts with respect to Generative design: (i) ideation (ii) critiquing

Generative design is a design approach that uses computer algorithms to generate and explore a wide range of design options, based on a set of constraints and requirements. Generative design can be used to create designs for a variety of products and systems, including architecture, engineering, and industrial design.

Ideation is the process of generating and exploring ideas for a design. In the context of generative design, ideation involves using computer algorithms to generate a large number of design options based on the specified constraints and requirements. These design options can then be evaluated and refined through a process of critiquing.

Critiquing is the process of evaluating and reviewing a design to identify its strengths and weaknesses and to suggest improvements. In the context of generative design, critiquing involves reviewing the design options generated by the computer algorithms and selecting the best ones for further refinement. Critiquing can be done by a team of designers, or it can be automated through the use of machine learning algorithms.

Overall, ideation and critiquing are important components of the generative design process. They allow the design team to explore a wide range of design options and to select the best ones for further refinement, which can help ensure that the final design meets the needs and requirements of the project.

(c) Explain why a UX Design team should have people with diverse skills and backgrounds. Briefly discuss a real-world example where such a team can have a positive impact on the design of a product.

A UX (user experience) design team should have people with diverse skills and backgrounds because this can help the team create a more well-rounded and effective design.

Having people with different skills and backgrounds can bring a variety of perspectives and ideas to the design process, which can help the team identify and solve problems in creative and innovative ways. For example, a team with a mix of designers, researchers, and developers may be able to approach problems from different angles and come up with more holistic and effective solutions.

In addition, having a diverse team can help ensure that the design meets the needs and expectations of a diverse user base. For example, a team with members from different cultural backgrounds may be better able to understand and design for the needs of users from different cultures.

One real-world example where a diverse UX design team can have a positive impact on the design of a product is the design of a healthcare app. In this case, a team with a mix of healthcare professionals, designers, and researchers may be able to create an app that is more effective and user-friendly for patients and healthcare providers. The team may be able to identify and address the specific needs and challenges of this user group, such as the need for privacy and security, the need for clear and concise information, and the need for easy navigation.

Overall, having a UX design team with diverse skills and backgrounds can help create more well-rounded and effective designs that are better able to meet the

needs and expectations of a diverse user base.

Q.5 **(a)** What are "quantitative" and "qualitative" UX evaluation data?

Quantitative and qualitative data are types of data that are used to evaluate the user experience (UX) of a product or system.

Quantitative data is data that can be measured and expressed in numerical form. This type of data is often used to track specific metrics, such as the number of clicks on a button or the time it takes to complete a task. Quantitative data is useful because it allows the UX team to analyze trends and patterns and to make data-driven decisions.

Qualitative data is data that cannot be measured and expressed in numerical form. This type of data is often gathered through methods such as interviews, focus groups, and surveys, and consists of words, opinions, and observations. Qualitative data is useful because it allows the UX team to understand the user's experience in more depth and to identify issues and problems that may not be apparent from quantitative data alone.

(b) How do Agile UX and Agile Software Engineering work together?

Agile UX (user experience) and Agile software engineering work together by following the principles of Agile development. Agile development is a flexible, iterative approach to software development that emphasizes collaboration, customer feedback, and rapid prototyping.

In Agile software engineering, the development team works in short sprints, delivering small increments of functionality to the customer on a regular basis. This allows the team to quickly gather feedback and make iterative improvements to the software.

Agile UX follows a similar approach, with the UX team working closely with the development team to design and test the user experience of the software. The UX team may use techniques such as prototyping and user testing to gather feedback and refine the design, and may work in parallel with the development team to ensure that the user experience is integrated into the software as it is being developed.

Overall, Agile UX and Agile software engineering work together by following the principles of Agile development, with the UX team and the development team collaborating closely to deliver high-quality software that meets the needs and expectations of the user.

(c) What is Empirical UX evaluation? With the help of a suitable example discuss the setting of Goals and Metrics for Empirical UX Evaluation.

Empirical UX (user experience) evaluation is a type of evaluation that is based on data and observations gathered from users. Empirical evaluation involves collecting data on how users interact with a product or system, and using this data to understand and improve the user experience.

To set goals and metrics for empirical UX evaluation, it is important to consider the specific goals and objectives of the evaluation. For example, if the goal of the evaluation is to improve the usability of a website, the goals and metrics might include things like the time it takes users to complete a task, the number of errors they make, and their overall satisfaction with the website.

To set these goals and metrics, the UX team may first need to identify the key user tasks that need to be supported by the website. They may then define specific goals and metrics for each task, such as the time it should take for users to complete the task, or the percentage of users who are able to complete the task successfully.

For example, consider a website that is designed to allow users to purchase tickets to a sporting event. The UX team might set the following goals and metrics for empirical UX evaluation:

- Goal: Reduce the time it takes users to purchase tickets
- Metric: Average time to purchase tickets
- Goal: Increase the percentage of users who are able to successfully purchase tickets
- Metric: Percentage of users who successfully complete the purchase process

Overall, empirical UX evaluation is an important tool for understanding and improving the user experience of a product or system. By setting clear goals and metrics, the UX team can focus their efforts on the areas that are most important to the user and measure their progress over time.

OR

Q.5 **(a)** What is a “benchmark” task?

A benchmark task is a standard or reference task that is used to evaluate the performance of a product or system. Benchmark tasks are typically chosen to be representative of the types of tasks that users are likely to perform with the product or system, and are used to measure the performance of the product or system in a consistent and objective way.

There are several types of benchmark tasks, including:

- Objective tasks: These tasks are designed to measure specific performance metrics, such as the time it takes to complete a task or the number of errors made.
- Subjective tasks: These tasks are designed to measure the user's subjective experience of using the product or system, such as their overall satisfaction or

perceived ease of use.

- Comparative tasks: These tasks are used to compare the performance of different products or systems, either within the same category or across different categories.

(b) Compare "Formative Evaluation" versus "Summative Evaluation".

Formative Evaluation	Summative Evaluation
Ongoing process of evaluating and refining a product or system as it is being developed	Comprehensive evaluation of a product or system that is conducted after it has been completed
Focused on improving the product or system	Focused on evaluating the overall effectiveness and value of the product or system
Typically informal	Typically more formal and structured
May involve techniques such as prototyping, user testing, and expert reviews	May involve techniques such as user testing, surveys, and focus groups
Example: usability testing during the development of a website	Example: user satisfaction survey after the launch of a new mobile app

(c) Discuss the following data collection methods for Analytic UX evaluation:

(i) Design walk-through (ii) Expert UX Inspection.

Analytic UX (user experience) evaluation is a type of evaluation that uses data and analysis to understand and improve the user experience of a product or system. There are several data collection methods that can be used for analytic UX evaluation, including design walk-throughs and expert UX inspections.

Design walk-throughs are a method of evaluating the usability of a product or system by walking through the design and identifying potential issues and problems. Design walk-throughs are typically conducted by a team of UX experts, who review the design and provide feedback on areas that may be confusing, difficult to use, or in need of improvement.

Expert UX inspections are another method of evaluating the usability of a product or system. In this method, a team of UX experts reviews the design of the product or system and identifies potential issues and problems using a set of established usability guidelines or heuristics. Expert UX inspections are typically more formal and structured than design walk-throughs, and may involve the use of checklists or scoring systems to assess the design.

Overall, design walk-throughs and expert UX inspections are both useful methods for collecting data and identifying issues and problems with the usability of a product or system. These methods can help the UX team understand the user's experience and identify areas for improvement, which can ultimately lead to a more effective and user-friendly design.

****

Seat No.: _____ Enrolment No._____

GUJARAT TECHNOLOGICAL UNIVERSITY
BE - SEMESTER–VII (NEW) EXAMINATION – SUMMER 2022
Subject Code:3171610

Date:08/06/20

22 Subject Name: Agile Development and UI/UX design
Time: 02:30 PM TO 05:00 PM **Total Marks: 70**

Q.1 **(a)** What is Agile Design? **03**

Agile design is a design approach that is based on the principles of Agile development. Agile development is a flexible, iterative approach to software development that emphasizes collaboration, customer feedback, and rapid prototyping.

Agile design is focused on creating designs that are flexible and adaptable, and that can be quickly modified and refined based on customer feedback. It is based on the idea that the design process should be collaborative and iterative, with the design team working closely with the development team and the customer to create a high-quality product that meets the needs and expectations of the user.

Agile design involves a number of practices, such as:

- Continuous collaboration: The design team works closely with the development team and the customer throughout the design process.
- Rapid prototyping: The design team creates prototypes of the product or system early in the design process, and gathers feedback from the development team and the customer to refine the design.
- Iterative design: The design team makes frequent, incremental improvements to the design based on customer feedback.

(b) How do we know how whether the design of a software system is Good? Explain in detail. **04**

There are several factors that can help determine whether the design of a software system is good. Some of these factors include:

- Usability: A good design should be easy for users to understand and use. This may involve things like clear and concise instructions, intuitive navigation, and logical organization of information.

- Functionality: A good design should provide the functionality that the user needs and expects. This may involve things like providing the right set of features and tools, and ensuring that the system is reliable and performs well.
- Aesthetics: A good design should be visually appealing and consistent with the overall look and feel of the product or system. This may involve things like using consistent colors and fonts, and ensuring that the layout is balanced and harmonious.
- User experience: A good design should provide a positive and enjoyable user experience. This may involve things like making the system easy to use, providing feedback to the user, and creating a sense of satisfaction and accomplishment for the user.

To determine whether the design of a software system is good, it is often necessary to conduct user testing and gather feedback from users. This can help identify any issues or problems with the design, and provide valuable insights into how the design can be improved.

(c) List out agile design principles. Explain them in detail. **07**

The 12 principles are:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Q.2 (a) What kind of projects is suitable for the Agile Methodology? **03**

Agile development is a flexible, iterative approach to software development that is particularly well-suited to projects where the requirements may change frequently or where the user's needs are not well-defined upfront. Some characteristics of projects that are well-suited to the Agile methodology include:

- Complexity: Agile development is well-suited to complex projects where it may be difficult to define all of the requirements upfront.
- Rapid delivery: Agile development emphasizes rapid delivery of small increments of functionality, which can be particularly useful for projects where time-to-market is critical.
- Collaboration: Agile development emphasizes collaboration between the development team and the customer, which can be particularly useful for projects where close collaboration is needed to ensure that the product meets the user's needs.
- Adaptability: Agile development is flexible and adaptable, which can be useful for projects where the requirements may change frequently or where it is important to be able to respond quickly to changes in the market or business environment.

(b) What are different types of Agile Methodology? **04**

There are several different types of Agile methodology, including:

1. Scrum: Scrum is a popular Agile methodology that is based on the idea of small, cross-functional teams working collaboratively to deliver small increments of functionality on a regular basis.
2. Extreme Programming (XP): Extreme Programming (XP) is an Agile methodology that emphasizes rapid delivery, continuous testing and integration, and close collaboration between the development team and the customer.
3. Lean: Lean is an Agile methodology that is based on the principles of continuous improvement and waste reduction. It emphasizes the importance of delivering value to the customer as quickly as possible, and of eliminating unnecessary activities and processes that do not add value.
4. Crystal: Crystal is a family of Agile methodologies that are designed to be customized to the needs and constraints of a specific project or organization. Crystal methodologies emphasize the importance of communication, flexibility, and continuous improvement.

(c) Explain the principles of extreme programming. **07**

Extreme Programming (XP) is an Agile software development methodology that emphasizes rapid delivery, continuous testing and integration, and close collaboration between the development team and the customer. The principles of XP are designed to help teams deliver high-quality software that meets the needs and expectations of the user.

The principles of XP include:

1. Communication: XP emphasizes the importance of effective communication between the development team and the customer. This may involve things like frequent face-to-face meetings, continuous feedback and iteration, and open and honest communication.
2. Simplicity: XP emphasizes the importance of keeping the design of the software simple and focused on the core functionality that is needed to meet the user's needs.
3. Feedback: XP emphasizes the importance of continuous feedback and iteration, and encourages the development team to gather feedback from the customer and incorporate it into the design of the software.
4. Courage: XP encourages the development team to be courageous in making decisions and taking risks, and to be willing to learn from their mistakes.
5. Respect: XP emphasizes the importance of treating everyone on the team with respect and valuing their contributions.

Overall, the principles of XP are designed to help teams deliver high-quality software that meets the needs and expectations of the user, and to encourage continuous improvement and innovation.

OR

(c) Discuss How is Agile Methodology different than Traditional Waterfall process? **07**

Agile methodology and the traditional Waterfall process are two different approaches to software development that have distinct characteristics and approaches. Some of the key differences between Agile and the Waterfall process include:

- Iterative vs. linear: Agile is an iterative approach to development, which means that it involves frequent iteration and refinement of the product or system. The Waterfall process is a linear approach, which means that it involves completing each phase of the development process before moving on to the next.
- Flexibility vs. predictability: Agile is designed to be flexible and adaptable, which means that it is well-suited to projects where the requirements may change frequently or where the user's needs are

not well-defined upfront. The Waterfall process is more predictable, which means that it is better suited to projects where the requirements are well-defined and are not expected to change significantly.

- Collaboration vs. documentation: Agile emphasizes collaboration and communication between the development team and the customer, and is focused on delivering working software as quickly as possible. The Waterfall process emphasizes documentation and is focused on completing each phase of the development process before moving on to the next.
- Incremental delivery vs. big-bang delivery: Agile emphasizes incremental delivery of small increments of functionality, while the Waterfall process is focused on delivering the entire product or system in one big "bang" at the end of the development process.

Overall, Agile and the Waterfall process are two different approaches to software development that are suited to different types of projects and environments. Agile is a flexible, iterative approach that is well-suited to projects where the requirements may change frequently, while the Waterfall process is a more predictable, linear approach that is better suited to projects where the requirements are well-defined and are not expected to change significantly.

Q.3 (a) Differentiate between Agile and Scrum. **03**

Agile	**Scrum**
A flexible, iterative approach to software development	A specific Agile methodology that is based on the idea of small, cross-functional teams working collaboratively to deliver small increments of functionality on a regular basis
Emphasizes collaboration, continuous improvement, and flexibility	Emphasizes small, self-organizing teams, frequent delivery of working software, and a focus on continuous improvement
May involve a variety of techniques and practices, such as prototyping, user testing, and pair programming	Involves specific roles (e.g. Product Owner, Scrum Master, Development Team), events (e.g. Sprint Planning, Daily Scrum, Sprint Review), and artifacts (e.g. Product Backlog, Sprint Backlog)
Example: Agile development can be used to develop a mobile app	Example: Scrum can be used to develop a mobile app using Agile principles and practices

(b) Briefly explain principles of agile methods. **04**

Agile methods are a set of principles and practices for software development that are designed to be flexible, adaptable, and responsive to change. The principles of Agile methods are based on the Agile Manifesto, which is a set of values and principles that guide the development of Agile software.

The principles of Agile methods include:

1. Prioritizing customer satisfaction: Agile methods prioritize the needs and satisfaction of the customer, and focus on delivering value to the customer as quickly as possible.
2. Welcoming change: Agile methods embrace change and view it as an opportunity to improve the product or system.
3. Delivering working software frequently: Agile methods emphasize the importance of delivering working software frequently, and encourage the development team to deliver small increments of functionality on a regular basis.
4. Working collaboratively: Agile methods emphasize collaboration and communication between the development team and the customer, and encourage the development team to work closely with the customer to ensure that the product meets the user's needs.
5. Measuring progress by working software: Agile methods view working software as the primary measure of progress, and encourage the development team to focus on delivering value to the customer through working software.

(c) Explain Agile Testing? What are the principles of Agile Testing? **07**

Agile testing is a testing approach that is based on the principles of Agile development. Agile development is a flexible, iterative approach to software development that emphasizes collaboration, customer feedback, and rapid prototyping.

Agile testing is focused on testing software in small increments, and on gathering feedback from the customer and the development team throughout the testing process. It is based on the idea that the testing process should be collaborative and iterative, with the testing team working closely with the development team and the customer to identify and fix issues as they arise.

The principles of Agile testing include:

1. Continuous testing: Agile testing emphasizes the importance of continuous testing throughout the development process, and encourages the testing team to test the software frequently and in small increments.

2. Collaboration: Agile testing emphasizes the importance of collaboration and communication between the testing team, the development team, and the customer.
3. Continuous feedback: Agile testing encourages the testing team to gather feedback from the customer and the development team throughout the testing process, and to use this feedback to refine and improve the testing process.
4. Adaptability: Agile testing is designed to be flexible and adaptable, and encourages the testing team to be responsive to changes in the requirements or the user's needs.

Overall, Agile testing is an effective approach for testing software in a flexible and collaborative manner, and for gathering feedback from the customer and the development team to ensure that the software meets the user's needs and expectations.

OR

Q.3 (a) What is Refactoring? **03**

Refactoring is the process of changing the structure or design of existing code in order to improve its quality, maintainability, or readability, without changing its functionality. Refactoring is an important technique in Agile development, as it allows teams to continuously improve the design of their codebase and make it more maintainable over time.

Refactoring can involve things like:

- Renaming variables or methods to make the code more readable and easier to understand
- Extracting code into separate methods or classes to make it more modular and reusable
- Removing duplication or unnecessary code to make the codebase simpler and easier to maintain
- Restructuring the code to make it more logical or easier to understand

(b) Discuss Agile Life cycle Processes. **04**

Agile development is a flexible, iterative approach to software development that emphasizes collaboration, continuous improvement, and rapid prototyping. The Agile life cycle is designed to be flexible and adaptable, and typically involves the following processes:

1. Planning: The planning process involves determining the goals and objectives of the project, and developing a plan for how to achieve these goals.
2. Development: During the development process, the development team works on creating the software. This may involve things like

writing code, testing the software, and gathering feedback from the customer.
3. Testing: The testing process involves verifying that the software meets the requirements and functions as expected. This may involve things like conducting unit tests, integration tests, and acceptance tests.
4. Deployment: The deployment process involves releasing the software to users and making it available for use.
5. Maintenance: The maintenance process involves supporting and maintaining the software after it has been deployed. This may involve things like fixing bugs, adding new features, and updating the software to meet changing user needs.

(c) Explain the Funnel Model of Agile UX. **07**

The Funnel Model of Agile UX is a framework for designing and developing user experiences that is based on the principles of Agile development. The Funnel Model is designed to be flexible and iterative, and emphasizes the importance of continuous collaboration, prototyping, and user testing in the design process.

The Funnel Model consists of four main stages:

1. Research: The research stage involves gathering data and insights about the user, the market, and the business needs. This may involve things like conducting user interviews, focus groups, and market research.
2. Design: The design stage involves creating and testing prototypes of the user experience. This may involve things like sketching, wireframing, and creating high-fidelity mockups.
3. Development: The development stage involves building and testing the user experience. This may involve things like writing code, conducting user testing, and gathering feedback from the development team and the customer.
4. Deployment: The deployment stage involves releasing the user experience to users and making it available for use.

Overall, the Funnel Model of Agile UX is a flexible and iterative approach to designing and developing user experiences that is based on the principles of Agile development. It emphasizes the importance of continuous collaboration, prototyping, and user testing in the design process, and is designed to help teams deliver high-quality user experiences that meet the needs and expectations of the user.

Q.4 (a) Discuss Empirical UX evaluation. **03**

Empirical UX evaluation is a process for evaluating the effectiveness of a user experience that is based on empirical data and observations. Empirical UX evaluation involves collecting data about how users interact

with a product or system, and using this data to identify areas for improvement and make informed design decisions.

Empirical UX evaluation typically involves the following steps:

1. Setting goals and metrics
2. Collecting data
3. Analyzing data
4. Making design decisions

(b) How to convert a user story to a task? Explain with example. **04**

A user story is a brief description of a feature or functionality that is desired by a user. It is typically written in the form of "As a [user], I want [some feature] so that [benefit]". User stories are often used in Agile development as a way to capture the requirements and needs of the user.

To convert a user story into a task, the development team can break down the user story into smaller, more specific tasks that can be completed by the development team. These tasks should be specific, actionable, and achievable within a reasonable amount of time.

For example, consider the following user story:

"As a customer, I want to be able to search for products on the website so that I can find what I'm looking for more easily."

To convert this user story into tasks, the development team might break it down into the following tasks:

- Design the search interface
- Implement the search functionality
- Test the search functionality
- Integrate the search functionality into the website

(c) Differentiate between Bottom-up versus Top-down Design. **07**

Bottom-up Design	Top-down Design
Starts with smaller components and builds up to larger ones	Starts with the larger, overall system and breaks it down into smaller components
Focuses on the details and implementation of the system	Focuses on the overall structure and functionality of the system
Good for understanding how individual components work and fit together	Good for understanding the overall functionality and architecture of the system

Example: Designing the individual parts of a car (e.g. engine, transmission, brakes)	Example: Designing the overall layout and functionality of a website

OR

Q.4 (a) What are the pros and cons of Agile Methodology? **03**

Pros of Agile Methodology:

1. Flexibility and adaptability: Agile allows teams to respond to changing requirements and priorities, and to adapt their work as needed.
2. Collaboration: Agile emphasizes the importance of collaboration and communication between team members, and encourages teamwork and collaboration.
3. Continuous feedback: Agile encourages teams to gather feedback from the customer and the development team throughout the development process, and to use this feedback to continuously improve the product.
4. Rapid prototyping: Agile allows teams to quickly prototype and test ideas, and to gather feedback from users early in the development process.

Cons of Agile Methodology:

1. Complexity: Agile can be complex, and may require more effort to plan and coordinate work across the team.
2. Lack of clear plan: Agile may lack the clear, upfront plan that is typical of more traditional development approaches, which can make it difficult to predict the final scope and schedule of the project.
3. Communication challenges: Agile can be challenging for teams that are distributed or have poor communication, as it requires frequent and effective communication between team members.
4. Limited documentation: Agile may result in less documentation than other development approaches, which can make it harder to understand the overall design and functionality of the product.

(b) List out Principles of Testing. **04**

There are several principles that are important in the practice of testing:

1. Test early and often: Testing early and often helps to identify issues and defects early in the development process, which can be less expensive to fix than issues that are found later.
2. Test at the appropriate level: It is important to test at the appropriate level of abstraction, depending on the goals of the testing. For example, unit tests focus on individual components or

functions, while integration tests focus on how different components work together.
3. Test the whole system: Testing should cover the whole system, including the user interface, business logic, and database.
4. Test for quality: Testing should be focused on ensuring that the product meets the quality standards and requirements of the customer.
5. Test with real data: Testing should use real data whenever possible, as this can help to uncover issues that may not be apparent with synthetic or artificially generated data.
6. Test with a variety of inputs: Testing should cover a variety of different inputs, including valid, invalid, edge cases, and boundary conditions.
7. Test to break: Testing should be focused on trying to break the system, in order to identify issues and defects that may not be apparent under normal use.

(c) Explain Ideation in Generative design. **07**

Ideation is the process of generating and developing ideas. In the context of generative design, ideation refers to the process of generating and exploring a wide range of design ideas in order to find the best solution to a design problem.

Ideation can take many forms, and may involve things like brainstorming sessions, sketching and prototyping, and using design tools and techniques like mind maps and storyboards. The goal of ideation is to generate a diverse range of ideas that can be explored and evaluated in order to find the best solution to the design problem.

Ideation is an important part of the generative design process, as it helps teams to explore a wide range of potential solutions and to find the one that best meets the needs and expectations of the user. By generating and evaluating a diverse range of ideas, teams can arrive at innovative and creative solutions that may not have been apparent through more traditional design processes.

Q.5 (a) Difference between extreme programming and scrum? **03**

Extreme Programming (XP)	**Scrum**
Emphasizes the importance of communication and collaboration between team members	Emphasizes the importance of iterative development and incremental delivery
Uses a set of specific practices and techniques, such as pair	Uses a flexible framework that can be adapted to the needs of the team

programming and continuous integration	
Uses a fixed set of roles, with each team member responsible for a specific aspect of the development process	Uses a flexible set of roles, with team members responsible for completing tasks and delivering value
Uses a fixed set of deliverables, including user stories and acceptance tests	Uses a flexible set of deliverables, including a backlog of user stories and a set of incremental releases

(b) How Does UX Differ From Other Design Disciplines? **04**

User experience (UX) design is a discipline that focuses on creating products and systems that are easy to use, effective, and satisfying for the user. UX design involves considering a wide range of factors that can impact the user experience, including usability, accessibility, aesthetics, and emotional impact.

UX design differs from other design disciplines in several key ways:

1. Focus on the user: UX design is focused on creating products and systems that meet the needs and expectations of the user. Other design disciplines, such as graphic design or industrial design, may focus more on aesthetics or functionality.
2. Multidisciplinary: UX design involves a range of disciplines, including psychology, sociology, human-computer interaction, and design. Other design disciplines may be more focused on a specific aspect of design, such as visual design or engineering.
3. Iterative process: UX design is often an iterative process that involves prototyping, testing, and refining the design based on user feedback. Other design disciplines may be more focused on creating a final, polished product.
4. Holistic approach: UX design takes a holistic approach to design, considering the entire experience of the user, from beginning to end. Other design disciplines may focus on specific aspects of the product or system.

(c) Explain Critiquing in Generative design. **07**

Critiquing is the process of evaluating and reviewing a design in order to identify its strengths and weaknesses, and to suggest improvements. In the context of generative design, critiquing is an important part of the design process, as it helps teams to refine and improve their designs.

Critiquing can take many forms, and may involve things like conducting user testing, gathering feedback from stakeholders, and using design

tools and techniques like heuristic evaluation. The goal of critiquing is to identify areas of the design that are working well, and areas that may need improvement, in order to create the best possible solution to the design problem.

Critiquing is an important part of the generative design process, as it helps teams to evaluate and improve their designs, and to ensure that they meet the needs and expectations of the user. By conducting regular reviews and evaluations of the design, teams can arrive at the best possible solution, and create a high-quality product or system.

OR

Q.5 (a) Discuss the nature of UX design. **03**

User experience (UX) design is a discipline that focuses on creating products and systems that are easy to use, effective, and satisfying for the user. UX design involves considering a wide range of factors that can impact the user experience, including usability, accessibility, aesthetics, and emotional impact.

The nature of UX design is multifaceted, and involves a range of activities and skills. Some key aspects of UX design include:

1. User-centered design: UX design is focused on creating products and systems that meet the needs and expectations of the user. This involves understanding the user's goals, motivations, and behaviors, and designing the product or system to support these.
2. Interaction design: UX design involves creating interfaces and interactions that are easy to use and understand, and that support the user's goals. This may involve designing the layout and navigation of a website, or the controls and functions of a mobile app.
3. Visual design: UX design often involves creating visual designs that are aesthetically pleasing and effective at communicating information. This may involve creating wireframes, mockups, and prototypes to explore different design ideas.
4. Research and evaluation: UX design involves gathering data about the user and the product or system, and using this data to inform design decisions. This may involve conducting user research, testing prototypes, and gathering feedback from users.

Overall, the nature of UX design is multifaceted, and involves a range of activities and skills that are focused on creating products and systems that are easy to use, effective, and satisfying for the user.

(b) What are the crucial Agile Matrices? **04**

In the context of Agile development, there are several matrices that can be used to help teams plan, track, and manage their work:

1. Product backlog: A product backlog is a list of features, functionalities, and requirements that are desired for a product or system. It is typically organized in order of priority, with the most important items at the top. The product backlog is a key tool in Agile development, as it provides a clear overview of the work that needs to be done, and helps teams to prioritize their efforts.
2. Sprint backlog: A sprint backlog is a list of tasks that are planned for a specific sprint, or iteration. It includes the tasks that are needed to complete the work in the sprint, as well as any additional tasks that may be needed to support the work. The sprint backlog is a key tool in Agile development, as it helps teams to understand the work that needs to be done in the sprint, and to track progress towards completing it.
3. Burn-down chart: A burn-down chart is a graphical representation of the work remaining in a sprint or project. It plots the remaining work over time, and shows how much work has been completed and how much remains. Burn-down charts are a key tool in Agile development, as they help teams to track progress, identify issues, and adjust their work as needed.

Overall, these matrices are crucial tools in Agile development, and can help teams to plan, track, and manage their work in a flexible and responsive way.

(c) Explain Data collection methods and Techniques in UX Evaluation. **07**

Data collection methods and techniques are methods and techniques used to gather data about the user and the product or system being evaluated. There are many different methods and techniques that can be used in UX evaluation, including:

1. Interviews: Interviews are a common method of data collection in UX evaluation. They can be conducted in person, over the phone, or online, and involve talking with users about their experiences and needs. Interviews can be structured, with a set of predetermined questions, or unstructured, with a more open-ended conversation.
2. Surveys: Surveys are a common method of data collection in UX evaluation. They involve presenting users with a set of questions and asking them to provide responses. Surveys can be conducted online or in person, and can be used to gather data about a wide range of topics, including user needs, preferences, and behaviors.
3. Usability testing: Usability testing is a method of data collection that involves observing users as they perform specific tasks with a product or system. Usability testing can help to identify issues and

defects in the design, and to gather data about how users interact with the product or system.

4. A/B testing: A/B testing is a method of data collection that involves comparing two or more versions of a product or system to see which performs better. A/B testing can be used to gather data about user preferences and behaviors, and to identify the best design solution.
5. Analytics: Analytics tools can be used to gather data about user behavior and usage patterns. This can include things like tracking page views, clicks, and time spent on a website, or tracking app usage and engagement.

Overall, data collection methods and techniques are an important part of UX evaluation, and can help teams to gather data about the user and the product or system in order to inform design decisions and improve the user experience.

****

1