|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| <hitle> | Agile | <chare> | 1 | <pext> | Agile is a popular set of methods and practices that majorly focuses  on interactive development.  Thanks to self-organizing collaborations  between cross-functional teams, the requirements  from their customers and potential solutions are obtained. | </end> |
| <hitle> | Agile vs Waterfall | <chare> | 2 | <pext> | 1. Agile is a continuous iteration lifecycle model used for developing  and testing software.  1-1. Waterfall is a linear sequential lifecycle model for developing  and testing software.  2. Agile is a flexible method to build a software.  2-1. Waterfall is a rigid and structured method  for software development  3. Agile is highly collaborative, thus yielding quality output.  3-1. Waterfall follows a rigid sequence of steps,  hence team collaboration is difficult  4. In Agile, the process of development is divided into sprints.  4-1. In Waterfall, the process is broken down into several phases.  5. In Agile, changes can be made  even after the initial planning is completed.  5-1. In Waterfall, Once the project development has started,  development requirements cannot be changed.  6. In Agile, Development is a collection of many projects.  6-1. In Waterfall, Development is completed  as a simple project or deliverable.  7. In Agile, Testing is performed in the same iteration.  7-1. In Waterfall, The testing phase follows the build phase | </end> |
| <hitle> | Waterfall | <chare> | 2 | <pext> | In "The Waterfall" approach,  the whole process of software development is divided  into separate phases.  In this Waterfall model, typically, the outcome of one phase acts  as the input for the next phase sequentially.  The sequential phases in Waterfall model are:  1. Requirement Gathering and analysis –  All possible requirements of the system to be developed  are captured in this phase and documented  in a requirement specification document.  2. System Design –  The requirement specifications from first phase are studied  in this phase and the system design is prepared.  This system design helps in specifying hardware  and system requirements and helps in defining  the overall system architecture.  3. Implementation − With inputs from the system design,  the system is first developed in small programs called units,  which are integrated in the next phase.  Each unit is developed and tested for its functionality,  which is referred to as Unit Testing.  4. Integration and Testing –  All the units developed in the implementation phase  are integrated into a system after testing of each unit.  Post integration the entire system is tested  for any faults and failures.  5. Deployment of system –  Once the functional and non-functional testing is done;  the product is deployed in the customer environment  or released into the market.  6. Maintenance –  There are some issues which come up in the client environment.  To fix those issues, patches are released.  Also to enhance the product some better versions are released.  Maintenance is done to deliver these changes in the customer environment.  All these phases are cascaded to each other in which progress  is seen as flowing steadily downwards (like a waterfall)  through the phases.  The next phase is started only after the defined set of goals  are achieved for previous phase and it is signed off,  so the name "Waterfall Model".  In this model, phases do not overlap. | </end> |
| <hitle> | Waterfall pros/cons | <chare> | 2 | <pext> | Waterfall pros:  1. Simple and easy to understand and use.  2. Easy to manage due to the rigidity of the model.  Each phase has specific deliverables and a review process.  3. Phases are processed and completed one at a time.  4. Works well for smaller projects where requirements  are very well understood.  5. Clearly defined stages.  6. Well understood milestones.  7. Easy to arrange tasks.  8. Process and results are well documented.  Waterfall cons:  1. No working software is produced until late during the life cycle.  2. High amounts of risk and uncertainty.  3. Not a good model for complex and object-oriented projects.  4. Poor model for long and ongoing projects.  5. Not suitable for the projects where requirements  are at a moderate to high risk of changing.  So, risk and uncertainty is high with this process model.  6. It is difficult to measure progress within stages.  7. Cannot accommodate changing requirements.  8. Adjusting scope during the life cycle can end a project.  9. Integration is done as a "big-bang.  At the very end, which doesn't allow identifying any technological  or business bottleneck or challenges early. | </end> |
| <hitle> | Spiral Model pros/cons | <chare> | 2 | <pext> | Spiral model is used for risk management that combines  the iterative development process model with elements  of the Waterfall model. | </end> |
| <hitle> | Spiral Model pros/cons | <chare> | 2 | <pext> | Spiral Model pros:  1. Changing requirements can be accommodated.  2. Allows extensive use of prototypes.  3. Requirements can be captured more accurately.  4. Users see the system early.  5. Development can be divided into smaller parts  and the risky parts can be developed earlier  which helps in better risk management.  Spiral Model cons:  1. Management is more complex.  2. End of the project may not be known early.  3. Not suitable for small or low risk projects  and could be expensive for small projects.  4. Process is complex  5. Spiral may go on indefinitely.  6. Large number of intermediate stages requires  excessive documentation. | </end> |
| <hitle> | Agile process principles | <chare> | 2 | <pext> | What are some important parts of the Agile process  The different principles of Agile are:  1. Customer Satisfaction  The needs of the customer need to be satisfied with the early  and quick delivery of the product.  2. Welcome Changes  Changing needs need to be addressed, even if they are brought  in late in the development process.  3. Deliver Frequently  Ensure frequent product delivery within a short timeframe,  thanks to a clear cut understanding of the product.  4. Work Together  Developers and other team members need to work together  through in collaboration with the course of the project.  5. Motivated Team  Motivated and driven individuals who are trusted to get the job done  can ensure the success of the project.  6. Face-to-Face Conversations  Having face-to-face conversations is proven  as the most efficient method for communication.  7. Working Software  This represents the primary measure of progress.  8. Constant Pace  With a constant pace, the agile process enables optimum sustainable development.  9. Good Design  By focusing on technical excellence and great design,  the agility of the project can be improved drastically.  10. Simplicity  The amount of time spent not doing work  has to be minimized and simplified.  11. Self-Organized  Self-organized teams provide the best architectures,  requirements, and designs.  12. Reflect and Adjust  The effectiveness of the Agile process can be improved  by regularly reflecting on it. | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | Explain Iterative and Incremental Development in Agile.  To understand how these processes work in Agile, we’ll have to talk about each one of them individually.  Incremental Development  In this form of development, the process is divided into small, workable increments. Each succeeding increment builds on the work completed in the previous increment. Over time, functionalities are added based on everything already created.  Iterative Development  This involves the development of a system that follows repeated cycles or iterations. Changes are made based on results from the most recent iteration. This enables the project to evolve over time.  Agile: Incremental+Iterative  The agile process involves the consideration and creation of a working product in an iteration, which is a part of the final product. Each successive iteration is one step further towards the final product. This continues until all product functionalities are satisfied.  Organizations and users use the working product and provide feedback that’s incorporated into the next iteration. With this, product roadmaps can be built, produced, and tested before the next iteration.  This is one of the most common and important Agile Interview Questions that everyone should know the answers to. | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | What are the different types of Agile methodologies?  1. Extreme Programming  It is a framework that enables teams to create high-quality software that helps improve their quality of life. It enables software development alongside relevant and appropriate engineering practices. It is applicable to handle changing software requirements, risks caused due to new software, working with a small, extended development team, and technology that allows automated unit and functional tests.  2. Kanban  It is a method that’s used to design, manage, and improve the flow of systems. Kanban enables organizations to visualize their flow of work and limit the amount of work in progress. It is used in situations where work arrives unpredictably, and where work needs to be deployed immediately without waiting for other work items.  3. Lean  It is a set of tools and principles that focuses on identifying and removing waste, to speed up process development. Value is maximized, and waste is minimized. It is used in just about every industry that produces waste in some form or the other.  4. Scrum  It is a framework that is used by teams to establish a hypothesis, test it, reflect on the experience, and make adjustments. It enables teams to incorporate practices from other frameworks depending on the requirements. It is used by cross-functional teams that are working on product development, and the work is split into more than one 2-4 week iterations.  5. Crystal  It focuses on people and their interactions, rather than on tools and processes. Aimed to streamline processes and improve optimization, Crystal works on the principle that projects are unique and dynamic. It is used when the focus is on strengthening team communication, continuous integration, active user involvement, and configurable processes. | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | What Are the Principles of Agile Testing?  Testing continuously to ensure continuous improvement of the product  Continuous feedback to ensure the product satisfies the business requirements  Team roles like testing and development need to be actively involved in the testing process  Actively involving the business team and user representatives can help obtain quick feedback for each iteration  Clean and simplified code ensures it is defect-free during the iteration  Documentation created must limit itself to a particular iteration  Testing is done alongside development and implementation, to ensure that the product is free from defects  Involving the user ensures the final product matches their requirements | </end> |
| <hitle> | Agile metrics | <chare> | 2 | <pext> | What are some Agile metrics that need to be focused on?  Measuring the burndown of deliverables,  usually represented in burndown charts  Velocity  Lead time  Cycle time  Code quality  Code covered in unit test  Deployment success rate  Net promoter score | </end> |
| <hitle> | Kanban | <chare> | 2 | <pext> | What Is Kanban?  Kanban is a visual system that helps the management of work  as it progresses through the process.  It visualizes and provides visibility into the process, workflows,  and the work that’s passing through the process.  It is also known as a pull system, as new work is pulled from a list,  executed, and moved from ‘in-progress’ status  to ‘done’ Tracking work is also made easier  as the workflow is visible and put on a display board.  Modern organizations can also use the digital display systems.  In Kanban, the goal is to identify constraints or potential bottlenecks  in the process and ensure they are addressed.  It also helps make the workflow smoother and more efficient. | </end> |
| <hitle> | Obstacles Agile | <chare> | 2 | <pext> | What are the obstacles to the Agile process?  Some of the obstacles that could be faced are:  1. Not having appropriate or sufficient tools and technologies  2. The lack of active involvement from the customers  3. Team members that are lacking in skills and capability  4. The inability to design systems based on unseen requirements  5. Successfully adopting the Agile culture to the organization | </end> |
| <hitle> | Agile vs Scrum | <chare> | 2 | <pext> | Differentiate between Agile and Scrum.  1. Agile is a set of principles that’s iterative and incremental in nature  1-1. Scrum, is an implementation of the Agile methodology  2. Agile, Suited for projects involving a small team of experts  2-1. Scrum, Used in teams that need to handle constant changing requirements  3. Agile, The project head takes care of all tasks and is vital to the project  3-1. Scrum, No leader. Issues are handled by the scrum master and the team  4. Agile, Changes cannot be handled frequently  4-1. Scrum, Teams can react to change quickly  5. Agile, Requires frequent delivery to the end-user  5-1. Scrum, Sprints provide workable builds of the final product to the user for feedback  6. Agile, Face-to-face interactions between cross-functional teams  6-1. Scrum, Daily stand-up meetings help with collaboration  7. Agile, Design and execution is simple  7-1. Scrum, Design and execution can be innovative and experimental | </end> |
| <hitle> | Agile certifications | <chare> | 2 | <pext> | What are some popular Agile certifications?  PMI-ACP Certification  Scrum Master Certification  Certified Scrum Master Certification  Prince 2 Agile Certification  Scrum Product Owner Certification | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | What are Agile frameworks?  Other than Scrum, there are various alternative frameworks, including Test-Driven Development, Feature-Driven Development, and Kanban. | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | When should you use Waterfall over Scrum?  Typically, Waterfall is employed when the requirements are straightforward, completely specified, predictable, understandable, and permanent. | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | How long does the scrum cycle last?  The length of the Scrum cycle is determined by the size of the team and the project; it typically takes 3 to 4 weeks to complete. | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | What is the scrum of scrums?  A separate meeting, known as a Scrum of Scrums, must be planned to coordinate among several teams. In the scrum of scrums, an ambassador speaks on behalf of their squad. | </end> |
| <hitle> | epic, user stories | <chare> | 2 | <pext> | Define epic, user stories, and tasks?  User Stories: Delivered throughout a specific sprint,  user stories are created based on the client's perspective,  such as project or business functions. | </end> |
| <hitle> | test-driven development | <chare> | 2 | <pext> | What is test-driven development?  This approach involves a developer writing an automated test case  that describes the new function, writing little code to pass the test,  and then re-factoring the new code to comply with the required standards. | </end> |
| <hitle> | scrum technique planning poker | <chare> | 2 | <pext> | What do you mean by scrum poker technique or planning poker?  The general agreement-based card-based estimation method  is called scrum poker or planning poker.  Among its characteristics are the following:  1. The client must read the agile user's story as the first step.  2. The estimator then comprehends the features.  3. Each estimator has a distinct planning card  with a different set of numbers on it.  The various numbers represent the plot points.  4. This procedure is carried out repeatedly until a consensus is obtained. | </end> |
| <hitle> | Agile cons | <chare> | 2 | <pext> | What do you mean by the disadvantages of the Agile Model?  The disadvantages are:  1. You will have trouble calculating the effort needed for a large project  if one is assigned.  2. It's not always best to concentrate on documentation and design. | </end> |
| <hitle> | Whento use agile | <chare> | 2 | <pext> | What is the right moment to use the agile model?  Lean software development, feature-driven development,  dynamic development, and crystal approaches  are a few methodologies and innovations that can employ agile. | </end> |
| <hitle> | release candidate | <chare> | 2 | <pext> | What do you mean by release candidate?  A code, version, or build—known as a release candidate—  is released to guarantee that no significant issues  remain during the final development stage.  This is used for testing and is the same as the finished build. | </end> |
| <hitle> | Agile vs other testing | <chare> | 2 | <pext> | Differentiate between agile testing methods and other testing methods?  The code is broken into smaller pieces in the main agile testing cases.  Each code branch is tested at one particular time.  On that section of the code, the communication is ongoing. | </end> |
| <hitle> | Zero sprint | <chare> | 2 | <pext> | What is Zero sprint in Agile?  The first sprint's pre-step is known as a zero sprint. | </end> |
| <hitle> | story point scrum | <chare> | 2 | <pext> | What does story point mean in the scrum?  A story point is the scrum term for the unit used to estimate the time  needed to complete or implement a backlog. | </end> |
| <hitle> | Scrum master recommends | <chare> | 2 | <pext> | In which process a Scrum master recommends follow-up action items?  Establishing a follow-up of the work that needs to be done  by the team members is the best way to follow up on action items.  Additionally, the scrum master will compile this data. | </end> |
| <hitle> | agile matrices | <chare> | 2 | <pext> | What are the most critical agile Matrices?  The most important agile matrices are listed below:  1. Velocity  2. Work category allocation  3. Defect removal awareness  4. The cumulative flow diagram | </end> |
| <hitle> | Agile tester qualities | <chare> | 2 | <pext> | What are the essential qualities a good Agile tester should have?  A qualified Agile tester must possess the following qualities:  1. Should swiftly comprehend the requirements  2. Must be well-versed in the concepts and practices of Agile. | </end> |
| <hitle> | Agile challenges | <chare> | 2 | <pext> | Mention the challenges involved in developing Agile Software?  The significant difficulties when creating Agile Software are:  1. more customer interaction and testing  2. management is more affected than developers  3. More preparation is needed. | </end> |
| <hitle> | When not to use Agile | <chare> | 2 | <pext> | When not to use Agile?  You can ditch Agile in the following circumstances:  1. Your team does not understand Agile  2. Your team is resisting Agile  3. Expectations do not support agile | </end> |
| <hitle> | product roadmap | <chare> | 2 | <pext> | Explain what it means by product roadmap?  A product roadmap is a source of truth used by all parties to describe  a product's long-term goals, priorities, and progress. | </end> |
| <hitle> | product burndown chart | <chare> | 2 | <pext> | What is the product burndown chart?  A project management chart called a burndown chart demonstrates  how rapidly a team is moving through a customer's user stories. | </end> |
| <hitle> | sprint burndown chart | <chare> | 2 | <pext> | What is a sprint burndown chart?  Sprint burndown chart shows the number of  completed and uncompleted sprints in the scrum cycle. | </end> |
| <hitle> | Release burndown chart | <chare> | 2 | <pext> | What is the Release burndown chart?  A release burndown chart summarizes the release progress  by displaying the remaining workload at the end of every sprint  versus the ideal workload. | </end> |
| <hitle> | sprint velocity | <chare> | 2 | <pext> | How is the velocity of the sprint measured?  Work is complete if capacity is expressed as a percentage  of 40-hour workweeks: story points \* team capacity.  If the capacity is expressed in terms of man-hours,  then the team's capacity is expressed in story points. | </end> |
| <hitle> | agile certifications | <chare> | 2 | <pext> | Are there any agile certifications?  The following list of agile certifications and scrum certifications:  ACP (Agile Certified Practitioner)  ASM (Agile Scrum Master)  Safe Agilist  CSM (Certified Scrum Master)  PSM (Professional Scrum Master) UpdationgRecords | </end> |
| <hitle> | agile quality strategies | <chare> | 2 | <pext> | What are the major agile quality strategies?  These are the main strategies:  1. Small feedback cycles  2. Re-factoring  3. Iterations  4. Analysis of dynamic code | </end> |
| <hitle> | Product backlog and Sprint backlog | <chare> | 2 | <pext> | Define Product backlog and Sprint backlog.  The project owner keeps a product backlog that includes  every requirement and product feature.  A Sprint backlog is viewed as a subset of a product backlog  that only includes needs and features pertinent to one specific sprint. | </end> |
| <hitle> | Pair Programming | <chare> | 2 | <pext> | What is Pair Programming, and what states its benefits?  Pair programming is a method where two programmers collaborate.  The work of one coder is reviewed by the other.  Benefits is Smooth knowledge transfer.  The codes and tactics can be taught to another partner  by an experienced partner.  Code quality is The likelihood of errors occurring will decrease  as the second partner simultaneously reviews the code. | </end> |
| <hitle> | Agile refactoring | <chare> | 2 | <pext> | What is refactoring?  Refactoring is making changes to code without altering its functionality  to increase performance. | </end> |
| <hitle> | QA add value to an agile team | <chare> | 2 | <pext> | How can QA add value to an agile team?  By considering various scenarios differently, QA adds value to the story.  It gives the developers swift feedback on whether the new functionality  is operating as intended. | </end> |
| <hitle> | Application Binary Interface | <chare> | 2 | <pext> | What is the Application Binary Interface?  Application Binary Interface describes the low-level interface  between an application and the operating system or the interface  to compile application applications. | </end> |
| <hitle> | tracer bullet | <chare> | 2 | <pext> | What is a tracer bullet?  Tracer Bullet investigates how an end-to-end process functions  and looks at its viability.  With the existing architecture or set of procedures, it is the spike. | </end> |
| <hitle> | Agile increment | <chare> | 2 | <pext> | What is the increment? Explain.  The sum of every product's backlog items during a sprint  is known as the increment.  Since they are cumulative, each increment comprises the values  from the previous sprint increment.  As it is a step toward achieving the aim, it should be configured  to be in the available mode in its upcoming release. | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | How to manage remote teams?  Utilizing remote project management solutions can enable you  to manage your remote team effectively  while using outsourced resources.  It assists you in managing people, tools, and resources  and gaining their support. | </end> |
| <hitle> | sprint retrospective meeting | <chare> | 2 | <pext> | What is meant by the sprint retrospective meeting?  It is the meeting held following the sprint review meeting  and takes place in the last stages of the sprint.  This conversation lasts for two to three hours  and will involve the entire team. | </end> |
| <hitle> | Agile to other testing | <chare> | 2 | <pext> | Can we apply agile methodologies to other testing apart from development and software testing?  Agile file testing approaches can be used in the biomedical,  biophysical, and biochemical fields and other areas  where there is a lack of data and a small team is required  to finish the project. | </end> |
| <hitle> | term impediment | <chare> | 2 | <pext> | What does the term impediment mean?  An impediment prevents the team from working efficiently  and hinders their ability to complete tasks more effectively. | </end> |
| <hitle> | sprint planning meetings | <chare> | 2 | <pext> | What do sprint planning meetings do?  A sprint planning meeting gathers the whole scrum team,  including the product owner, scrum master, and other team members.  It is planned to discuss the items on the product backlog  and the key team characteristics. | </end> |
| <hitle> | places Scrum and Kanban | <chare> | 2 | <pext> | Mention the places where the Scrum and Kanban methodologies are used?    Scrum is used when switching to the most visible  and relevant methodology is necessary.  Kanban is deployed when there is a need  to enhance an operating process with few adjustments. | </end> |
| <hitle> | Scrum Master essential qualities | <chare> | 2 | <pext> | What are the essential qualities to have as a Scrum Master?  The main characteristics of a Scrum Master are sharing knowledge  and comprehending the goals and objectives. | </end> |
| <hitle> | scrum poker technique planning | <chare> | 2 | <pext> | What do you mean by planning poker or scrum poker technique?  The characteristics of planning poker include the following:  1. The client must first read the agile user's story.  The estimator then comprehends the features.  2. Each estimator has a distinct planning card with  a unique set of numbers.  The different numbers represent the story points.  3. This procedure is carried out continuously  until an agreement is obtained. | </end> |
| <hitle> | right moment to use the agile | <chare> | 2 | <pext> | What is the right moment to use the agile model?  Certain approaches and projects can use agile,  including lean software development, dynamic development,  and crystal methodologies. | </end> |
| <hitle> | Agile vs other testing | <chare> | 2 | <pext> | How do agile testing methods differ from other testing methods?  The code is divided into smaller branches  when using an agile testing methodology.  Each specific unit of code is tested at the same time.  In that section of the code, communication is continuing as well. | </end> |
| <hitle> | Agile to other testing | <chare> | 2 | <pext> | Is it possible to apply agile methodology to other testing apart from software testing and development testing?  The file testing methodology can be used in the fields  where there is a lack of data and a small team is required  to finish the project. | </end> |
| <hitle> | Howto know using agile | <chare> | 2 | <pext> | How to know that you are using agile development?  You are adopting agile development when you use  a time-boxed task board, pair programming, daily stand-up meetings,  test-driven development, and other practices. | </end> |
| <hitle> | main roles scrum | <chare> | 2 | <pext> | Describe the main roles in the scrum.  1. Scrum team: Responsible for organizing a group effort  to finish a specific assignment.  2. Scrum Master: is in charge of putting the scrum team's final product on paper.  3. Product owner: In charge of outlining the entire idea  for the product and communicating it to the team. | </end> |
| <hitle> | build breaker | <chare> | 2 | <pext> | What does build breaker mean?  Sometimes a software flaw is unintentionally introduced by the developer.  This flaw could cause warring or stop the compilation process.  It is the root of failure when testing is carried out normally.  It is deemed that the build is broken in such circumstances. | </end> |
| <hitle> | Places Scrum and Kanban used | <chare> | 2 | <pext> | What are the places where Scrum and Kanban are used?  Scrum is used when it's necessary to move  toward a suitable and noticeable method.  Kanban is employed when process improvement is required,  provided that there aren't many changes. | </end> |
| <hitle> | Agile software testing | <chare> | 2 | <pext> | Can agile methodology also be applied other than software testing and development projects?  In the fields of biophysics, biochemistry, and biomedicine,  agile methodology can be used in a range of situations,  such as when there is a lack of data when a small team is required  to complete the project when the nature of the job is uncertain,  and in several other situations. | </end> |
| <hitle> | major Agile components | <chare> | 2 | <pext> | What are the major Agile components?  The major Agile elements are:  1. Pair programming, continuous deployment, test-driven development, etc.  2. Collaboration cards and class responsibilities  3. stand-up meetings daily | </end> |
| <hitle> | master recommends | <chare> | 2 | <pext> | What is the process in which a master recommends following up on action items?  The best way to accomplish this is by creating a follow-up  of the job the team members are expected to complete. | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | UpdationgRecords | <chare> | 2 | <pext> | UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords  UpdationgRecords | </end> |
| <hitle> | Git, Github | <chare> | 1 | <pext> | Git is version control tool | </end> |
| <hitle> | Create repository | <chare> | 2 | <pext> | 1. login github and click “+” button on top-right.  2. “New repository” and give name you want, “test”  3. In the pc/mac, go to the project folder and right click and open terminal for that folder  4. “git init” to init repo  5. “git add .” to add all files to repo  6. “git remote add origin https://github.com/howdyev/test.git” to add the repo link that can be copied on the account.  7. “git commit –m (first test)” to commit codebase  8. “git push” to push the codebase | </end> |
| <hitle> | manage multiple GitHub accounts | <chare> | 2 | <pext> | 1. Generating the SSH keys  Before generating an SSH key, we can check to see if we have any existing SSH keys:  # ls -al ~/.ssh  This will list out all existing public and private key pairs, if any.  If ~/.ssh/id\_rsa is available, we can reuse it, or else we can first generate a key to the default ~/.ssh/id\_rsa by running:  # ssh-keygen -t rsa  When asked for the location to save the keys, accept the default location by pressing enter.  A private key and public key ~/.ssh/id\_rsa.pub will be created at the default ssh location ~/.ssh/.  Let’s use this default key pair for our personal account.  For the work accounts, we will create different SSH keys. The below code will generate the SSH keys, and saves the public key with the tag “email@work\_mail.com” to ~/.ssh/id\_rsa\_work\_user1.pub  # ssh-keygen -t rsa -C "email@work\_mail.com" -f "id\_rsa\_work\_user1"  We have two different keys created:  ~/.ssh/id\_rsa  ~/.ssh/id\_rsa\_work\_user1  2. Adding the new SSH key to the corresponding GitHub account  We already have the SSH public keys ready, and we will ask our GitHub accounts to trust the keys we have created. This is to get rid of the need for typing in the username and password every time you make a Git push.  Copy the public key  # pbcopy < ~/.ssh/id\_rsa.pub  and then log in to the GitHub account:  2-1. Go to “Settings”  2-2. Select “SSH and GPG keys” from the menu to the left.  2-3. Click on “New SSH key”, provide a suitable title, and paste the key in the box below  2-4. Click “Add key” and you’re done!  For the work accounts, use the corresponding public keys  # pbcopy < ~/.ssh/id\_rsa\_work\_user1.pub  and repeat the above steps in your GitHub work accounts.  3. Registering the new SSH Keys with the ssh-agent  To use the keys, we have to register them with the ssh-agent on our machine. Ensure ssh-agent is running using the command  # eval "$(ssh-agent -s)"  Add the keys to the ssh-agent like so:  # ssh-add ~/.ssh/id\_rsa  # ssh-add ~/.ssh/id\_rsa\_work\_user1  Make the ssh-agent use the respective SSH keys for the different SSH Hosts.  This is the crucial part, and we have two different approaches:  Using the SSH configuration file (Step 4), and having only one active SSH key in the ssh-agent at a time (Step 5).  4. Creating the SSH config File  Here we are actually adding the SSH configuration rules for different hosts, stating which identity file to use for which domain.  The SSH config file will be available at ~/.ssh/config. Edit it if it exists, or else we can just create it.  # cd ~/.ssh/  # touch config // Creates the file if not exists  # code config // Opens the file in VS code, use any editor  Make configuration entries for the relevant GitHub accounts similar to the one below in your ~/.ssh/config file:  # Personal account, - the default config  Host github.com  HostName github.com  User git  IdentityFile ~/.ssh/id\_rsa    # Work account-1  Host github.com-work\_user1  HostName github.com  User git  IdentityFile ~/.ssh/id\_rsa\_work\_user1  “work\_user1” is the GitHub user id for the work account.  “github.com-work\_user1” is a notation used to differentiate the multiple Git accounts. You can also use “work\_user1.github.com” notation as well. Make sure you’re consistent with what hostname notation you use. This is relevant when you clone a repository or when you set the remote origin for a local repository  The above configuration asks ssh-agent to:  4-1. Use id\_rsa as the key for any Git URL that uses @github.com  4-2. Use the id\_rsa\_work\_user1 key for any Git URL that uses @github.com-work\_user1  5. One active SSH key in the ssh-agent at a time  This approach doesn’t require the SSH config rules. Rather we manually ensure that the ssh-agent has only the relevant key attached at the time of any Git operation.  ssh-add -l will list all the SSH keys attached to the ssh-agent. Remove all of them and add the one key you are about to use.  If it’s to a personal Git account that you are about to push:  # ssh-add -D //removes all ssh entries from the ssh-agent  # ssh-add ~/.ssh/id\_rsa // Adds the relevant ssh key  The ssh-agent now has the key mapped with the personal GitHub account, and we can do a Git push to the personal repository.  To push to your work GitHub account-1, change the SSH key mapped with the ssh-agent by removing the existing key and adding the SSH key mapped with the GitHub work account.  # ssh-add -D  # ssh-add ~/.ssh/id\_rsa\_work\_user1  The ssh-agent at present has the key mapped with the work Github account, and you can do a Git push to the work repository. This requires a bit of manual effort, though.  Setting the git remote Url for the local repositories  Once we have local Git repositories cloned /created, ensure the Git config user name and email is exactly what you want. GitHub identifies the author of any commit from the email id attached with the commit description.  To list the config name and email in the local Git directory, do git config user.name and git config user.email. If it’s not found, update accordingly.  # git config user.name "User 1" // Updates git config user name  # git config user.email [user1@workMail.com](mailto:user1@workMail.com)  6. While Cloning Repositories  Note: step 7 will help, if we have the repository already available on local.  Now that the configurations are in place, we can go ahead and clone the corresponding repositories. On cloning, make a note that we use the host names that we used in the SSH config.  Repositories can be cloned using the clone command Git provides:  # git clone [git@github.com:personal\_account\_name/repo\_name.git](mailto:git@github.com:personal_account_name/repo_name.git)  The work repository will require a change to be made with this command:  # git clone [git@github.com-work\_user1:work\_user1/repo\_name.git](mailto:git@github.com-work_user1:work_user1/repo_name.git)  This change is made depending on the host name defined in the SSH config. The string between @ and : should match what we have given in the SSH config file.  7. For Locally Existing Repositories  If we have the repository already cloned:  List the Git remote of the repository,  #git remote -v  Check whether the URL matches our GitHub host to be used, or else update the remote origin URL.  # git remote set-url origin [git@github.com-worker\_user1:worker\_user1/repo\_name.git](mailto:git@github.com-worker_user1:worker_user1/repo_name.git)  Ensure the string between @ and : matches the Host we have given in the SSH config.  If you are creating a new repository on local:  Initialize Git in the project folder  # git init  Create the new repository in the GitHub account and then add it as the Git remote to the local repository.  # git remote add origin [git@github.com-work\_user1:work\_user1/repo\_name.git](mailto:git@github.com-work_user1:work_user1/repo_name.git)  Ensure the string between @ and : matches the Host we have given in the SSH config.  Push the initial commit to the GitHub repository:  # git add .  # git commit -m "Initial commit"  # git push -u origin master  We are done! | </end> |
| <hitle> | Create github access token | <chare> | 2 | <pext> | 1. On github page, click “Settings”  2. click “Developer settings”  3. click “Personal access tokens” / “Fine-graned personal access tokens”  4. click “Generate new token”  5. Give “Token name” and select “Expiration”  6. Optionally give “Description”  7. And generate token  8. Important Note: You have to save the token because it is only one time show at creating. | </end> |
| <hitle> | SourceTree | <chare> | 2 | <pext> | ------------Clone the remote repository -----------------------  1. Click “File status” and select files what needs to be pushed.  2. Enter commit message at the bottom.  3. Click checkbox “Push changes imm…”  4. Click “commit”  ------------Pull remote updates with stash-----------------------------  Suppose we are on the working branch, like “callFeature”  1. Click “Stash” to stash everything.  We can give stash name. “stash1”, “stash2”, …  2. Double click “master” branch  3. Click “Pull”  4. Double click the working branch, like “callFeature”  5. Rigth click on “master’ while selecting the working branch.  6. Merge master into the working branch – “callFeature”  We should not “commit” checked.  7. Right click on “Stashes/stash1” and click “Apply stash”  8. Click “File status” and check warning signed files.  9. Right click on warning signed file, and “Resolve conflicts/Resolve Using ‘mine’/’them’”  ------------Push my changes to remote-----------------------  1. Click “File status” and select files what needs to be pushed.  2. Enter commit message at the bottom.  3. Click checkbox “Push changes imm…”  4. Click “commit” | </end> |
| <hitle> | Local History Android Stuido | <chare> | 2 | <pext> | We can recover old work from Local History in Android Stuiod.  Right click on a file and click “Local History/ShowHistory”.  We can see all the history of changes from the creation.  We can click “Up arrow” and “Down arrow” to apply old changes.  Or we can just copy any piece of code from history and use it  to recover. | </end> |
| <hitle> | Local project to github | <chare> | 2 | <pext> | 1. Suppose we have a project created in local “GatsbyTest”  2. In the local folder,  # git init  # git add .  # git commit –m “first commit”  3. Create a repository on github and copy the link.  [git@github.com:Howdyev/Gatsby-Test.git](mailto:git@github.com:Howdyev/Gatsby-Test.git)  4. In the local folder,  # git remote add origin [git@github.com:Howdyev/Gatsby-Test.git](mailto:git@github.com:Howdyev/Gatsby-Test.git)  # git push –u origin master | </end> |
| <hitle> | Open local project in SourceTree | <chare> | 2 | <pext> | 1. In the first screen, New-Clone/Add Existing Local Repository.  2. Select local project folder. | </end> |
| <hitle> | UpdatingRecords | <chare> | 2 | <pext> | UpdatingRecords  UpdatingRecords  UpdatingRecords  UpdatingRecords  UpdatingRecords | </end> |