Lab Manual 2

Course Code: CS-165L **Course Title: Software Engineering**

Course Instructor: Laeeq Khan Niazi

Last Week tasks progress?

Requirement Gathering.

1. **Interviews:** Conducting interviews with stakeholders, end-users, and subject matter experts is a traditional but valuable way to gather requirements. It allows for in-depth discussions and clarifications.
2. **Surveys and Questionnaires:** Surveys and questionnaires can be used to collect input and feedback from many stakeholders, making them useful for gathering a broad range of opinions.
3. **Brainstorming:** Brainstorming sessions with relevant stakeholders can generate creative ideas and uncover hidden requirements.
4. **Use Cases:** Use cases are graphical representations that describe how a system will interact with its users. Use case diagrams help in visualizing requirements and system behavior.
5. **User Stories:** Commonly used in Agile methodologies, user stories are short, simple descriptions of a feature from an end-user perspective.

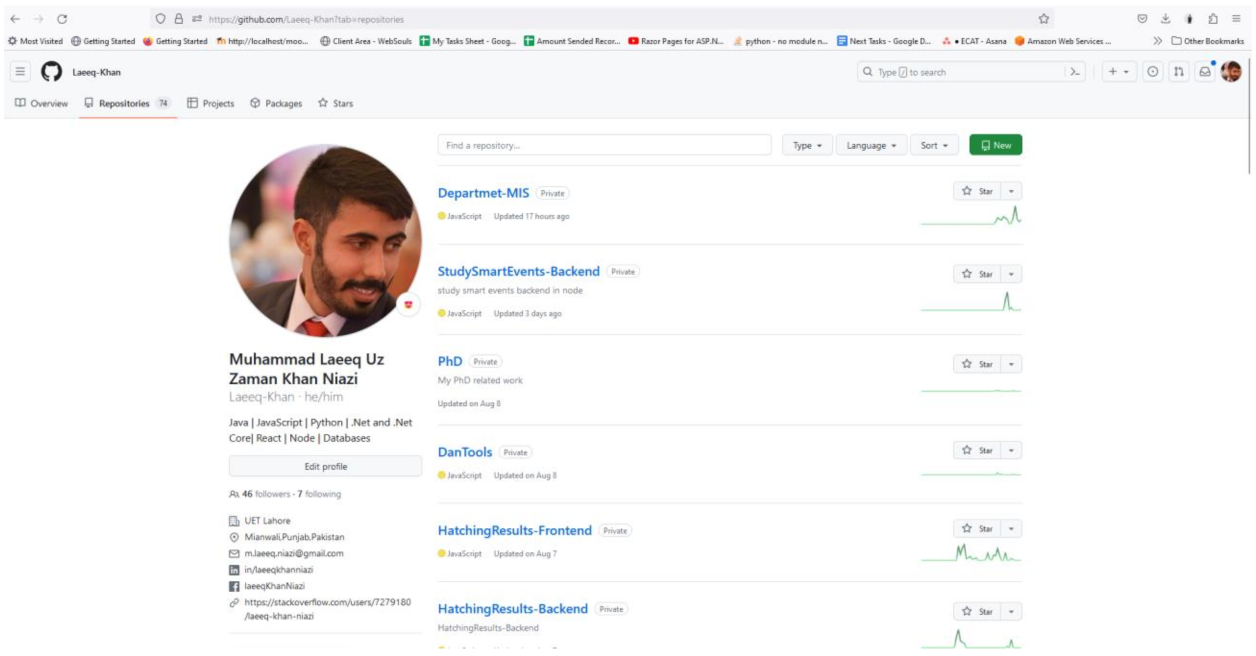
   They often follow the "As a [user], I want [functionality] so that [benefit]" format.

6. **Prototyping Tools:** Prototyping tools like Balsamiq, Sketch, or Axure RP allow you to create interactive mockups or prototypes to visualize and refine requirements.
7. **Requirements Management Software:** Tools like IBM Engineering Requirements Management DOORS, Jama Connect, or Atlassian Jira can help in documenting, tracking, and managing requirements throughout the project's lifecycle.
8. **Version Control Systems:** Version control systems like Git can be used to manage changes to requirements documents, ensuring that you have a history of changes and can revert to previous versions if necessary.
9. **Collaboration and Communication Tools:** Tools like Slack, Microsoft Teams, and project management software facilitate communication and collaboration among team members and stakeholders, which is crucial for requirement gathering and clarification.
10. **Requirements Workshops:** Organizing workshops or focus groups with stakeholders can be an effective way to gather requirements collaboratively and ensure everyone's input is considered.
11. **Mind Mapping Software:** Tools like MindMeister or XMind can be used to create visual representations of requirements and their relationships.
12. **Simulation and Modeling Tools:** For complex systems, simulation and modeling tools can help validate requirements by simulating how the system will behave under various conditions.

Class Task 1→ Every developer should meet with his/her client and take the requirements in soft form and discuss the project in detail. After that everything should be communicated on slack. (Total 30 Minutes Task, 15 minutes for each group)

GitHub Guide

GitHub is a web-based platform and service that provides a wide range of functionality for **software development** and **version control**. It's one of the most popular and widely used platforms for hosting and collaborating on software projects. There are other that can be use for the same purpose like GitLab, Bitbucket, Gitkraken, AWS CodeCommit, Beanstalk etc.



Departmet-MIS Private
JavaScript Updated 17 hours ago

StudySmartEvents-Backend Private
study smart events backend in node
JavaScript Updated 3 days ago

PhD Private
My PhD related work
Updated on Aug 8

DanTools Private
JavaScript Updated on Aug 8

HatchingResults-Frontend Private
JavaScript Updated on Aug 7

HatchingResults-Backend Private
HatchingResults-Backend

Muhammad Laeeq Uz Zaman Khan Niazi
Laeeq-Khan · he/him
Java | JavaScript | Python | .Net and .Net Core| React | Node | Databases
Edit profile
46 followers · 7 following
UET Lahore
Mianwali,Punjab,Pakistan
m.laeeq.niazi@gmail.com
in/laeeqkhanniazi
laeeqKhanNiazi
https://stackoverflow.com/users/7279180/laeeq-khan-niazi

Git commands that you can use

1. **git init**:
   - `git init` is used to initialize a new Git repository in the current directory. It creates a hidden `.git` directory where Git stores all the metadata and configuration for version control.
2. **git add**:
   - `git add` is used to stage changes for a commit. It tells Git which files or changes you want to include in the next commit. You can use it to stage specific files or directories.
     - Example: `git add file.txt` stages a specific file for the next commit.
3. **git commit**:
   - `git commit` creates a new commit with the changes that you've staged using `git add`. It also allows you to add a commit message to describe the changes you're making.

        - Example: `git commit -m "Add new feature"` creates a commit with the message "Add new feature."

4. **git status**:
    - `git status` shows the status of your working directory and any changes that are staged or not staged for commit. It's a helpful command to understand what's going on in your repository.
        - Example: `git status` provides information about the current state of your branch.
5. **git branch**:
    - `git branch` lists all the branches in your Git repository. It also shows the currently active branch, which is marked with an asterisk (*).
        - Example: `git branch` lists all branches in your repository.
6. **git setup Origin**
    - In Git, the git remote add origin command is used to set up a remote repository as the "origin" for your local Git repository.
    - git remote add origin <repository_url>. Here add your git repo url.
7. **git push**:
    - `git push` is used to push your local commits to a remote repository, typically on platforms like GitHub or GitLab. It updates the remote branch with your changes.
        - Example: `git push origin master` pushes your local changes in the "master" branch to the remote repository named "origin." First time when you push the code on GitHub you have to run git upstream command too suggested by gitbash.

Commit Message Format:

A typical commit message consists of a brief one-line summary followed by an optional detailed description. The one-line summary should be concise and descriptive, while the detailed description provides more context about the changes.

One-line summary guidelines:

- Keep it concise but informative.
- Use the present tense (e.g., "Add feature" instead of "Added feature").
- Start with an action verb (e.g., "Fix," "Add," "Update," "Remove," etc.).
- Use imperative mood (e.g., "Fix bug" instead of "Fixing bug").

Detailed description guidelines:

- Add a blank line after the one-line summary.
- Provide context, explain why the change is necessary, and how it affects the codebase.
- Mention any issues or references related to the commit.
- Use bullet points or paragraphs for readability.

Example Commits:

Example 1. Adding new Feature commit

One line summary

Add user authentication feature

This commit adds a new user authentication feature that allows users to log in using their email and password. It also includes the necessary database schema changes to store user credentials securely.

Fixes #123

Detailed Description

Example 2. Fixing a bug

Fix null pointer exception in data processing

This commit addresses a bug where a null pointer exception was occurring during data processing. The issue was caused by improper handling of empty data arrays. This fix adds proper null checks to prevent the exception.

Closes #456

Example 3. Updating documentation

Update installation instructions in README

This commit updates the installation instructions in the README file to reflect the new requirements and installation process. It also includes additional troubleshooting tips for common issues users might encounter during installation.

Related to #789

Class Task 2 → Setup your project GitHub repo make it public and setup readme file with project title, project description, project technology, developer name, course name and add your developer into the repo as collaborator. (20 Minutes Task)

Note: If you find any issue or bug add that on the GitHub, at time of evaluation you must have created 20 issues at least on the GitHub and these issues should be addressed. Please follow the GitHub guidelines for issues addressing.

What is Agile software development?

Agile development is a set of software development methodologies and practices that prioritize flexibility, collaboration, and customer-centricity. It is based on the Agile Manifesto, a document created by a group of software development experts in 2001, which outlines the core principles of Agile software development.

1. **Philosophy:** Agile is a philosophy or mindset that emphasizes flexibility, customer collaboration, and responsiveness to change. It values individuals and interactions over processes and tools.
2. **Methodologies:** Agile includes various methodologies like Scrum, Kanban, Extreme Programming (XP), and more. These methodologies provide specific frameworks and practices for implementing Agile principles. **(Details of the Scrum, Kanban and XP given below try to select which method you will use for this project with reason)**
3. **Iterations:** Agile development typically divides the project into short, fixed-length iterations called sprints (e.g., 2-4 weeks). Each sprint results in a potentially shippable product increment.
4. **Requirements:** Agile projects often start with a high-level set of requirements but allow for changes and adjustments throughout the development process. Requirements can evolve based on customer feedback**. (Discuss your project requirements with developer and if there is any suggestion or feedback mention in the backlog document)**
5. **Customer Involvement:** Agile encourages continuous collaboration with customers and stakeholders. They are involved throughout the project, providing feedback that drives development priorities.
6. **Deliverables:** Agile teams aim to deliver working, tested software at the end of each iteration. This allows for early and frequent product releases.
7. **Change Management:** Agile welcomes changing requirements, even late in the project. Changes are expected and accommodated without significant disruptions.
8. **Testing:** Testing is an integral part of Agile development, with a focus on automated testing and continuous integration. **(Try to select some tools you can use for automated testing of your project) (10 Minutes Task)**

1. Scrum:

Scrum is an Agile framework that provides a structured approach to software development, emphasizing collaboration, transparency, and adaptability. It organizes work into **fixed-length time-boxed** iterations called sprints, typically lasting 2-4 weeks. Here's an explanation with an example:

Principles of Scrum:

- **Roles:** Scrum defines three key roles:
  - **Product Owner:** Represents the customer and defines the product backlog ("backlog" refers to a prioritized list of work items, features, user stories, or tasks that need to be completed in a software development project).

    - **Scrum Master:** A Scrum Master is like a team coach in Scrum. They make sure the team follows Scrum rules and helps the team overcome any obstacles or problems that might get in their way.
    - **Development Team:** Cross-functional team members who deliver the product.
- **Artifacts:**
    - **Product Backlog:** A prioritized list of all desired features and requirements.
    - **Sprint Backlog:** A subset of the product backlog selected for the current sprint.
    - **Increment:** A potentially shippable product increment created during each sprint.
- **Events:**
    - **Sprint Planning:** The team plans the work for the upcoming sprint.
    - **Daily Standup:** A daily meeting where team members share progress and discuss impediments. (In your case you can arrange 2 meetings in week)
    - **Sprint Review:** A review of the completed increment with stakeholders.
    - **Sprint Retrospective:** A reflection on the sprint to improve processes.

Example: Imagine a software development team working on a project management tool. The product owner maintains a product backlog with features like task management, user roles, and reporting. For a specific sprint, the team selects tasks from the product backlog and creates a sprint backlog. During the sprint, they work on these tasks, and at the end of the sprint, they have a potentially shippable increment that might include task management features. The sprint review involves showcasing these features to stakeholders for feedback.

2. Kanban: (We will follow this approach in development of your semester project)

Kanban is an Agile method that focuses on visualizing work, limiting work in progress (WIP), and optimizing workflow efficiency. Unlike Scrum, Kanban does not use fixed-length iterations. Here's an explanation with an example:

Principles of Kanban:

- **Visualize Workflow:** Use a Kanban board to visualize tasks or work items as they move through various stages (e.g., to-do, in progress, done).
- **Limit WIP:** Set limits on the number of tasks allowed in each stage to prevent overloading the team and maintain a smooth flow.
- **Manage Flow:** Ensure work items move through stages efficiently, identifying bottlenecks and addressing them.
- **Make Process Policies Explicit:** Define clear rules for how work is handled at each stage.

Example: Imagine a customer support team using Kanban to manage customer requests. Their Kanban board has columns for "New Requests," "In Progress," "Testing," and "Completed." The team limits the number of requests in each column to prevent overloading. When a new request comes in, it's added to the "New Requests" column. Team members pull items into "In Progress" when they have capacity. This approach allows the team to manage their workload efficiently and ensures a steady flow of support requests.

3. Extreme Programming (XP):

Extreme Programming (XP) is an Agile methodology that focuses on engineering practices to improve software quality and responsiveness to changing requirements. XP emphasizes close collaboration between developers and customers. Here's an explanation with an example:

Principles of XP:

- **Continuous Integration:** Developers integrate code frequently to identify and resolve issues early.
- **Test-Driven Development (TDD):** Write tests before writing code to ensure code quality and validate functionality.
- **Pair Programming:** Two developers work together at one computer, improving code quality and knowledge sharing.
- **Customer Collaboration:** Involve customers throughout the development process to get constant feedback.

Example: Suppose a software development team is building an e-commerce website using XP. Developers follow TDD, writing automated tests for each new feature before coding. They practice pair programming, where one developer writes code while the other reviews it in real-time. The customer is actively involved, providing feedback on new features as they are developed. Continuous integration ensures that code changes are frequently integrated and tested, reducing integration issues.

In summary, Scrum, Kanban, and Extreme Programming are three distinct Agile approaches with their own principles and practices. The choice of which method to use depends on the specific needs and characteristics of the project and the team's preferences. Each of these methods aims to improve collaboration, adaptability, and the quality of the delivered software.

Difference between Scrum, Kanban, and XP

1. **Scrum:**
   - **Focus:** Scrum is like running a series of sprints in a relay race.
   - **Iterations:** It divides work into fixed-time periods called sprints, where the team completes a set of tasks.
   - **Roles:** Scrum defines roles like Product Owner, Scrum Master, and Development Team.
   - **Planning:** Sprint planning meetings help decide what to do in each sprint.
   - **Changes:** Scrum allows changes between sprints but not during them.
2. **Kanban:**
   - **Focus:** Kanban is like a continuous flow on an assembly line.
   - **Iterations:** It doesn't have fixed time periods; work flows continuously.
   - **Visual:** Tasks are visualized on a board, moving from "To Do" to "Done."
   - **Limits:** It sets limits on work in progress to avoid bottlenecks.
   - **Changes:** Kanban is flexible and allows changes at any time.
3. **Extreme Programming (XP):**

  - **Focus:** XP is like a buddy system where two people code together.
  - **Practices:** It emphasizes coding practices like test-driven development (TDD) and pair programming.
  - **Customer:** Customers are closely involved, giving feedback regularly.
  - **Changes:** XP is adaptive and welcomes changes at any stage.

In a nutshell:

- Scrum is about structured sprints with predefined roles and plans.
- Kanban is about a continuous flow with visual tracking and work limits.
- XP is about coding practices, customer involvement, and adaptability.

Backlog With Example.?

In software engineering, a "backlog" refers to a prioritized list of work items, tasks, features, or requirements that need to be completed in a software development project. The backlog serves as a dynamic and evolving document that outlines all the work that needs to be done to build or enhance a software product. Here's an example to illustrate what a backlog might look like:

Project: Developing a Task Management Application

Product Backlog:

1. **User Registration:** Users can create accounts to access the application.
2. **User Login:** Users can log in to their accounts securely.
3. **Task Creation:** Users can add new tasks with titles, descriptions, and due dates.
4. **Task Editing:** Users can edit existing task details.
5. **Task Deletion:** Users can delete tasks they no longer need.
6. **Task Priority:** Tasks can be marked with different priority levels (e.g., high, medium, low).
7. **Task Status:** Tasks have statuses like "To Do," "In Progress," and "Done."
8. **User Profile:** Users can view and update their profile information.
9. **Notification System:** Users receive email notifications for upcoming tasks.
10. **Mobile App Support:** Create a mobile version of the application for iOS and Android.

Sprint Backlog (for Sprint 1):

- **User Registration**
- **User Login**
- **Task Creation**

In this example, the "Product Backlog" represents the overall list of features and requirements for the Task Management Application. It includes everything that the development team envisions for the product, listed in priority order based on their importance and potential value to users.

The "Sprint Backlog" represents a subset of items from the Product Backlog selected for a specific sprint (e.g., Sprint 1). The team chooses these items based on their capacity and the sprint's objectives. During Sprint 1, the team will focus on implementing the "User Registration," "User Login," and "Task Creation" features. Once these are completed, they will move on to the next set of items in the Product Backlog, continually refining and updating it as the project progresses.

The backlog is a critical tool in Agile methodologies like Scrum because it provides a clear roadmap of what needs to be done, helps the team prioritize work, and ensures that the most valuable and important features are addressed first. It's a living document that evolves as the project progresses, allowing for adjustments based on feedback, changing requirements, and emerging priorities.

Class Task → Every developer should create product/sprint backlog of the project from the requirements taken from the client. (20 Minutes Task)

Class Task → Add the details from backlog on the Click up project management tool https://clickup.com/teams/project-management (30 Minutes Task)