**DAY 1 - Assignment 2 MRIGANKA PATRA**

**Tasks:**

1. Simulate a simple Agile project scenario (eg, 'To-do List Application').

* Break down the feature into 3-4 user stories.
* Assign each story to a sprint and identify acceptance criteria.

|  |  |  |  |
| --- | --- | --- | --- |
| User Story ID | Description | Assigned Sprint | Acceptance Criteria |
| US01 | I want to add a “NEW TO-DO” item so I can track my tasks | Sprint 1 | - User can add a to-do item with a title.  - The item appears in the list immediately after submission. |
| US02 | I want to mark a to-do item as completed so I can track what’s done. | Sprint 1 | - User can mark items as completed.  - Completed items appear visually distinct. |
| US03 | I want to delete a to-do item so I can remove unnecessary tasks. | Sprint 2 | - User can delete an item.  - Deleted items no longer appear in the list. |
| US04 | As a user, I want to view my to-do list sorted by creation date so I can easily find recent tasks. | Sprint 2 | - List is sorted by newest to oldest by default.  - Sorting persists when navigating the app. |

2. Create a sprint board (can be done in a table format) showing the user stories in stages: To Do, In Progress, Done

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Story ID | Description | To Do | In Progress | Done |
| US01 | Add New Item |  | ✓ |  |
| US02 | Mark item as completed |  |  | ✓ |
| US03 | Delete an item | ✓ |  |  |
| US04 | Sort to-do list | ✓ |  |  |

3. For one of the user stories, write the Functional test cases, Edge test cases, Test data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Expected Result | Test Type | Test Data |
| T01 | Add a to-do item with a valid title | Item is added and appears in the list | Functional | “Buy Groceries” |
| T02 | Add a multiple item sequentially | All items appear in the list in the correct order | Functional | “Call mom”, “Finish Project” |
| T03 | Verify "Add" button is disabled when input is empty | "Add" button remains disabled if no title is entered | Functional | “” (empty input) |
| T04 | Edit an existing to-do item using the "Edit" button | When "Edit" is clicked: - The item becomes editable. - User can update text. - Changes are saved and reflected in the list | Functional | “Call mom” 🡪Edit🡪”Call dad” |
| T05 | Cancel edit without saving | Changes are discarded; original item text is preserved | Functional | “Call mom” 🡪Edit🡪”Call dad” 🡪Cancel |
| T06 | Add a to-do item with 20 characters in title | Item is added successfully or appropriate validation is displayed | Edge | “theiejndhndjfejbkb we kwlwoihiohiofhfhoihoi” |
| T06 | Add a to-do item with only special characters | Item is added or rejected per app validation rules | Edge | “!@#$%^&\*()\_+{}[]<>?” |

4. Identify testing types used at each stage of development (Unit, Integration, System, etc.) and justify their application.

|  |  |  |
| --- | --- | --- |
| Stage | Testing Type | Justification |
| Development (Code level) | Unit Testing | Developers test individual components (e.g., AddItem component, List component) to ensure basic functionality works. |
| Integration | Integration Testing | Ensure that adding an item updates the list and that components interact correctly (form submission updates the list view). |
| System Testing | System Testing | Verify the entire to-do list functionality works end-to-end in the deployed environment (UI + backend). |
| Regression Testing | Regression Testing | Ensure that changes in Sprint 2 (e.g., sorting, delete functionality) do not break adding items. |
| Acceptance Testing | Acceptance Testing | Product Owner / stakeholders verify that adding a to-do item meets the user story acceptance criteria. |

5. Reflect on how Agile methodology helped in managing testing in smaller iterations versus a waterfall model.

|  |  |
| --- | --- |
| Agile Approach | Waterfall Approach |
| Testing is integrated into each sprint, allowing for continuous feedback and quality assurance. | Testing happens after full development is completed, leading to delayed feedback and potential rework. |
| Smaller, testable increments ensure bugs are caught early and can be fixed quickly. | Bugs found late can require major changes across many modules. |
| Testing evolves with changing requirements; adaptability is high. | Late-stage changes are expensive and harder to incorporate. |
| Encourages team collaboration between developers, testers, and stakeholders in real time. | Testing team often works in isolation after development ends |