**1**.Create a Kanban Board to Visualize the Tasks.

• Create Columns for To Do, In-Progress and Done.

• Add Atleast 5 Sample Tasks

• Move the Tasks across the Columns to Simulate the Workflow.

**Aim**

To create a Kanban board to visualize tasks, organize them into columns, and simulate task movement across the workflow.

**Requirements**

1. Computer with internet access.
2. Kanban board software (e.g., Trello, Jira, or Microsoft Planner) or a physical whiteboard with sticky notes.
3. Sample tasks for demonstration.

**Procedure**

**Step 1: Set Up the Kanban Board**

1. Open the Kanban board tool or prepare a physical board.
2. Create three columns labeled:
   * **To Do**
   * **In Progress**
   * **Done**

**Step 2: Add Sample Tasks**

1. Define at least five tasks to be managed on the board:
   * Task 1: Research project requirements.
   * Task 2: Design project layout.
   * Task 3: Develop the application prototype.
   * Task 4: Test the application.
   * Task 5: Prepare final documentation.
2. Place all tasks under the "To Do" column initially.

**Step 3: Simulate the Workflow**

1. Move **Task 1** (Research project requirements) from "To Do" to "In Progress."
2. Once completed, move **Task 1** to "Done."
3. Repeat the process for other tasks:
   * Move **Task 2** to "In Progress" and then to "Done."
   * Follow the same steps for **Task 3**, **Task 4**, and **Task 5.**
4. Ensure tasks progress logically through the columns to reflect their status.

**Step 4: Review the Board**

1. Observe the distribution of tasks across the columns.
2. Verify that all tasks eventually move to the "Done" column, indicating completion.

**Result**

A Kanban board was successfully created with tasks visualized across "To Do," "In Progress," and "Done" columns. The workflow simulation demonstrated effective task tracking and management.

2. Sketch a Simple Prototype of a Bus Ticket Booking System using Figma Tool

**Aim**

To design and develop a simple prototype of a bus ticket booking system using the Figma tool, focusing on user-friendly and intuitive interfaces.

**Requirements**

1. **Software:**
   * Figma (Online or Desktop Application)
   * Web Browser (for accessing Figma if using online)
2. **Skills:**
   * Basic understanding of UI/UX principles
   * Familiarity with Figma tools and components
3. **Hardware:**
   * Computer or laptop with internet connectivity
4. **Features to Implement:**
   * Home Screen with navigation options
   * Bus search interface (source, destination, date, time)
   * Display of available buses with details (bus type, timings, fare)
   * Seat selection interface
   * Payment confirmation screen

**Procedure**

1. **Set Up Figma:**
   * Open Figma and create a new project file.
   * Name the project "Bus Ticket Booking System."
2. **Create Wireframes:**
   * Design low-fidelity wireframes for each interface screen:
     + Home Screen: Include a search bar and navigation buttons.
     + Bus Search Screen: Add dropdowns for source and destination, a calendar for date selection, and a search button.
     + Results Screen: Display a list of available buses with details like time, fare, and bus type.
     + Seat Selection Screen: Show a graphical representation of bus seats for selection.
     + Payment Screen: Include fields for payment details and a confirmation button.
3. **Design the Prototype:**
   * Use Figma's design tools to add colors, fonts, and interactive components.
   * Ensure a consistent theme across all screens (e.g., color scheme, button styles).
   * Link the screens to create a clickable prototype for navigation.
4. **Test the Prototype:**
   * Preview the prototype in Figma.
   * Simulate user interactions such as searching for buses, selecting seats, and completing the booking.
5. **Save and Share:**
   * Save the project in Figma.
   * Export the prototype as a PDF or share the Figma link for review.

**Result**

A functional prototype of a bus ticket booking system was successfully designed using Figma. The prototype includes screens for searching buses, selecting seats, and confirming payment, providing a seamless user experience.

3. The stakeholders have conflicting views on the user interface design for an E-Commerce mobile app. Create a prototype using Figma tool to discuss with the stakeholders to get their feedback and approval.

**Aim**

To design a prototype of an E-Commerce mobile app using the Figma tool to gather feedback and approval from stakeholders with conflicting views on the user interface design.

**Requirements**

1. **Software:**
   * Figma (Online or Desktop Application)
   * Web Browser (if using Figma online)
2. **Skills:**
   * Knowledge of UI/UX principles
   * Familiarity with Figma's tools and features
3. **Hardware:**
   * A computer or laptop with internet connectivity
4. **Features to Include:**
   * Home screen with product categories and search bar
   * Product listing screen with filters and sorting options
   * Product detail page with "Add to Cart" and "Buy Now" buttons
   * Shopping cart page with item summary and total cost
   * Checkout screen with payment and delivery details

**Procedure**

1. **Understand Stakeholder Requirements:**
   * Gather input from stakeholders to identify key design elements and features they prioritize.
   * Document conflicting views to address them during the design phase.
2. **Set Up Figma:**
   * Open Figma and create a new project titled "E-Commerce Mobile App Prototype."
   * Define the screen dimensions for a mobile device (e.g., 375x812 px for iPhone).
3. **Design Wireframes:**
   * Sketch low-fidelity wireframes for each app screen:
     + **Home Screen:** Include a header with a search bar, product categories, and promotional banners.
     + **Product Listing Screen:** Show a grid or list of products with filter and sort options.
     + **Product Detail Screen:** Add product images, descriptions, price, and action buttons (e.g., "Add to Cart").
     + **Shopping Cart Screen:** Display items in the cart, quantity adjustments, and total cost.
     + **Checkout Screen:** Include fields for payment and delivery details.
4. **Create High-Fidelity Mockups:**
   * Add colors, fonts, and interactive elements to the wireframes using Figma's design tools.
   * Apply consistent design patterns and themes for a professional look.
   * Incorporate stakeholder-specific preferences where feasible.

**5.Link Screens for Prototyping:**

* + Use Figma’s prototyping feature to link the screens for seamless navigation.
  + Simulate user interactions such as adding items to the cart, navigating product pages, and completing a purchase.

**6.Present to Stakeholders:**

* + Share the prototype link or export it as a PDF for stakeholders to review.
  + Collect feedback on the design and functionality of the prototype.

**7.Iterate Based on Feedback:**

* + Make adjustments to the prototype based on stakeholder input.
  + Resolve conflicting views by highlighting the rationale behind specific design choices.

**Result**

A high-fidelity prototype of the E-Commerce mobile app was successfully designed using Figma. The prototype serves as a tool for stakeholders to provide feedback, resolve conflicting views, and approve the final user interface design.

4. Create a Scrum Project in Jira. • Add a backlog with at least 5 items (e.g., "Create user registration page", "Develop API for login"). • Prioritize the backlog and create a 1-week sprint. • Move backlog items into the sprint and start the sprint. • Finally show the Screenshot of the sprint board at the start and end of the sprint.

**Aim**

To create and manage a Scrum project in Jira by adding backlog items, prioritizing them, creating a sprint, and monitoring its progress using a sprint board.

**Requirements**

1. **Software:**
   * Jira Software (Cloud or Server version)
2. **Skills:**
   * Basic understanding of Agile Scrum methodology
   * Familiarity with Jira's interface
3. **Hardware:**
   * A computer or laptop with internet access
4. **Key Features to Implement:**
   * Backlog creation with at least 5 items
   * Prioritization of backlog
   * Sprint creation and management
   * Screenshot of the sprint board at the start and end of the sprint

**Procedure**

**1. Set Up Jira Project:**

* Log in to Jira using your credentials.
* Create a new Scrum project:
  + Click on **"Create Project"**.
  + Select **"Scrum"** from the project templates.
  + Name the project (e.g., "E-Commerce Development").

**2. Add Backlog Items:**

* Navigate to the **Backlog** tab.
* Add at least 5 backlog items (user stories or tasks) with clear descriptions:
  + Example backlog items:
    1. Create user registration page.
    2. Develop API for login functionality.
    3. Design product listing page.
    4. Implement shopping cart feature.
    5. Test and debug checkout process.

**3. Prioritize the Backlog:**

* Drag and drop the backlog items to prioritize them based on importance and dependencies.
* Ensure the most critical tasks are at the top.

**4. Create a Sprint:**

* Click **"Create Sprint"** in the Backlog view.
* Name the sprint (e.g., "Sprint 1").
* Set the sprint duration to 1 week.

**5. Move Backlog Items into the Sprint:**

* Drag the top-priority items from the backlog into the sprint.
* Ensure the total workload aligns with the team's capacity.

**6. Start the Sprint:**

* Click **"Start Sprint"**.
* Add a sprint goal (e.g., "Develop and test basic user flows for the E-Commerce app").

**7. Monitor the Sprint:**

* Use the **Sprint Board** to track the progress of tasks.
* Move tasks through the columns (To Do, In Progress, Done) as they are worked on.

**8. End the Sprint:**

* After 1 week, complete the sprint by clicking **"Complete Sprint"**.
* Review the sprint summary to evaluate completed and incomplete tasks.

**9. Take Screenshots:**

* Capture the **Sprint Board** at the start of the sprint (with all tasks in "To Do").
* Capture the **Sprint Board** at the end of the sprint (with tasks in their respective columns).

**Result**

The Scrum project was successfully created and managed in Jira. A sprint was initiated, backlog items were prioritized and added, and progress was tracked on the sprint board. Screenshots at the start and end of the sprint demonstrate the workflow and task completion.

5. Use the following requirements for a Library Management System: • Add a feature to search books by title and author. • Implement an online book reservation system. • Generate monthly reports on borrowed books for administrators. • Enable email notifications for overdue books. • Add support for QR code scanning for borrowing and returning books. • Create a user-friendly dashboard for librarians. • Allow users to review and rate books. • Integrate a chatbot for user assistance. • Develop a mobile app version of the system. • Provide multi-language support. Categorize each requirement using MOSCOW Method (Must-Have, Should-Have, Could-Have, or Won’t-Have) based on the following criteria: • Impact on the users and stakeholders. • Feasibility considering time, budget, and resource constraints. Finally Submit the completed Google Sheet or Excel file with all requirements categorized and justified.

**Aim**

To categorize the requirements of a Library Management System using the MOSCOW Method based on their impact on users, stakeholders, and feasibility, and to document the categorization in a spreadsheet.

**Requirements**

1. **Software:**
   * Google Sheets or Microsoft Excel
2. **Skills:**
   * Understanding of the MOSCOW Method
   * Ability to evaluate feasibility and impact
3. **Hardware:**
   * A computer or laptop with spreadsheet software and internet access
4. **Features to Categorize:**
   * Search books by title and author
   * Online book reservation system
   * Monthly reports on borrowed books
   * Email notifications for overdue books
   * QR code scanning for borrowing and returning
   * User-friendly dashboard for librarians
   * User reviews and ratings for books
   * Chatbot for user assistance
   * Mobile app version
   * Multi-language support

**Procedure**

**1. Understand the Requirements:**

* Review each feature and its purpose.
* Consider the needs of users (e.g., library members, librarians) and stakeholders (e.g., administrators).

**2. Define MOSCOW Categories:**

* **Must-Have:** Essential for the system to function.
* **Should-Have:** Important but not critical; workarounds are possible.
* **Could-Have:** Desirable but not necessary; adds value if included.
* **Won’t-Have (for now):** Out of scope due to constraints.

**3. Analyze Impact and Feasibility:**

* Evaluate the impact of each feature on users and stakeholders.
* Assess feasibility based on time, budget, and resource constraints.

**4. Categorize Requirements:**

* Open Google Sheets or Excel.
* Create columns for **Requirement**, **MOSCOW Category**, and **Justification**.
* Add each requirement to the sheet and assign a category with a brief justification.

**5. Document the Justifications:**

* Example:
  + **Requirement:** Search books by title and author
  + **Category:** Must-Have
  + **Justification:** Core functionality for users to locate books easily.

**6. Finalize the Spreadsheet:**

* Review the categorizations to ensure consistency and accuracy.
* Save the file with an appropriate name (e.g., "Library\_System\_MOSCOW.xlsx").

**7. Submit the Spreadsheet:**

* Share the completed file via email or upload it to the designated platform.

**Result**

The requirements of the Library Management System were successfully categorized using the MOSCOW Method. A spreadsheet was created to document the categories and justifications, aiding in prioritization and decision-making for development.

6. Link Jira tasks with Confluence to streamline task tracking and progress monitoring for the Library Management System development. • Create a new page in Confluence titled "Library Management System Project Overview." • Embed at least 5 Jira issues related to the development of the Library Management System (e.g., tasks from the sprint like "Develop book search functionality," "Create user login page," etc.). • Use the Jira macro to display issues with status (e.g., "To Do," "In Progress," "Done"). • Add a progress bar in the Confluence page to visually track the completion of each embedded Jira task (e.g., percentage of tasks completed in the sprint). • Submit a screenshot of the Confluence page showing the embedded Jira tasks and the progress bar.

**Aim**

To create a Confluence page titled "Library Management System Project Overview" and embed Jira issues related to the development of the Library Management System for streamlined task tracking and progress monitoring.

**Requirements**

1. **Software:**
   * Jira Software
   * Confluence
2. **Skills:**
   * Basic understanding of Jira and Confluence integration
   * Familiarity with the Jira macro in Confluence
3. **Hardware:**
   * A computer or laptop with internet connectivity
4. **Features to Implement:**
   * Embed Jira issues on a Confluence page
   * Display issue statuses (e.g., "To Do," "In Progress," "Done")
   * Add a progress bar to track task completion

**Procedure**

**1. Set Up Confluence Page:**

* Log in to Confluence.
* Create a new page titled **"Library Management System Project Overview"**.
* Add a brief description of the project at the top of the page.

**2. Identify Jira Issues:**

* Log in to Jira.
* Identify at least 5 tasks related to the Library Management System (e.g., "Develop book search functionality," "Create user login page," etc.).
* Ensure the tasks are part of a sprint or backlog with statuses updated (e.g., "To Do," "In Progress," "Done").

**3. Embed Jira Issues in Confluence:**

* On the Confluence page, click **"Insert"** (the "+" icon) and select **"Jira Issue/Filter"**.
* Use one of the following methods to embed Jira issues:
  + **Search for Issues:** Search for specific Jira tasks by keywords or project name.
  + **Enter JQL Query:** Use Jira Query Language to filter tasks (e.g., project = LMS AND sprint = "Sprint 1").
* Select the issues to embed and click **"Insert"**.
* Ensure the embedded issues display their statuses.

**4. Add a Progress Bar:**

* Add a **Progress Bar Macro** to the Confluence page:
  + Type /progress bar and select the macro.
  + Set the percentage based on the completion of tasks in Jira (e.g., if 3 out of 5 tasks are done, set the progress bar to 60%).
* Label the progress bar (e.g., "Sprint 1 Progress").

**5. Format the Page:**

* Arrange the Jira issues and progress bar neatly on the page.
* Add headers for clarity, such as:
  + **Project Description**
  + **Jira Tasks**
  + **Sprint Progress**

**6. Take a Screenshot:**

* Capture the Confluence page showing:
  + Embedded Jira issues with their statuses
  + The progress bar indicating task completion percentage

**7. Submit the Screenshot:**

* Save the screenshot with an appropriate name (e.g., "Confluence\_LMS\_Screenshot.png").
* Submit the screenshot as per instructions (e.g., via email or upload).

**Result**

A Confluence page titled **"Library Management System Project Overview"** was successfully created, embedding Jira tasks and tracking progress using a visual progress bar. The integration streamlines task monitoring and enhances team collaboration.

7. You are designing a Task Management System for a small team. The system should include the following features: 1.User Login and Role Assignment 2.Task Creation and Assignment 3.Task Prioritization and Deadlines 4.Progress Tracking and Reporting Prioritize these requirements using the MoSCoW and Kano models in Jira.

**Aim**

To prioritize the requirements of a Task Management System for a small team using the MoSCoW and Kano models in Jira for effective project planning and execution.

**Requirements**

1. **Software:**
   * Jira Software
   * Spreadsheet or document editor for documentation
2. **Skills:**
   * Understanding of MoSCoW and Kano prioritization models
   * Familiarity with Jira task creation and labeling
3. **Hardware:**
   * A computer or laptop with internet connectivity
4. **Features to Prioritize:**
   * User Login and Role Assignment
   * Task Creation and Assignment
   * Task Prioritization and Deadlines
   * Progress Tracking and Reporting

**Procedure**

**Step 1: Understand the Prioritization Models**

1. **MoSCoW Model:**
   * **Must-Have:** Essential features required for the system to function.
   * **Should-Have:** Important but not critical features.
   * **Could-Have:** Desirable but not necessary features.
   * **Won’t-Have:** Features not included in this phase.
2. **Kano Model:**
   * **Basic Needs (Must-Haves):** Features users expect; absence causes dissatisfaction.
   * **Performance Needs (Should-Haves):** Features that improve satisfaction when delivered.
   * **Excitement Needs (Could-Haves):** Features that delight users but are not expected.

**Step 2: Create Tasks in Jira**

* Log in to Jira and create a new project titled **"Task Management System Development"**.
* Create tasks for each feature:
  1. User Login and Role Assignment
  2. Task Creation and Assignment
  3. Task Prioritization and Deadlines
  4. Progress Tracking and Reporting

**Step 3: Apply MoSCoW Prioritization in Jira**

* Add a custom field or label in Jira called **"MoSCoW Priority"**.
* Assign priorities to tasks:
  + **Must-Have:** User Login and Role Assignment, Task Creation and Assignment
  + **Should-Have:** Task Prioritization and Deadlines
  + **Could-Have:** Progress Tracking and Reporting
  + **Won’t-Have:** None for this phase.

**Step 4: Apply Kano Prioritization in Jira**

* Add another custom field or label called **"Kano Priority"**.
* Categorize tasks based on Kano model:
  + **Basic Needs:** User Login and Role Assignment
  + **Performance Needs:** Task Creation and Assignment, Task Prioritization and Deadlines
  + **Excitement Needs:** Progress Tracking and Reporting

**Step 5: Visualize Prioritization**

* Use Jira’s board or filter features to create views for:
  + Tasks categorized by **MoSCoW Priority**.
  + Tasks categorized by **Kano Priority**.

**Step 6: Document Prioritization**

* Export the Jira tasks and their prioritization labels into a spreadsheet or Confluence page for documentation.
* Include the following columns:
  + **Feature**
  + **MoSCoW Priority**
  + **Kano Priority**
  + **Justification**

**Step 7: Review and Finalize**

* Share the prioritization with stakeholders for feedback.
* Make adjustments based on team discussions and resource constraints.

**Result**

The requirements for the Task Management System were successfully prioritized using the MoSCoW and Kano models in Jira. The prioritized tasks were documented and visualized, aiding in efficient project planning.

8. You are tasked with developing an Online Learning Platform. The platform should include the following functionalities: 1. Course Enrollment and Registration 2. Video Lecture Streaming 3. Interactive Quizzes and Assignments 4. Progress Tracking Dashboard 5. Peer-to-Peer Discussion Forums 6. Certificate Generation Use Jira to categorize and prioritize these requirements using MoSCoW and Kano techniques.

**Aim**

To categorize and prioritize the requirements of an Online Learning Platform using the MoSCoW and Kano techniques in Jira for effective planning, development, and delivery.

**Requirements**

1. **Software:**
   * Jira Software (Cloud or Server version)
   * Spreadsheet or document editor for documentation (optional)
2. **Skills:**
   * Understanding of MoSCoW and Kano prioritization models
   * Familiarity with Jira task creation, labels, and filters
3. **Hardware:**
   * A computer or laptop with internet connectivity
4. **Features to Prioritize:**
   * Course Enrollment and Registration
   * Video Lecture Streaming
   * Interactive Quizzes and Assignments
   * Progress Tracking Dashboard
   * Peer-to-Peer Discussion Forums
   * Certificate Generation

Here's a structured lab manual for categorizing and prioritizing the requirements of an **Online Learning Platform** using the **MoSCoW** and **Kano** models in **Jira**:

**9) Categorizing and Prioritizing Requirements for an Online Learning Platform using MoSCoW and Kano Models**

**Aim**

To categorize and prioritize the requirements of an Online Learning Platform using the MoSCoW and Kano techniques in Jira for effective planning, development, and delivery.

**Requirements**

1. **Software:**
   * Jira Software (Cloud or Server version)
   * Spreadsheet or document editor for documentation (optional)
2. **Skills:**
   * Understanding of MoSCoW and Kano prioritization models
   * Familiarity with Jira task creation, labels, and filters
3. **Hardware:**
   * A computer or laptop with internet connectivity
4. **Features to Prioritize:**
   * Course Enrollment and Registration
   * Video Lecture Streaming
   * Interactive Quizzes and Assignments
   * Progress Tracking Dashboard
   * Peer-to-Peer Discussion Forums
   * Certificate Generation

**Procedure**

**Step 1: Understand the Prioritization Models**

1. **MoSCoW Model:**
   * **Must-Have:** Essential features for the platform to function properly.
   * **Should-Have:** Important features but not critical for launch.
   * **Could-Have:** Features that would enhance the user experience but are not essential.
   * **Won’t-Have:** Features that are not part of the current scope.
2. **Kano Model:**
   * **Basic Needs (Must-Haves):** Features users expect; absence of these causes dissatisfaction.
   * **Performance Needs (Should-Haves):** Features that improve satisfaction when delivered.
   * **Excitement Needs (Could-Haves):** Features that delight users but are not expected.

**Step 2: Create Tasks in Jira**

* Log in to Jira and create a new project titled **"Online Learning Platform Development"**.
* Create tasks for each feature:
  1. Course Enrollment and Registration
  2. Video Lecture Streaming
  3. Interactive Quizzes and Assignments
  4. Progress Tracking Dashboard
  5. Peer-to-Peer Discussion Forums
  6. Certificate Generation

**Step 3: Apply MoSCoW Prioritization in Jira**

* Add a custom field or label in Jira called **"MoSCoW Priority"**.
* Assign priorities to tasks based on the MoSCoW model:
  + **Must-Have:**
    - Course Enrollment and Registration (critical for users to access courses)
    - Video Lecture Streaming (core feature for content delivery)
  + **Should-Have:**
    - Interactive Quizzes and Assignments (important for learner engagement)
    - Progress Tracking Dashboard (key for students to track their progress)
  + **Could-Have:**
    - Peer-to-Peer Discussion Forums (enhances engagement but not critical)
    - Certificate Generation (valuable but not necessary for course completion)
  + **Won’t-Have:**
    - None (all features are important for the platform)

**Step 4: Apply Kano Prioritization in Jira**

* Add another custom field or label called **"Kano Priority"**.
* Categorize tasks based on the Kano model:
  + **Basic Needs (Must-Haves):**
    - Course Enrollment and Registration (expected by users to access the platform)
    - Video Lecture Streaming (essential for course delivery)
  + **Performance Needs (Should-Haves):**
    - Interactive Quizzes and Assignments (directly impacts user satisfaction)
    - Progress Tracking Dashboard (improves user experience and engagement)
  + **Excitement Needs (Could-Haves):**
    - Peer-to-Peer Discussion Forums (delights users, adds community element)
    - Certificate Generation (adds value but is not a core requirement)

**Step 5: Visualize Prioritization**

* Use Jira’s board or filter features to create views for:
  + Tasks categorized by **MoSCoW Priority**.
  + Tasks categorized by **Kano Priority**.
* Create a filter or board view for each priority category to track tasks accordingly.

**Step 6: Document Prioritization**

* Export the Jira tasks and their labels into a spreadsheet or Confluence page for documentation.
* Include the following columns:
  + **Feature**
  + **MoSCoW Priority**
  + **Kano Priority**
  + **Justification** (brief explanation of why the task was categorized as such)

**Step 7: Review and Finalize**

* Share the prioritization with stakeholders (e.g., project managers, developers) for feedback.
* Adjust priorities based on team discussions, available resources, and time constraints.

**Result**

The requirements for the Online Learning Platform were successfully categorized and prioritized using the MoSCoW and Kano models in Jira. The tasks were documented, and their priorities were visualized, aiding in efficient project planning and decision-making.