**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.

Steps to do it:

To create a Kanban board for task management, we can break it down into the following steps:

1. Understand the Kanban Board Structure

A Kanban board typically has three columns (also known as lanes) to represent different stages of the task workflow:

* To Do: Tasks that need to be done.
* In-Progress: Tasks that are actively being worked on.
* Done: Tasks that have been completed.

2. Choose Tools for Creating the Board

A Kanban board can be created using various methods. Some popular tools include:

* Trello (web-based)
* Jira (for software development)
* Microsoft Planner (if you're using Microsoft 365)
* Physical Kanban Board (using sticky notes or cards on a whiteboard)

For simplicity, we will describe how to use a physical board or any virtual tool where you can drag and drop cards/tasks.

3. Create the Kanban Columns

For this example, we will create three main columns:

* To Do: Where we add tasks that need to be completed.
* In-Progress: Where tasks that are actively being worked on will go.
* Done: Where tasks that have been completed will be moved.

These columns are placed next to each other in a linear format to visualize the flow of tasks.

4. Add Sample Tasks

Now, let's define at least 5 sample tasks. Each task will be represented by a card that contains the task name and possibly additional details.

Sample Tasks:

* Task 1: "Complete monthly report"
* Task 2: "Fix bug in website login page"
* Task 3: "Prepare presentation for the meeting"
* Task 4: "Respond to client emails"
* Task 5: "Update project documentation"

Each of these tasks will initially be placed in the To Do column because none of them have been started yet.

5. Simulate the Workflow

Now, we'll move these tasks through the columns to simulate their workflow:

Step 1: Start in "To Do" Column

All five tasks begin in the To Do column because none have been worked on yet. Here's what the board looks like at the beginning:

To Do

----------------------------

| Complete monthly report |

| Fix bug in website login |

| Prepare presentation |

| Respond to client emails |

| Update project documentation|

----------------------------

Step 2: Move Tasks to "In-Progress"

Now, let's say that we start working on a few tasks. We can move some tasks from the To Do column to the In-Progress column. For example, we start working on:

* "Fix bug in website login page"
* "Respond to client emails"

The board now looks like this:

To Do In-Progress

----------------------- ----------------------------

| Complete monthly report | Fix bug in website login |

| Prepare presentation | Respond to client emails |

| Update project doc | |

----------------------- ----------------------------

Step 3: Move Tasks to "Done"

Once tasks are completed, we move them to the Done column. For example:

* "Respond to client emails" is completed, so it moves to Done.
* "Fix bug in website login page" is also completed, so it moves to Done.

The board now looks like this:

To Do In-Progress Done

----------------------- ------------------ -------------------------

| Complete monthly report | | Fix bug in website login |

| Prepare presentation | | Respond to client emails |

| Update project doc | | |

----------------------- ------------------ -------------------------

Step 4: Continue Moving Tasks

As work continues, the other tasks are moved across the board:

* Once "Complete monthly report" is started, it moves to In-Progress.
* When it is completed, it moves to Done.
* Similarly, other tasks like "Prepare presentation" and "Update project documentation" move through the same process.

After all tasks are completed, the board will look like this:

To Do In-Progress Done

----------------------- ------------------ -------------------------

| | | Fix bug in website login |

| | | Respond to client emails |

| | | Complete monthly report |

| | | Prepare presentation |

| | | Update project doc |

----------------------- ------------------ -------------------------

6. Final Review

At this point, all tasks have moved from To Do to In-Progress and then to Done. The Kanban board now visually shows the progress of each task and allows you to see which tasks are completed, which are in progress, and which are yet to start.

7. Advantages of the Kanban Board

* Visualization: Helps you visualize the workflow and progress of tasks.
* Prioritization: By moving tasks around, you can quickly identify what's left to do, what's being worked on, and what’s completed.
* Workload Management: It prevents overloading the system by showing tasks that are "in-progress" at any given time.

8. Next Steps

* Add more tasks as needed.
* Set WIP (Work In Progress) limits to prevent overloading the In-Progress column.
* Review and update tasks as they move through the board.

This is the basic process of creating and using a Kanban board to visualize and manage tasks effectively.

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.

**Steps to do it:**

**Creating a simple prototype of a Bus Ticket Booking System using Figma involves several steps. In this guide, I will walk you through each stage of the process. We'll focus on designing the key screens for a bus ticket booking system and then simulate user interaction through a prototype.**

**Step-by-Step Guide to Creating a Bus Ticket Booking System in Figma:**

**Step 1: Create a New Figma Project**

1. **Sign in to Figma: Go to** [**Figma**](https://www.figma.com/) **and sign in or create an account if you don't have one already.**
2. **Create a New File:** 
   * **Once logged in, click on the New File button to create a blank canvas where you'll design your project.**
   * **Give the project a name such as "Bus Ticket Booking System."**

**Step 2: Design the First Screen – Home Screen**

**This will be the landing page where the user will start their ticket booking journey.**

**2.1. Create a Frame for the Home Screen**

* **Click the Frame Tool (F) in the top bar to create the mobile or desktop frame. Let’s start with a mobile screen size (e.g., iPhone 13, 375x812 px).**
* **Add a frame for your home screen.**

**2.2. Add UI Elements to the Home Screen**

**On the home screen, we will have several key components:**

* **Search Bar: For the user to search for bus routes.**
* **Bus Icon/Logo: To represent the bus booking service.**
* **Departure and Arrival Locations: Input fields for departure and arrival cities.**
* **Departure Date: Input field or a date picker for selecting the date.**
* **Search Button: Button to start the search process.**

**To add these elements:**

1. **Text: Use the text tool (T) to add headings like "Bus Ticket Booking" or "Search for buses."**
2. **Shapes: Use shapes (rectangle, circle, etc.) to create the input fields for departure and arrival cities.** 
   * **Rectangle tool for input fields (for departure city, arrival city, and date).**
   * **Add icons (e.g., magnifying glass for the search) from the Figma Assets panel.**
3. **Button: Create a Search button using a rounded rectangle (width: 200 px, height: 50 px) and label it "Search."**

**Now, the home screen layout should look something like this:**

**+--------------------------------------------------+**

**| [Logo] Bus Ticket Booking |**

**|--------------------------------------------------|**

**| Departure City [\_\_] |**

**| Arrival City [\_\_] |**

**| Departure Date [\_\_] |**

**| [Search Button] |**

**+--------------------------------------------------+**

**Step 3: Design the Second Screen – Bus Search Results**

**Once the user clicks the "Search" button, they will be directed to the search results page that displays a list of available buses.**

**3.1. Create a New Frame for the Search Results Screen**

* **Use the Frame Tool (F) to create a new frame for the search results screen.**
* **Keep the same mobile size as before (e.g., iPhone 13).**

**3.2. Add UI Elements to the Search Results Screen**

1. **List of Buses:** 
   * **Use rectangles to create each bus result card. The card should have the following information:** 
     + **Bus name or operator.**
     + **Departure time.**
     + **Arrival time.**
     + **Price.**
   * **Add icons for bus logos or use placeholder text like "Bus Name."**
2. **Filter Options: At the top of the screen, add a dropdown or button for filters (e.g., to filter by price or bus operator).**
3. **Select Button: Each bus card will have a Select button that allows the user to choose their preferred bus.**

**Now, the search results layout should look like this:**

**+--------------------------------------------------+**

**| [Filter Options] |**

**|--------------------------------------------------|**

**| Bus Name Departure Time Price |**

**| [Select Button] $20 |**

**|--------------------------------------------------|**

**| Bus Name Departure Time Price |**

**| [Select Button] $25 |**

**|--------------------------------------------------|**

**| Bus Name Departure Time Price |**

**| [Select Button] $30 |**

**+--------------------------------------------------+**

**Step 4: Design the Third Screen – Passenger Details**

**Once the user selects a bus, they will be directed to the passenger details page where they will input their details.**

**4.1. Create a New Frame for the Passenger Details Screen**

* **Use the Frame Tool (F) to create a new frame for the passenger details page.**

**4.2. Add UI Elements to the Passenger Details Screen**

1. **Text Fields:** 
   * **Use rectangles to represent text input fields for passenger name, contact number, and email.**
2. **Seat Selection:** 
   * **Create an area where the user can select their seat (e.g., using circles for seats). Represent available seats in green and booked seats in red.**
3. **Continue Button:** 
   * **Create a button at the bottom labeled "Continue to Payment."**

**Now, the layout for passenger details will look like this:**

**+--------------------------------------------------+**

**| Name [\_\_\_\_\_\_\_\_\_\_\_\_\_\_] |**

**| Contact Number [\_\_\_\_\_\_\_\_\_\_\_\_\_\_] |**

**| Email [\_\_\_\_\_\_\_\_\_\_\_\_\_\_] |**

**|--------------------------------------------------|**

**| [Seat 1] [Seat 2] [Seat 3] [Seat 4] |**

**|--------------------------------------------------|**

**| [Continue to Payment] |**

**+--------------------------------------------------+**

**Step 5: Design the Fourth Screen – Payment**

**The payment screen allows the user to enter payment details to complete their booking.**

**5.1. Create a New Frame for the Payment Screen**

* **Use the Frame Tool (F) again to create the final frame for the payment page.**

**5.2. Add UI Elements to the Payment Screen**

1. **Card Details: Use input fields for card number, expiration date, and CVV.**
2. **Confirm Payment Button: A button to confirm the payment at the bottom of the screen, labeled "Confirm Payment."**
3. **Summary Section: At the top, add a summary of the bus ticket, including bus name, seat number, and price.**

**Now, the layout for payment will look like this:**

**+--------------------------------------------------+**

**| Bus Name Seat Number Price |**

**|--------------------------------------------------|**

**| Card Number [\_\_\_\_\_\_\_\_\_\_\_\_\_\_] |**

**| Expiry Date [\_\_ / \_\_] |**

**| CVV [\_\_\_] |**

**|--------------------------------------------------|**

**| [Confirm Payment] |**

**+--------------------------------------------------+**

**Step 6: Create Interactions and Prototype**

**Now, we will add the interactivity to simulate the flow of the booking process.**

1. **Create Prototypes: Select an element (e.g., "Search" button), click the Prototype tab on the right panel, and drag an arrow to the next screen you want to navigate to.** 
   * **For instance, when the user clicks "Search," link it to the Search Results Screen.**
   * **Link the "Select" button to the Passenger Details Screen.**
   * **Link the "Continue to Payment" button to the Payment Screen.**
2. **Test the Prototype: After connecting all the screens, click the Play button (the triangle in the top right corner) to preview your prototype. This will allow you to simulate the entire booking journey from home screen to payment confirmation.**

**Step 7: Final Review and Sharing**

1. **Review the Prototype: Check if all interactions are working as expected.**
2. **Sharing the Prototype: Click the Share button in the top-right corner of Figma to share the design with stakeholders or team members. You can also copy the link to allow others to interact with the prototype.**

**Conclusion:**

**By following these steps, you have successfully created a simple bus ticket booking system in Figma. This prototype includes the major screens a user would interact with, from searching for buses to completing a payment. You can now refine the design further, add more features, and continue testing the 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.

**Steps to do it:**

**Creating a prototype for an E-Commerce mobile app using Figma to resolve conflicting views among stakeholders involves designing key screens that represent the user interface (UI) and enabling interactivity to simulate the app experience. The goal is to get feedback on the design elements, functionality, and flow of the app. I'll guide you step by step through the process.**

**Step-by-Step Guide for Prototyping an E-Commerce Mobile App in Figma:**

**Step 1: Sign in to Figma and Create a New Project**

1. **Sign in to Figma:** 
   * **Go to** [**Figma**](https://www.figma.com/) **and sign in using your account credentials or create a new one if you don’t already have one.**
2. **Create a New File:** 
   * **Once logged in, click on the New File button to create a new design file.**
   * **Name your project something like "E-Commerce Mobile App Prototype".**

**Step 2: Set Up Your Canvas for Mobile Screen Design**

1. **Frame Tool:**
   * **Use the Frame Tool (F) to create a mobile screen frame. Choose a mobile frame size (e.g., iPhone 13, 375x812 px) for your prototype.**
   * **Select a size suitable for your target platform (e.g., iOS or Android).**
2. **Workspace Setup:**
   * **Set up a grid or layout for consistency. You can add a 4-column grid or a layout guide to help with the spacing of UI elements, especially for mobile designs.**

**Step 3: Design the Key Screens for Your E-Commerce App**

**You'll need several key screens for the app prototype. Let's break it down by the most critical screens of an E-Commerce app:**

**3.1 Home Screen Design**

**The Home Screen is the first screen that users will see, displaying products, categories, and promotions.**

1. **Header:**
   * **Create a search bar at the top for users to search for products (use a rectangle for the search bar and add a search icon).**
   * **Add a logo on the left side of the header.**
2. **Main Content:**
   * **Create a carousel banner at the top for promotions or featured products. You can use a rectangle to represent the banner and add placeholder images for the promotions.**
   * **Create product categories (e.g., Men’s Wear, Women’s Wear, Electronics). Use simple icons or images and text labels to categorize products.**
3. **Products Grid:**
   * **Below the categories, create a grid layout for displaying featured products. Each product will have a rectangular container with:** 
     + **Product image (placeholder).**
     + **Product name.**
     + **Product price.**
     + **A button labeled "Add to Cart".**

**The layout for the Home Screen might look something like this:**

**+-------------------------------------------------+**

**| [Logo] Search Bar [🔍] |**

**+-------------------------------------------------+**

**| [Promo Banner] |**

**+-------------------------------------------------+**

**| Categories: Men's | Women's | Electronics |**

**+-------------------------------------------------+**

**| Product Image Product Name $Price [Add to Cart] |**

**| Product Image Product Name $Price [Add to Cart] |**

**+-------------------------------------------------+**

**3.2 Product Details Screen**

**Once a user selects a product, they should be able to view more detailed information.**

1. **Product Image:**
   * **Create a large product image at the top.**
2. **Product Information:**
   * **Below the image, show the product name, description, and price.**
   * **Create a quantity selector for users to choose how many items they want to purchase.**
3. **Add to Cart Button:**
   * **Below the product information, add a prominent "Add to Cart" button.**

**The product details screen might look like this:**

**+-----------------------------------------------+**

**| [Product Image] |**

**| Product Name |**

**| Description of the product |**

**| Price: $Price |**

**| Quantity [–][1][+] |**

**| [Add to Cart Button] |**

**+-----------------------------------------------+**

**3.3 Cart Screen**

**The Cart Screen will allow users to review their selected items and proceed to checkout.**

1. **Product List:** 
   * **Each product in the cart will be displayed with an image, name, quantity, price, and a remove item button.**
2. **Total Price:** 
   * **Below the product list, display the total cost for all items in the cart.**
3. **Checkout Button:** 
   * **A "Proceed to Checkout" button should be at the bottom of the cart.**

**The layout might look like this:**

**+------------------------------------------------+**

**| [Product Image] Product Name $Price Quantity |**

**| [Product Image] Product Name $Price Quantity |**

**+------------------------------------------------+**

**| Total Price: $Total Amount |**

**| [Proceed to Checkout] |**

**+------------------------------------------------+**

**3.4 Checkout Screen**

**This screen will allow the user to input their shipping and payment details.**

1. **Shipping Information:**
   * **Add input fields for the shipping address (Name, Address, City, Zip Code).**
2. **Payment Information:**
   * **Input fields for credit card information (Card number, Expiry, CVV).**
3. **Confirm Purchase Button:**
   * **A "Confirm Purchase" button at the bottom of the page.**

**Step 4: Adding Interactivity and Prototyping**

**Figma’s Prototyping feature allows you to link different screens together to simulate user flow.**

1. **Create Prototype Links:** 
   * **Go to the Prototype tab in the right sidebar after selecting an element (e.g., a button or product).**
   * **Link the "Search" button to the Home Screen or to another screen.**
   * **Link the "Add to Cart" button on the product details page to the Cart Screen.**
   * **Link the "Proceed to Checkout" button to the Checkout Screen.**
2. **Transitions and Animations:** 
   * **Add transitions between screens (e.g., slide, fade, etc.) to create a more realistic prototype experience.**

**Step 5: Gather Stakeholder Feedback**

1. **Test the Prototype:**
   * **After setting up the links between screens, click the Play button (triangle icon in the top-right corner) to preview the app prototype.**
   * **You can interact with the app by clicking on the buttons and navigating through the screens as a real user would.**
2. **Share the Prototype:**
   * **Click on the Share button in the top-right corner to generate a shareable link.**
   * **You can share this link with stakeholders so they can interact with the prototype themselves.**
   * **Alternatively, you can record a walkthrough (using tools like Loom or Figma's presentation mode) to explain the app flow.**
3. **Collect Feedback:**
   * **Ask stakeholders to review the app and provide specific feedback about:** 
     + **The look and feel (e.g., colors, fonts, layout).**
     + **The app flow (e.g., navigation between screens, button placements).**
     + **Features they feel should be added or removed.**
   * **Incorporate their feedback by revising the design accordingly.**

**Step 6: Iterate Based on Feedback**

**Once you receive feedback:**

1. **Refine the Design:**
   * **Adjust the UI elements (buttons, images, text) based on the stakeholders' suggestions.**
   * **If certain pages seem too cluttered, consider simplifying them or changing their layout.**
2. **Update the Prototype:**
   * **After implementing feedback, update the prototype to reflect the changes.**
   * **Share the new version with stakeholders for final approval.**

**Step 7: Final Approval**

**Once the stakeholders are satisfied with the prototype:**

* **Export the Design:** 
  + **You can export assets like images or icons from Figma, or you can generate a link for them to download assets directly.**
* **Hand-off to Development:** 
  + **Figma has a developer handoff feature that allows developers to inspect the design, download assets, and get the necessary code snippets.**

**Conclusion**

**By following these steps, you've created an interactive E-Commerce mobile app prototype using Figma. This prototype can now be shared with stakeholders to gather feedback and approval. The use of Figma's interactive features and collaborative tools ensures that the design process is streamlined and transparent, helping you get clear and actionable feedback.**

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.

**Steps to do it:**

**Creating a Scrum project in Jira involves several steps to set up your project, add backlog items, prioritize them, and create a sprint. Below, I will walk you through a detailed process to create and manage a Scrum project in Jira from start to finish.**

**Step-by-Step Guide to Create a Scrum Project in Jira**

**Step 1: Create a New Scrum Project in Jira**

1. **Log in to Jira:**
   * **Open your Jira dashboard and log in using your credentials.**
   * **If you don’t have a Jira account, you can create one or use the free version provided by Atlassian.**
2. **Create a New Project:**
   * **From the Jira homepage, click on the "Create Project" button.**
   * **Select "Scrum Software Development" template (this is for Scrum methodology).**
   * **Click on Next.**
3. **Configure Project Settings:**
   * **Give your project a name (e.g., "E-Commerce App Development").**
   * **Choose the project key, which will be used as the prefix for issue IDs (e.g., "ECOM").**
   * **Select the project lead (this is typically you or the project manager).**
   * **Configure other settings like the permissions (you can use default settings for now).**
   * **Click Create.**

**Step 2: Set Up the Backlog**

**Once the Scrum project is created, you'll be directed to the Backlog view where you can add backlog items.**

1. **Navigate to the Backlog:**
   * **On the left sidebar, you’ll find an option called Backlog under your project name. Click on it to open the backlog view.**
2. **Add Backlog Items:**
   * **On the Backlog page, click the Create Issue button.**
   * **Create at least 5 backlog items (also known as User Stories or Tasks). These are the tasks that need to be completed during the project.**

**Example items:**

* + **Create user registration page (Story)**
  + **Develop API for login (Story)**
  + **Design homepage layout (Task)**
  + **Set up database schema (Task)**
  + **Implement product listing page (Story)**

1. **Add Details to the Backlog Items:**
   * **For each item, provide a summary, description, and assignee (if you know who will handle it). You can also assign an epic if necessary (e.g., "Frontend Development").**
   * **Set the priority for each item. Jira uses a priority scale of "Low," "Medium," and "High."**

**Step 3: Prioritize the Backlog**

1. **Drag and Drop Backlog Items:**
   * **In the backlog view, you will see all the items you’ve created.**
   * **Prioritize the items by dragging and dropping them in the order you want them to be worked on. The most critical items should be at the top, with lower-priority tasks at the bottom.**
2. **Set Story Points (Optional but recommended):**
   * **To estimate the work involved, you can set Story Points for each item. These represent the complexity or effort required to complete the task.**
   * **Story Points can be set by opening each backlog item, and on the right-hand side, you will see an option to add Story Points or an Estimate.**

**Step 4: Create a 1-Week Sprint**

1. **Create a Sprint:**
   * **Scroll up to the top of your Backlog page. You'll see an option called "Create Sprint". Click on it.**
   * **This will create an empty sprint ready to be filled with items from the backlog.**
2. **Set the Sprint Duration:**
   * **In the sprint view, you’ll find an option to set the start date and end date of the sprint.**
   * **Set the start date to today, and the end date to one week from today.**
   * **This creates a 1-week sprint.**

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

1. **Drag Items into the Sprint:** 
   * **Once the sprint is created, you’ll see it as a section in your backlog.**
   * **Select the top 5 backlog items that you want to complete within this sprint and drag them into the sprint section.**
2. **Verify Items in Sprint:** 
   * **Check that the items moved to the sprint are correctly listed and the estimated effort (Story Points) is visible.**
   * **If necessary, adjust priorities within the sprint.**

**Step 6: Start the Sprint**

1. **Start the Sprint:** 
   * **Once the backlog items are in the sprint, you can start the sprint by clicking on the "Start Sprint" button at the top of the sprint section.**
   * **A dialog box will appear asking for the sprint name, dates, and the goal for the sprint. You can provide a Sprint Goal (e.g., "Complete initial e-commerce pages and API").**
   * **Click Start to begin the sprint.**

**Step 7: Sprint Board at the Start and End**

**Now, we will discuss the sprint board and how it will look at the start and end of the sprint.**

1. **Sprint Board at the Start:**
   * **After starting the sprint, go to the Active Sprint view. This is where you can see the Scrum board with columns for To Do, In Progress, and Done.**
   * **Initially, all tasks will be under the "To Do" column because no work has been completed yet.**

**Example of the start of the sprint board:**

**+-----------------------------------------+**

**| To Do |**

**+-----------------------------------------+**

**| Create user registration page |**

**| Develop API for login |**

**| Design homepage layout |**

**| Set up database schema |**

**| Implement product listing page |**

**+-----------------------------------------+**

1. **Sprint Board at the End:**
   * **As work progresses, you or your team members can move the tasks from To Do to In Progress and finally to Done.**
   * **At the end of the sprint, most of the tasks will have been completed and moved to the Done column.**

**Example of the end of the sprint board:**

**+-----------------------------------------+**

**| To Do |**

**+-----------------------------------------+**

**| [Empty - all tasks completed] |**

**+-----------------------------------------+**

**+-----------------------------------------+**

**| In Progress |**

**+-----------------------------------------+**

**| [Empty - tasks moved to Done] |**

**+-----------------------------------------+**

**+-----------------------------------------+**

**| Done |**

**+-----------------------------------------+**

**| Create user registration page |**

**| Develop API for login |**

**| Design homepage layout |**

**| Set up database schema |**

**| Implement product listing page |**

**+-----------------------------------------+**

**Step 8: Take Screenshots of the Sprint Board**

1. **Screenshot at the Start:**
   * **Take a screenshot of the Active Sprint board with all tasks in the To Do column.**
2. **Screenshot at the End:**
   * **After the sprint ends, take another screenshot of the Active Sprint board with the tasks moved to the Done column.**

**Conclusion:**

**By following these steps, you have successfully:**

* **Created a Scrum project in Jira.**
* **Added backlog items (User Stories, Tasks).**
* **Prioritized the backlog and created a 1-week sprint.**
* **Moved backlog items into the sprint and started the sprint.**
* **Collected screenshot evidence of the sprint board at the start and end of the sprint.**

**This Scrum workflow in Jira helps you track progress and collaborate with your team to ensure successful sprint 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.

**Steps to do it:**

**The MOSCOW Method is a prioritization technique used to help categorize and prioritize requirements based on their importance and feasibility. The categories in the MOSCOW method are as follows:**

* **Must-Have: Essential features that are non-negotiable. Without them, the system would not be viable or functional.**
* **Should-Have: Important features that are desirable but not absolutely essential. They should be included if possible within the project’s constraints.**
* **Could-Have: Nice-to-have features that would add value but are not essential. They can be included if time and resources allow.**
* **Won’t-Have: Features that are explicitly excluded from the current project scope. They may be considered for future releases.**

**Let's categorize the provided features of the Library Management System using the MOSCOW method:**

**1. Add a feature to search books by title and author**

* **Impact on Users and Stakeholders: High. Users and librarians need an efficient search function to quickly find books, improving the user experience.**
* **Feasibility: High feasibility. This is a standard feature in most systems and can be implemented easily within time and budget constraints.**

**Category: Must-Have**

**Justification: This is a core functionality for a Library Management System. Without this, the system would fail to meet basic user expectations.**

**2. Implement an online book reservation system**

* **Impact on Users and Stakeholders: High. It allows users to reserve books remotely, which increases convenience and accessibility.**
* **Feasibility: Medium. Implementing an online reservation system requires integrating the database, adding user authentication, and managing availability in real-time.**

**Category: Must-Have**

**Justification: This is crucial for modern library management systems. Users expect to be able to reserve books online, especially in today’s digital environment.**

**3. Generate monthly reports on borrowed books for administrators**

* **Impact on Users and Stakeholders: High. Administrators need this feature to manage inventory, track borrowed books, and make data-driven decisions.**
* **Feasibility: High. Monthly reports are a common feature and can be automated with relatively low development cost.**

**Category: Should-Have**

**Justification: While important, this can be considered a supporting feature. It’s highly useful for administrators, but the system could operate without it in the short term.**

**4. Enable email notifications for overdue books**

* **Impact on Users and Stakeholders: High. This improves user experience by reminding them of overdue books and encourages timely returns.**
* **Feasibility: Medium. Implementing email notifications involves integration with email servers and user management systems.**

**Category: Must-Have**

**Justification: Timely notifications are essential for maintaining a smooth workflow in library operations and ensuring users return books on time.**

**5. Add support for QR code scanning for borrowing and returning books**

* **Impact on Users and Stakeholders: Medium. It enhances the user experience, especially for mobile users. It could also reduce manual intervention.**
* **Feasibility: Medium. Implementing QR code scanning involves hardware (scanners) and software integration, but it is achievable within budget and resources.**

**Category: Could-Have**

**Justification: While it’s an innovative and convenient feature, the system could operate without it, and it may be better suited for a later phase depending on available resources.**

**6. Create a user-friendly dashboard for librarians**

* **Impact on Users and Stakeholders: High. Librarians need an efficient interface to manage the library system, track books, users, and reservations.**
* **Feasibility: High. Dashboards are a standard feature in modern systems and can be implemented with common UI frameworks.**

**Category: Must-Have**

**Justification: A user-friendly dashboard is essential for the librarian’s workflow and critical to system efficiency.**

**7. Allow users to review and rate books**

* **Impact on Users and Stakeholders: Medium. It enhances user interaction with the system, allowing them to share feedback and choose books based on ratings.**
* **Feasibility: Medium. Implementing reviews and ratings is technically feasible but requires managing user input and ensuring the integrity of the reviews.**

**Category: Could-Have**

**Justification: While adding value, it’s not critical for the primary operation of the system, so it can be considered for future phases.**

**8. Integrate a chatbot for user assistance**

* **Impact on Users and Stakeholders: Medium. It can improve the user experience by providing quick answers to common questions about book availability, library policies, etc.**
* **Feasibility: Low. Building and maintaining a functional chatbot is resource-intensive and requires AI integration.**

**Category: Won’t-Have**

**Justification: While beneficial, it’s not essential for the core functionality of the library management system and may exceed resource limitations.**

**9. Develop a mobile app version of the system**

* **Impact on Users and Stakeholders: High. A mobile app version would improve accessibility and allow users to manage their library activities on the go.**
* **Feasibility: Medium. Developing a mobile app adds significant complexity and requires separate development, but it’s achievable within a reasonable time frame with sufficient resources.**

**Category: Should-Have**

**Justification: It adds great value by increasing user accessibility, but the core functionality can still be handled through a website. Thus, it's not essential in the initial version.**

**10. Provide multi-language support**

* **Impact on Users and Stakeholders: Medium. Multi-language support increases accessibility for users who speak different languages.**
* **Feasibility: Medium to Low. This requires extensive localization and translation, which may increase the project scope and cost.**

**Category: Could-Have**

**Justification: While this can significantly broaden the user base, it is not critical to the system's initial release and can be considered a nice-to-have feature for later iterations.**

**Google Sheet or Excel File Submission**

**You will organize the requirements and their MOSCOW categorization in a table format for easy review. Here’s a sample structure for the Google Sheet or Excel file:**

| **Requirement** | **Impact on Users and Stakeholders** | **Feasibility** | **MOSCOW Category** | **Justification** |
| --- | --- | --- | --- | --- |
| **Add a feature to search books by title and author** | **High** | **High** | **Must-Have** | **Essential for core system functionality.** |
| **Implement an online book reservation system** | **High** | **Medium** | **Must-Have** | **Critical for user convenience and modern library experience.** |
| **Generate monthly reports on borrowed books for admins** | **High** | **High** | **Should-Have** | **Important for admins but not critical for basic operation.** |
| **Enable email notifications for overdue books** | **High** | **Medium** | **Must-Have** | **Ensures timely returns, essential for library management.** |
| **Add support for QR code scanning** | **Medium** | **Medium** | **Could-Have** | **Enhances experience but not essential for the core system.** |
| **Create a user-friendly dashboard for librarians** | **High** | **High** | **Must-Have** | **Core feature for managing the library system efficiently.** |
| **Allow users to review and rate books** | **Medium** | **Medium** | **Could-Have** | **Enhances interaction but not essential for system operation.** |
| **Integrate a chatbot for user assistance** | **Medium** | **Low** | **Won’t-Have** | **Not essential and exceeds the current project scope.** |
| **Develop a mobile app version of the system** | **High** | **Medium** | **Should-Have** | **Increases user accessibility but not essential at launch.** |
| **Provide multi-language support** | **Medium** | **Medium** | **Could-Have** | **Broadens user base, but not necessary for the initial release.** |

**Conclusion:**

**After categorizing the features based on the MOSCOW Method, we have provided a clear view of which features are critical for the Library Management System's core functionality and which can be added later or excluded from the initial release. The Must-Have features form the system’s foundation, while the Should-Have and Could-Have features provide additional value, and the Won’t-Have features are outside the current scope.**

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.

**Steps to do it:**

**Linking Jira tasks with Confluence helps streamline task tracking and progress monitoring for projects. In this case, you will link Jira tasks related to the Library Management System development to a Confluence page, providing a clear visual overview of the project’s progress. Below, I’ll guide you step-by-step on how to do this.**

**Step 1: Create a New Page in Confluence**

1. **Log in to Confluence:**
   * **Open Confluence in your browser and log in to your Confluence instance.**
2. **Create a New Page:**
   * **On the left-hand navigation bar, click on the "Create" button to create a new page.**
   * **Choose the appropriate template (you can select a Blank Page template to start with).**
3. **Title the Page:**
   * **In the page editor, give the page a title, for example, "Library Management System Project Overview".**
4. **Add Introduction (Optional):**
   * **You can add an introductory section at the top of the page to explain the purpose of this page, such as:** 
     + **"This page tracks the progress of the Library Management System development project, including task status and completion metrics."**

**Step 2: Embed Jira Issues into the Confluence Page**

1. **Use the Jira Issue Macro:**
   * **In the Confluence page editor, click on the "+" (Insert More Content) icon in the toolbar at the top of the page.**
   * **Select Jira from the dropdown menu. This will open the Jira Issue/Filter macro dialog.**
2. **Search and Select Jira Issues:**
   * **In the macro dialog, you can either search for specific Jira issues or use a Jira filter.**
   * **For this task, select "Insert Jira Issue".**
   * **In the search bar, enter the task names or issue keys related to the development of the Library Management System (e.g., "Develop book search functionality", "Create user login page", etc.).**

**Sample Jira Issues:**

* + **"Develop book search functionality"**
  + **"Create user login page"**
  + **"Design homepage layout"**
  + **"Implement product listing page"**
  + **"Set up database schema"**

1. **Insert Issues:**
   * **Select the relevant issues for your Library Management System project.**
   * **Once the issues are selected, click Insert.**

**This will embed the selected Jira issues into the Confluence page. The issues will display with key details such as the issue title, assignee, and status (e.g., "To Do", "In Progress", "Done").**

**Step 3: Display Issues with Status**

1. **Customize the Display (Optional):** 
   * **After inserting the Jira issues, you can customize how they appear on the page.**
   * **In the Jira Issue Macro dialog, you can choose which columns to display (such as Status, Assignee, Priority, etc.). You can ensure that the Status column is included to track the task status (e.g., To Do, In Progress, Done).**
   * **Choose to display the list of issues or a detailed table view for better tracking.**

**Step 4: Add a Progress Bar in Confluence**

1. **Add a Progress Bar Macro:**
   * **In Confluence, click on the "+" (Insert More Content) icon again and select Other Macros.**
   * **Search for Progress Bar and select the Progress Bar macro.**
2. **Configure the Progress Bar:**
   * **A dialog box will open asking for the Percentage of progress.**
   * **To track the progress of your Jira tasks, calculate the percentage of completed tasks in the sprint or project. For example:** 
     + **If 3 out of 5 tasks are marked as "Done", the progress is 60%.**
   * **Enter the calculated percentage (e.g., 60) in the progress bar settings.**

**You can customize the color and appearance of the progress bar as well, depending on your preference.**

1. **Insert the Progress Bar:**
   * **After configuring the progress bar, click Insert. The progress bar will now appear on the page and visually track the completion of tasks.**

**Step 5: Finalize and Save the Page**

1. **Review the Page:**
   * **Check that the Jira issues are properly embedded and displayed with their status (e.g., To Do, In Progress, Done).**
   * **Ensure the progress bar is showing the correct percentage based on the completion of tasks.**
2. **Publish the Page:**
   * **Once you are satisfied with the page content and the layout, click the "Publish" button to save and publish the page.**

**Step 6: Take a Screenshot of the Confluence Page**

1. **View the Published Page:**
   * **Go to the Confluence page where you’ve embedded the Jira issues and the progress bar.**
   * **Ensure everything is displayed correctly.**
2. **Take a Screenshot:**
   * **Use your computer’s screenshot tool (e.g., Snipping Tool on Windows or Command + Shift + 4 on Mac) to capture an image of the page showing the embedded Jira tasks and the progress bar.**

**Example Layout of the Confluence Page**

**Here’s a possible layout for your Confluence Page:**

**Library Management System Project Overview**

**This page tracks the progress of the Library Management System development project, including task status and completion metrics.**

**Jira Tasks Overview:**

| **Task Name** | **Assignee** | **Status** |
| --- | --- | --- |
| **Develop book search functionality** | **John Doe** | **In Progress** |
| **Create user login page** | **Jane Smith** | **To Do** |
| **Design homepage layout** | **Mary Lee** | **Done** |
| **Implement product listing page** | **Peter Pan** | **In Progress** |
| **Set up database schema** | **Steve Doe** | **Done** |

**Project Progress:**

**Progress Bar:**

* **[Progress Bar (60%)] — indicating that 3 out of 5 tasks are completed.**

**Conclusion:**

**By following these steps, you have created a Confluence page titled "Library Management System Project Overview" and embedded Jira issues related to the project. You’ve also included a progress bar to visually track the completion of each task. This integration between Jira and Confluence helps streamline the process of task tracking and progress monitoring, making it easier for stakeholders to stay updated on the project’s status. Finally, you can submit a screenshot of the Confluence page showing the embedded Jira tasks and the progress bar.**

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.

**Steps to do it:**

**Prioritizing the features of a Task Management System using both the MoSCoW method and Kano model helps ensure that the most critical features are prioritized while considering user satisfaction and system functionality. We will go step-by-step to explain how to apply these models in Jira.**

**Step 1: Understanding the MoSCoW Method**

**The MoSCoW Method is a popular prioritization technique used in Agile project management. The acronym stands for:**

* **Must-Have: Features that are essential for the system’s functionality and user needs.**
* **Should-Have: Features that are important but not crucial for the immediate release.**
* **Could-Have: Features that are nice to have and can be included if time and resources allow.**
* **Won’t-Have: Features that are out of scope for this iteration or release.**

**Step 2: Understanding the Kano Model**

**The Kano model is a framework that helps prioritize features based on user satisfaction:**

* **Basic Needs: Features that users expect and are dissatisfied if absent, but their presence doesn’t delight them.**
* **Performance Needs: Features that increase user satisfaction as they are improved.**
* **Excitement Needs: Features that users do not expect but can greatly increase satisfaction when included.**
* **Indifferent Needs: Features that neither increase nor decrease user satisfaction.**

**Step 3: Prioritizing Features Using MoSCoW in Jira**

**We will now apply MoSCoW to the features of the Task Management System.**

**1. User Login and Role Assignment**

* **Impact: A basic feature necessary to ensure proper access control and security.**
* **Feasibility: Easy to implement, requiring standard authentication and role assignment logic.**
* **Priority: Must-Have** 
  + **Justification: Without user login and role assignment, the system cannot function securely or efficiently.**

**2. Task Creation and Assignment**

* **Impact: The core functionality of any task management system.**
* **Feasibility: Standard feature, relatively easy to implement.**
* **Priority: Must-Have** 
  + **Justification: Essential for the system to manage tasks and allocate them to appropriate team members.**

**3. Task Prioritization and Deadlines**

* **Impact: Critical for managing workloads and ensuring that tasks are completed in a timely manner.**
* **Feasibility: Slightly more complex than basic task creation, as it may require advanced features for prioritization (e.g., drag-and-drop, color coding).**
* **Priority: Should-Have** 
  + **Justification: While important for managing workflows efficiently, the system can still function without complex prioritization and deadlines. A simple priority flag (e.g., High, Medium, Low) would suffice initially.**

**4. Progress Tracking and Reporting**

* **Impact: Important for monitoring progress and providing insights to the team and stakeholders.**
* **Feasibility: Medium complexity, as it requires building dashboards and generating reports.**
* **Priority: Should-Have** 
  + **Justification: While valuable, progress tracking and reporting can be added in a future release if the core features (task creation and login) are working.**

**Summary of MoSCoW Prioritization in Jira:**

| **Feature** | **MoSCoW Priority** |
| --- | --- |
| **User Login and Role Assignment** | **Must-Have** |
| **Task Creation and Assignment** | **Must-Have** |
| **Task Prioritization and Deadlines** | **Should-Have** |
| **Progress Tracking and Reporting** | **Should-Have** |

**Step 4: Prioritizing Features Using the Kano Model in Jira**

**Now let’s apply the Kano model to the same features to evaluate how each feature impacts user satisfaction.**

**1. User Login and Role Assignment**

* **Kano Category: Basic Needs** 
  + **Justification: Users expect to have proper authentication and role assignment. If absent, it will lead to dissatisfaction, but its presence does not delight users. It’s a fundamental requirement for system security and usability.**

**2. Task Creation and Assignment**

* **Kano Category: Basic Needs** 
  + **Justification: This is the core feature of a task management system. Users expect this functionality to be present. Its absence would make the system unusable, but its presence is expected and does not create additional excitement.**

**3. Task Prioritization and Deadlines**

* **Kano Category: Performance Needs** 
  + **Justification: The ability to prioritize tasks and set deadlines directly influences user satisfaction. The better the prioritization and deadline management, the more satisfied the users will be. This is an important feature that enhances the usability of the system but is not strictly essential for the system to function.**

**4. Progress Tracking and Reporting**

* **Kano Category: Excitement Needs** 
  + **Justification: While progress tracking and reporting are nice to have, they are not necessarily expected in the first iteration of the system. However, their presence can delight users and improve the overall satisfaction with the system by providing clear insights into performance.**

**Summary of Kano Model Prioritization in Jira:**

| **Feature** | **Kano Model Category** |
| --- | --- |
| **User Login and Role Assignment** | **Basic Needs** |
| **Task Creation and Assignment** | **Basic Needs** |
| **Task Prioritization and Deadlines** | **Performance Needs** |
| **Progress Tracking and Reporting** | **Excitement Needs** |

**Step 5: Implementing Prioritization in Jira**

**To implement MoSCoW and Kano prioritization in Jira, follow these steps:**

**1. Create the Issues in Jira**

* **Go to your Jira project and create issues for each feature:** 
  + **For example: "User Login and Role Assignment", "Task Creation and Assignment", etc.**

**2. Add MoSCoW Priorities**

* **Set Custom Fields for MoSCoW:** 
  + **You can either use a Custom Field for MoSCoW priority or use the Priority field in Jira and customize the options.**
  + **Go to Jira Settings > Issues > Custom Fields to create a new custom field called "MoSCoW Priority" with options like "Must-Have", "Should-Have", "Could-Have", and "Won't-Have."**
  + **When creating or editing each issue, set the priority based on the MoSCoW method.**

**3. Add Kano Model Categories**

* **Create a Custom Field for Kano Categories:** 
  + **Similarly, create a Custom Field for the Kano model (e.g., Basic Needs, Performance Needs, Excitement Needs).**
  + **Add this field to each issue and categorize the features based on the Kano model.**

**4. Sprint Planning**

* **When planning your sprint in Jira:** 
  + **Focus on Must-Have MoSCoW features for the current sprint.**
  + **Prioritize Basic Needs features first in your backlog since they are necessary to make the system functional.**
  + **For Performance Needs and Excitement Needs, prioritize them later based on available time and resources.**

**5. Reporting and Monitoring**

* **Use Jira’s Dashboard to create a report of the issues, displaying their MoSCoW priorities and Kano categories.** 
  + **You can create JQL filters to show issues based on the MoSCoW priorities or Kano categories, making it easier to track and adjust priorities as the project progresses.**

**Step 6: Monitor Progress and Adjust**

**As you progress with the development, you can review and adjust priorities based on feedback and evolving project needs. Jira’s Agile board, combined with MoSCoW and Kano prioritization, will allow you to track the development of features and ensure that the team focuses on the most impactful tasks.**

**Conclusion:**

**By applying MoSCoW and Kano models to the Task Management System features in Jira, you can ensure that essential features are prioritized (Must-Have), user satisfaction is maximized (Performance and Excitement Needs), and unnecessary scope creep is avoided (Won’t-Have). Tracking these priorities in Jira allows your team to align their efforts with both the functional requirements and user expectations.**

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**:

**Lab Manual: 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.

**Steps to do it:**

**In this task, we are tasked with developing an Online Learning Platform and categorizing and prioritizing its functionalities using both the MoSCoW and Kano prioritization techniques in Jira. Below, I’ll guide you step-by-step on how to categorize and prioritize the features of the platform using these techniques.**

**Step 1: Understand the Functionalities**

**The platform requires the following features:**

1. **Course Enrollment and Registration - Users must be able to register and enroll in courses.**
2. **Video Lecture Streaming - The platform needs the ability to stream video lectures.**
3. **Interactive Quizzes and Assignments - Quizzes and assignments need to be created for learner interaction.**
4. **Progress Tracking Dashboard - A dashboard to track user progress in their courses.**
5. **Peer-to-Peer Discussion Forums - A place for students to discuss topics with one another.**
6. **Certificate Generation - A feature to generate certificates for completed courses.**

**Step 2: MoSCoW Prioritization**

**In the MoSCoW prioritization method, features are categorized into four categories:**

* **Must-Have: Essential features for the system to function.**
* **Should-Have: Important features but not critical for launch.**
* **Could-Have: Nice-to-have features, but not essential.**
* **Won’t-Have: Features that are not necessary for the current release.**

**Categorizing Features Using MoSCoW**

1. **Course Enrollment and Registration**
   * **MoSCoW Category: Must-Have**
   * **Justification: Without the ability for users to enroll and register, the platform would not be usable. This is a basic feature for any learning platform.**
2. **Video Lecture Streaming**
   * **MoSCoW Category: Must-Have**
   * **Justification: Video streaming is the core method of delivering content in an online learning platform. Without this feature, the learning experience would be incomplete.**
3. **Interactive Quizzes and Assignments**
   * **MoSCoW Category: Should-Have**
   * **Justification: Quizzes and assignments enhance the learning experience and reinforce content, but a basic learning platform could technically function without them. Still, they should be prioritized for the next release.**
4. **Progress Tracking Dashboard**
   * **MoSCoW Category: Should-Have**
   * **Justification: A dashboard to track progress helps users stay motivated, but it's not as critical as the ability to enroll in courses and watch lectures. It should be included in the next release after core features are developed.**
5. **Peer-to-Peer Discussion Forums**
   * **MoSCoW Category: Could-Have**
   * **Justification: While helpful for community engagement and peer learning, discussion forums are not essential for the platform's core functionality. They can be added in later versions.**
6. **Certificate Generation**
   * **MoSCoW Category: Should-Have**
   * **Justification: Certificate generation is an important feature, especially for user motivation, but the system could still function without it, so it should be prioritized after the must-have features.**

**Summary of MoSCoW Prioritization:**

| **Feature** | **MoSCoW Priority** |
| --- | --- |
| **Course Enrollment and Registration** | **Must-Have** |
| **Video Lecture Streaming** | **Must-Have** |
| **Interactive Quizzes and Assignments** | **Should-Have** |
| **Progress Tracking Dashboard** | **Should-Have** |
| **Peer-to-Peer Discussion Forums** | **Could-Have** |
| **Certificate Generation** | **Should-Have** |

**Step 3: Kano Model Prioritization**

**The Kano Model helps prioritize features based on user satisfaction. Features are categorized into five categories:**

* **Basic Needs: Expected by users; their absence causes dissatisfaction, but their presence does not delight users.**
* **Performance Needs: Features that increase user satisfaction as they are improved.**
* **Excitement Needs: Features that users do not expect but that can create delight when present.**
* **Indifferent Needs: Features that neither increase nor decrease user satisfaction.**
* **Reverse Needs: Features that cause dissatisfaction when present and delight when absent (rare).**

**Categorizing Features Using the Kano Model**

1. **Course Enrollment and Registration**
   * **Kano Category: Basic Needs**
   * **Justification: Users expect to be able to enroll and register in courses. The absence of this feature would make the platform unusable, but its presence is not a source of delight for users.**
2. **Video Lecture Streaming**
   * **Kano Category: Basic Needs**
   * **Justification: Video streaming is a core feature, and users expect it. Its absence would cause dissatisfaction, but its presence doesn’t necessarily excite users beyond the basic expectation.**
3. **Interactive Quizzes and Assignments**
   * **Kano Category: Performance Needs**
   * **Justification: Quizzes and assignments increase user engagement and satisfaction as they help learners assess their understanding and progress. The better these features are implemented, the more satisfied users will be.**
4. **Progress Tracking Dashboard**
   * **Kano Category: Performance Needs**
   * **Justification: The ability to track progress directly influences user satisfaction. Users expect this functionality, and the more detailed and user-friendly it is, the happier users will be.**
5. **Peer-to-Peer Discussion Forums**
   * **Kano Category: Excitement Needs**
   * **Justification: Discussion forums are not a basic requirement but can significantly enhance user satisfaction and engagement if implemented well. This feature is often not expected, but when present, it can delight users and increase community engagement.**
6. **Certificate Generation**
   * **Kano Category: Excitement Needs**
   * **Justification: While users may appreciate receiving a certificate after completing a course, they don’t necessarily expect it. If the certificate is of high quality or is customizable, it can delight users.**

**Summary of Kano Model Prioritization:**

| **Feature** | **Kano Category** |
| --- | --- |
| **Course Enrollment and Registration** | **Basic Needs** |
| **Video Lecture Streaming** | **Basic Needs** |
| **Interactive Quizzes and Assignments** | **Performance Needs** |
| **Progress Tracking Dashboard** | **Performance Needs** |
| **Peer-to-Peer Discussion Forums** | **Excitement Needs** |
| **Certificate Generation** | **Excitement Needs** |

**Step 4: Implementing in Jira**

**To categorize and prioritize the features in Jira, follow these steps:**

**1. Create Issues in Jira**

* **For each functionality, create an issue in Jira. Each issue will represent a feature in the platform (e.g., Course Enrollment and Registration, Video Lecture Streaming, etc.).**

**2. Add MoSCoW Priority Custom Field**

* **Go to Jira Settings > Issues > Custom Fields and create a MoSCoW Priority field with the following options: Must-Have, Should-Have, Could-Have, Won't-Have.**
* **When creating or editing each issue in Jira, assign a priority based on the MoSCoW method.**

**3. Add Kano Category Custom Field**

* **Similarly, create a Custom Field for the Kano Category (e.g., Basic Needs, Performance Needs, Excitement Needs).**
* **Assign a category for each feature based on the Kano model.**

**4. Sprint Planning**

* **When you plan sprints in Jira, focus on completing Must-Have features first (such as Course Enrollment and Video Lecture Streaming).**
* **After completing Must-Have features, tackle Should-Have and Performance Needs (like quizzes and progress tracking).**
* **Add Excitement Needs and Could-Have features (e.g., Discussion Forums and Certificate Generation) in later releases or as part of future sprints.**

**5. Create Jira Filters and Reports**

* **You can create Jira filters based on MoSCoW priorities or Kano categories. For example, you could create a filter to show all Must-Have features or all Basic Needs features.**
* **Create Jira dashboards to track progress across these priorities, allowing you to see how many Must-Have and Performance Needs features have been completed at any time.**

**Step 5: Monitor and Adjust**

**As development progresses, monitor the progress of each feature. If needed, adjust priorities based on user feedback, market conditions, or resource constraints. Using the Jira Board and Sprints, you can easily track and manage the completion of prioritized features.**

**Conclusion:**

**By applying both the MoSCoW and Kano models to the features of your Online Learning Platform in Jira, you can ensure that the platform is developed with a clear focus on what is most important to your users. You prioritize essential features for launch (Must-Have), ensure satisfaction with performance features, and delight users with exciting, unexpected features. This structured approach helps guide your development process efficiently and effectively.**