Northwind Technologies, a growing software development company, faces several challenges. As the company expands, its sales team struggles with managing increasing customer interactions and sales opportunities using outdated methods like spreadsheets and manual record-keeping.

This lack of an integrated system results in several critical issues:

* Sales representatives find maintaining accurate and up-to-date customer information difficult without a centralized database. It often leads to duplicate records and lost opportunities.
* The absence of a streamlined process for tracking sales opportunities causes delays in the sales cycle, making it hard for the team to prioritize leads and close deals efficiently.
* Managers lack access to real-time sales data, making it challenging to assess team performance, identify trends, and make informed decisions.
* Manual tracking of follow-up tasks leads to inconsistent customer engagement, which affects customer satisfaction and retention rates.
* The sales team uses multiple disconnected tools for communication and scheduling, leading to inefficiencies and missed opportunities for collaboration.

To address these challenges and support its growth objectives, Northwind Technologies has decided to implement a CRM system tailored to its specific needs. The goal is to provide the sales team with the ability to manage customer relationships, streamline sales processes, and monitor the sales process through advanced reporting and analytics.

As the need for a CRM system becomes apparent, Emily Johnson, Director of Sales Operations, and Michael Lee, Head of IT, assemble a project team with relevant experience and expertise in CRM development and Agile project management.

Scrum Team

* Product Owner: Sarah Thompson (Roadmap, Epics, Product Vision, Business Value)
* Scrum Master: David Kim (Scrum Ceremonies, provides support to the Scrum Team)
* Business Analyst: Tom Brown (User Stories and UAT)
* UX/UI Designer: Rachel Green (User Interface and interaction design)
* Frontend Developer: Lisa Chen (User Interface development and handling client-side logic such as user interaction, data validation, and rendering data)
* Backend Developer: Mark Wilson (Server-side development, database management, security and authentication)
* QA Engineer: Alex Martinez (Testing)

The team conducts interviews and workshops with sales representatives, managers, and other stakeholders to gather insights into their challenges and requirements.

Sarah, the Product Owner, identifies the following product features:

1. **Contact Management**

This feature allows sales representatives to easily manage customer contacts.

**Requirements:**

|  |  |
| --- | --- |
| REQ\_01 | Enable sales representatives to list, view, add, edit, and delete customer contacts. Contact details must include customer name, company name, email, phone, **and** additional details. |
| REQ\_02 | Allow users to import and export contact lists. |

**2. Sales Opportunity Tracking**

This feature allows users to track sales opportunities and update their status.

**Requirements:**

|  |  |
| --- | --- |
| REQ\_03 | Implement functionality for users to create and update sales opportunities. |
| REQ\_04 | Implement dashboard analytics to track opportunities through different stages of the sales cycle. |

**3. Reporting and Analytics**

This feature allows sales managers to gain insights into team performance and customer interactions.

**Requirements:**

|  |  |
| --- | --- |
| REQ\_05 | Generate reports on sales performance. |
| REQ\_06 | Provide dashboard analytics to visualize sales data. |

**4. Integration and Automation**

This feature enables automated reminders to sales staff about appointments and tasks.

**Requirements:**

|  |  |
| --- | --- |
| REQ\_07 | Integrate the CRM with email and calendar systems. |
| REQ\_08 | Automate reminders and follow-up tasks for sales representatives. |

**5.  System Performance Requirements**

These requirements describe general system performance.

|  |  |
| --- | --- |
| REQ\_09 | The system should be able to handle at least 500 concurrent users without any degradation in performance. |
| REQ\_10 | Page load times for any CRM feature must not exceed 3 seconds under normal network conditions. |

**Project Roadmap**

As a result of a collaborative process involving input from the stakeholders, the Scrum team, and end-users, the Product Owner develops the project roadmap.

The roadmap is divided into phases to ensure a manageable and iterative development process. Each phase focuses on delivering specific functionality that provides immediate value to the sales team.

The Scrum team agrees to work in 2-week sprints.

|  |  |  |
| --- | --- | --- |
| **ID** | **Phase/Milestone** | **Duration** |
| 1 | **Contact Management**: Sales representatives can manage customer contacts easily. | 2 sprints |
| 2 | **Sales Opportunity Tracking:** Users can track sales opportunities and update their status. | 1 sprint |
| 3 | **Reporting and Analytics:** Sales managers can gain insights into team performance and customer interactions. | 3 sprints |
| 4 | **Integration and Automation:** CRM integrates with Outlook email & calendar; sales staff receive automated reminders about meetings and tasks | 2 sprints |
|  | **Total project duration** | **8 sprints** |

**Your tasks:**

1. Write high-level User Stories (Epics) that describe each product feature.
2. Split the Contact Management Epics into low-level User Stories.
3. Based on the user stories and the roadmap, create a Gantt chart for this project.
4. Create a traceability matrix mapping requirements to User Stories.

Are you ready for this challenge? Let's begin.

**Step 1: Write high-level User Stories (Epics)**

1. Use the downloaded **Project Workbook** Microsoft Excel file for your work in this project.

2. Write a high-level User Story for each product feature. Use the Project Workbook (*EPICS* tab) to document the Epics. Ensure that each Epic clearly describes the user, their requirement, and the reason behind the requirement.

* For the user stories, use the "As a [user], I want to [goal], so that [reason]" format.
* Consider different user perspectives (e.g. sales representative, sales manager, customer support).
* Add one or two acceptance criteria to each Epic to better describe the product feature. The acceptance criteria at this level don't have to follow any specific format or cover every scenario. Epics will be refined and split into user stories in the next step, so don't worry about covering all the angles.

**Example: Reporting & Analytics**

As a Sales Manager, I want to view dashboard analytics and sales reports to gain insights into team performance and customer interactions.

Acceptance criteria:

* Sales Managers can view dashboard analytics that show team performance and customer interactions
* Sales Managers can generate and download sales reports.

**Step 2: Break EPICS into User Stories**

This step aims to help you understand how to effectively deconstruct complex project requirements into smaller, more actionable tasks that can be prioritized, estimated, and completed within a single iteration or sprint. This exercise is fundamental to Agile development practices and enhances your ability to manage and deliver complex software projects.

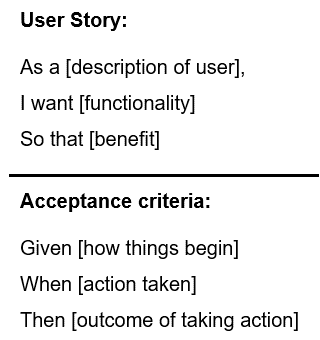
In software development, particularly within Agile methodologies, breaking down large pieces of work into manageable units is crucial for efficient project management. This activity involves taking a broad, high-level requirement known as an "epic" and decomposing it into smaller, more detailed "User Stories."

In Step 1, you should have written one Epic for each product feature. Now, you need to break them into lower-level components—User Stories.

1. Select an Epic that describes the Contact Management feature.
2. Break this Epic into User Stories. Use the *User Stories* tab in the **Project Workbook** to document them.

**Tips:**

* Each User Story should describe a single functionality from the perspective of a specific type of user, e.g. a sales representative.
* Make sure each User Story is Independent, Negotiable, Valuable, Estimable, Small, and Testable (INVEST Principle).
* Think of some hidden scenarios, e.g. system error or when the user does something wrong. Some of the scenarios may not be apparent from the requirements; however, as a Business Analyst, you should be able to imagine them.
* Add Acceptance Criteria to describe the expected system behaviour to the Development Team.
* Use the following format for each User Story:



**Step 3. Create a Gantt chart for the project**

You now need to plan project delivery in greater detail by creating a Gantt chart. A roadmap provides a high-level strategic overview of the product's goals and features planned over time, focusing on the direction and priorities of development. In contrast, a Gantt chart is a more detailed project management tool that outlines the timeline, dependencies, and scheduling of specific tasks. Since Scrum places emphasis on flexible, iterative progress rather than fixed timelines, let’s just plan the first phase of the project and see how it plays out.

**Steps to Create a Gantt chart:**

1. List the project phases and mark key milestones on the project timeline (e.g. contact management module completion and analytics dashboard launch).
2. Break the Contact Management Phase into smaller components, such as User Stories.
3. Break **one** User Story (List Contacts) from the Contact Management Epic into tasks and allocate the tasks to your team members, e.g.

* TASK-01"Contact list" page design – Rachel (UX designer)

1. Estimate the task duration and note dependencies between tasks. Your estimation does not need to be realistic. It is a fictitious exercise aimed at demonstrating various steps in Project Delivery.
2. Plot the task duration on a timeline.
3. Now estimate the duration of the remaining user stories and plot them on the timeline.
4. Finally, plot the duration of the remaining phases.
5. Now compare the project timeline in the Gantt chart and the roadmap. Are they the same?

**Step 4: Develop a Traceability Matrix**

Create a traceability matrix to ensure that all requirements are covered by User Stories and acceptance criteria.

Acceptance criteria define the conditions under which a User Story is considered complete and acceptable to the stakeholders. They outline specific requirements and scenarios that must be met for the functionality to be deemed correct. Test cases are derived from the acceptance criteria to ensure the implemented feature behaves as expected. For this activity, assume that each acceptance criterion becomes a test case (direct mapping).

**Steps to Create a Traceability Matrix:**

1. Project Workbook (Traceability tab) lists the requirements provided in the scenario
2. Map each requirement to the corresponding User Story.
3. Give each acceptance criterion an individual reference number.
4. Ensure that each requirement is mapped to one or more test cases (derived from the acceptance criteria), as in the example below: