**Common Types of Specification Documents for Projects**

**1. Business Requirements Specification (BRS)**

A BRS outlines the business needs for a project. It focuses on what the business wants to achieve with the system.

**Common Sections:**

**Introduction:** Overview of the project and its purpose.

**Objectives:** High-level goals the system must achieve.

**Scope:** Defines what is in scope and out of scope for the project.

**Business Process:** Description of the current process and the desired process.

**Stakeholders:** Key stakeholders involved in the project.

**Requirements:** High-level functional and non-functional requirements.

**Assumptions:** Conditions assumed to be true for the project.

**Risks:** Identified risks that could impact the project.

**Dependencies:** Dependencies on other systems or processes.

**Appendices:** Any additional documents or references.

**2. Functional Requirements Specification (FRS)**

An FRS describes how the system will operate and the functions it must perform to meet the business needs.

**Common Sections:**

**Introduction:** Overview of the system and its purpose.

**Functional Requirements:** Detailed list of functions the system must perform.

**Data Requirements:** Description of the data that the system will process and store.

**User Interface (UI) Requirements:** Details about the design and layout of the user interface.

**Workflows:** Flowcharts or diagrams showing how processes should work.

**Integration Requirements:** Requirements for how the system will interact with other systems.

**Reporting Requirements:** Descriptions of any reports the system needs to generate.

**Error Handling:** How the system should respond to errors or failures.

**Security Requirements**: Requirements for access control, authentication, and data protection.

**3. Technical Specification (TS)**

A Technical Specification outlines the technical architecture, technologies, and configurations needed for a project.

Common Sections:

**Introduction:** Project background and purpose.

**System Architecture:** High-level diagram of the system architecture (e.g., servers, databases, APIs).

**Technology Stack:** List of technologies, frameworks, and programming languages to be used.

**System Interfaces:** Details about APIs or integration points with other systems.

**Database Design:** Structure of the database, including tables and relationships.

**Data Flow Diagrams:** How data moves through the system.

**Security:** Detailed security measures, including encryption and access controls.

**Performance Requirements:** Technical performance targets (e.g., response times, load capacity).

**Backup and Recovery:** How data and the system will be backed up and restored.

**4. Product Requirements Document (PRD)**

A PRD focuses on defining the product's features and how they should behave from the user's perspective.

Common Sections:

**Introduction:** Overview of the product and its goals.

**Target Audience:** Description of who the product is designed for.

**Features:** Detailed list of features, including use cases and user stories.

**User Personas:** Fictional users representing different user types.

**User Experience (UX):** Description of the desired user experience, including wireframes or mockups.

**Functional Requirements:** Detailed functional requirements for each feature.

**Non-Functional Requirements:** Performance, scalability, and security requirements.

Acceptance Criteria: Conditions that must be met for the feature to be considered complete.

**5. Non-Functional Requirements Specification (NFRS)**

A NFRS describes the non-functional aspects of a system, such as performance, security, and usability.

**Common Sections:**

**Performance:** Requirements for system speed, throughput, and response times.

**Scalability:** The system’s ability to handle growth in users or data.

**Security:** Requirements for data protection, access control, and encryption.

**Reliability:** Uptime requirements and how failures should be handled.

**Usability:** Requirements related to ease of use, user experience, and accessibility.

**Maintainability:** How easily the system can be maintained or updated.

**Compliance:** Requirements to comply with legal or regulatory standards (e.g., GDPR).

**6. Test Plan/Specification (TPS)**

A Test Plan/Specification defines how the system will be tested to ensure it meets its requirements.

**Common Sections:**

**Test Objectives:** What the testing is intended to achieve.

**Test Scope:** The features and functions that will be tested.

**Test Cases:** Detailed test cases, including inputs, expected outcomes, and actual results.

**Test Environment:** Description of the environment where testing will take place (e.g., hardware, software, and network configurations).

**Testing Tools:** Tools and software used for testing.

**Test Schedule:** Timeline for conducting tests.

**Acceptance Criteria:** Conditions that must be met for the system to pass testing.

Defect Tracking: How defects and bugs will be tracked and resolved.

**7. User Requirement Specification (URS)**

A URS outlines the needs and expectations of end-users from the system.

**Common Sections:**

**Introduction:** Background and purpose of the system from a user perspective.

**User Needs:** A description of the user’s needs, including features, ease of use, and accessibility.

**Functional Requirements:** Specific functions that users need the system to perform.

**Performance Requirements:** How fast and reliable the system needs to be for users.

**Training Requirements:** Any training or help documentation that users will need.

**User Interface Requirements:** Details on the user interface design, including accessibility for all users.

Capabilities, business processes user personas, user stories, and requirements.

Understanding how these elements are interconnected is crucial.   **1. Capabilities**

**Definition** Capabilities are the main things the system needs to do. They describe the high-level functions of the system.

**Example**:  
For the **Appointment Booking System**, the key capability could be:

* **"Automated Scheduling of Appointments"**: The system should enable officers to define their availability and allow students to book appointments based on the officer's availability.
* **"Notifications and Reminders"**: The system should automatically send email reminders to both students and officers before the scheduled appointment.

**Explanation to Team**: Capabilities define the "what" of the system at a high level. These guide the development process by ensuring that all functionalities and features being developed align with what the system is fundamentally supposed to achieve.

**2. Business Processes**

**Definition**: Business processes refer to the structured activities or tasks that an organization follows to deliver its services. These processes are often broken down into three levels:

* **Level 1 (High-Level)**: The overall major functions of the organization.
* **Level 2 (Mid-Level)**: Key steps within each major function.
* **Level 3 (Low-Level)**: Detailed tasks and activities within each key step.

**Example**:  
For the **SASO Officer and Student Booking System**:

* **Level 1**: Appointment Management Process.
* **Level 2**:
  1. Officer sets availability.
  2. Student books time slot.
  3. Officer accepts or denies request.
* **Level 3**:
  1. Officer logs into the system and selects available dates.
  2. The system sends availability to students.
  3. Student chooses an available slot and submits a booking request.
  4. The system sends the officer a notification to approve the request.
  5. Upon approval, the system sends the student a confirmation email with meeting details.

**Explanation** : Business processes help us understand how things should work step by step, from the big picture down to the details.

**3. User Personas**

**Definition**: User personas are fictional characters that represent the different types of people who will use the system. They help us understand the users' needs.

**Example**:

* **Cristina Pokharel (Student Persona)**:
  + **Age**: 20
  + **Occupation**: University student
  + **Technology Use**: Familiar with booking systems, prefers intuitive mobile apps
  + **Goals**: Cristina wants to book a session with an officer for academic guidance and receive a Teams meeting link for online sessions.
  + **Challenges**: Cristina finds it difficult when there are limited time slots, or when there are delays in receiving notifications.
* **Anjay Pokharel (Officer Persona)**:
  + **Age**: 38
  + **Occupation**: University officer
  + **Technology Use**: Proficient with administrative systems
  + **Goals**: Anjay wants an easy way to manage her availability and communicate with students. He prefers automated reminders for upcoming sessions.
  + **Challenges**: Anjay struggles with managing high volumes of student requests within a limited time frame.

**Explanation** : User personas help the team keep the end users in mind throughout the development process. By understanding the different types of users, we can design features and processes that cater to their specific needs, ensuring the system is user-friendly and effective.

**4. User Stories**

**Definition**: A user story is a short statement that describes what a user wants to do and why. It helps us understand what features are important.Example:

* **Student’s User Story**:
  + "As a student, I want to book an appointment with an officer online so that I can get academic guidance without having to go to campus."
* **Officer’s User Story**:
  + "As an officer, I want to set my availability for appointments so that I can manage my schedule efficiently and avoid overlapping sessions."

**Explanation:** User stories are key for agile development. They define specific features or functionality from the user's perspective, ensuring that each development task ties back to user needs and expectations. These stories provide clear, actionable goals for the development team.

**5. Requirements**

Requirements are the detailed instructions that describe what the system should do. These can be functional (what the system does) or non-functional (how the system performs).

**Example**:

* **Functional Requirements**:
  + "The system must allow officers to set their availability for appointments, specifying whether sessions are online or offline."
  + "The system must allow students to book available time slots, receive confirmation emails, and get a Teams link or location details."
* **Non-Functional Requirements**:
  + "The system must ensure email delivery within 5 minutes of booking confirmation."
  + "The system must support up to 1000 concurrent users without performance degradation."

**Explanation:** Requirements are the specifics of what the system needs to do. Functional requirements tell us what features we need to build, and non-functional requirements tell us how the system should perform (e.g., speed, security).

**Interconnections**

* **Capabilities** represent the high-level goals that the system needs to achieve.
* **Business Processes** detail the steps needed to deliver those capabilities.
* **User Personas** guide the understanding of who the system is for and what their needs are.
* **User Stories** translate user personas' needs into specific features or functionality.
* **Requirements** define exactly what the system needs to do to fulfill the user stories and capabilitie

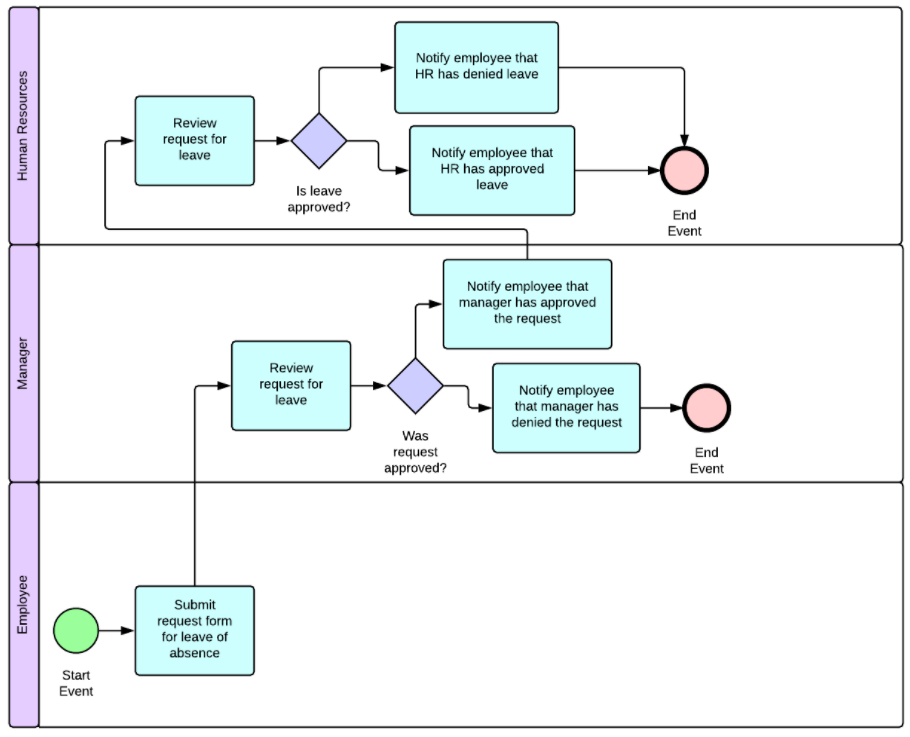
# 1. Business Process Model and Notation (BPMN) Diagram

A **Business Process Model and Notation (BPMN) Diagram** is a visual tool used to map out the steps of a business process. It uses standardized symbols to show tasks, decision points, events, and the flow between them, making it easier for teams to understand and analyze how a process works.

**Key Components of a BPMN Diagram:**

1. **Swimlanes**: Represent different roles or departments involved in the process. Each swimlane shows the tasks performed by a particular participant (e.g., Employee, Manager, HR).
2. **Events**:
   * **Start Event**: The green circle indicates where the process begins.
   * **End Event**: The red circle marks the end of the process.
3. **Tasks/Activities**: Represented by rounded rectangles, they show the specific actions that need to be completed (e.g., "Submit request form for leave of absence").
4. **Decision Gateways**: Diamond shapes used to represent points where a decision is made (e.g., "Is leave approved?" or "Was request approved?").
5. **Flow/Arrows**: Show the sequence of steps or the flow of the process from one activity to another.

This BPMN **(Business Process Model and Notation) Diagram** that represents the workflow for submitting and approving a **Leave Request**.



In this **Leave Request Process** BPMN diagram:

**Employee’s Lane:**

1. The **Employee** starts the process by **submitting a request form for leave of absence** (start event).

**Manager’s Lane:**

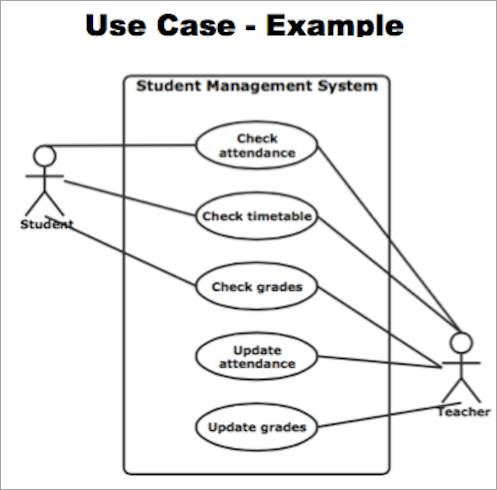
1. The **Manager** receives the request and **reviews the request for leave**.
2. A decision is made: **Was the request approved?**
   * **If approved**, the manager **notifies the employee** that the request is approved.
   * **If denied**, the manager **notifies the employee** that the request is denied.

**HR’s Lane:**

1. After the manager's decision, **HR reviews the request**.
2. A decision is made: **Is the leave approved by HR?**
   * **If approved**, HR **notifies the employee** that the leave is approved.
   * **If denied**, HR **notifies the employee** that the leave is denied.
3. The process ends with the **End Event** after HR’s notification.

# 2. Use Case Diagram

A Use Case Diagram shows how users (actors) interact with the system and what actions they can perform.



**Use Cases**: Inside the Student Management System, there are five use cases:

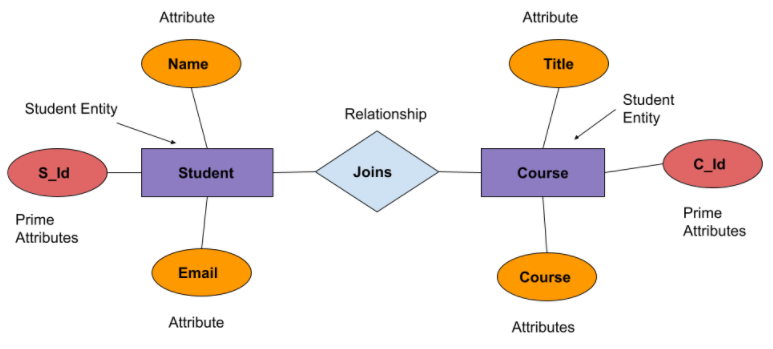
* + Check attendance: Both the student and teacher can check attendance.
  + Check timetable: Only the student can check their timetable.
  + Check grades: Both the student and teacher can view grades.
  + Update attendance: Only the teacher can update attendance.
  + Update grades: Only the teacher can update grades.

# 3. Entity-Relationship (ER) Diagram

An Entity-Relationship (ER) Diagram is used to visually represent the structure of a database, showing how different entities (objects) relate to each other.

Key Components in an ER Diagram:

1. **Entities**: These are objects or things in the system (e.g., Student and Course). Entities are represented by rectangles.
2. Attributes: These describe properties or details of an entity (e.g., Name, Email for the Student entity and Title, Course for the Course entity). Attributes are represented by ovals connected to the entity they describe.
3. **Primary Key:** These are unique identifiers for each entity instance (e.g., S\_Id for Student, C\_Id for Course). They are represented as attributes with an oval and often marked in red or underlined.
4. **Relationships:** These represent the association between entities (e.g., Joins between Student and Course). Relationships are depicted by a diamond shape.



For example:

**Entities**:

**Student** with attributes:

* S\_Id (Primary Key)
* Name
* Email

**Course** with attributes:

* C\_Id (Primary Key)
* Title
* Course

**Relationship**:

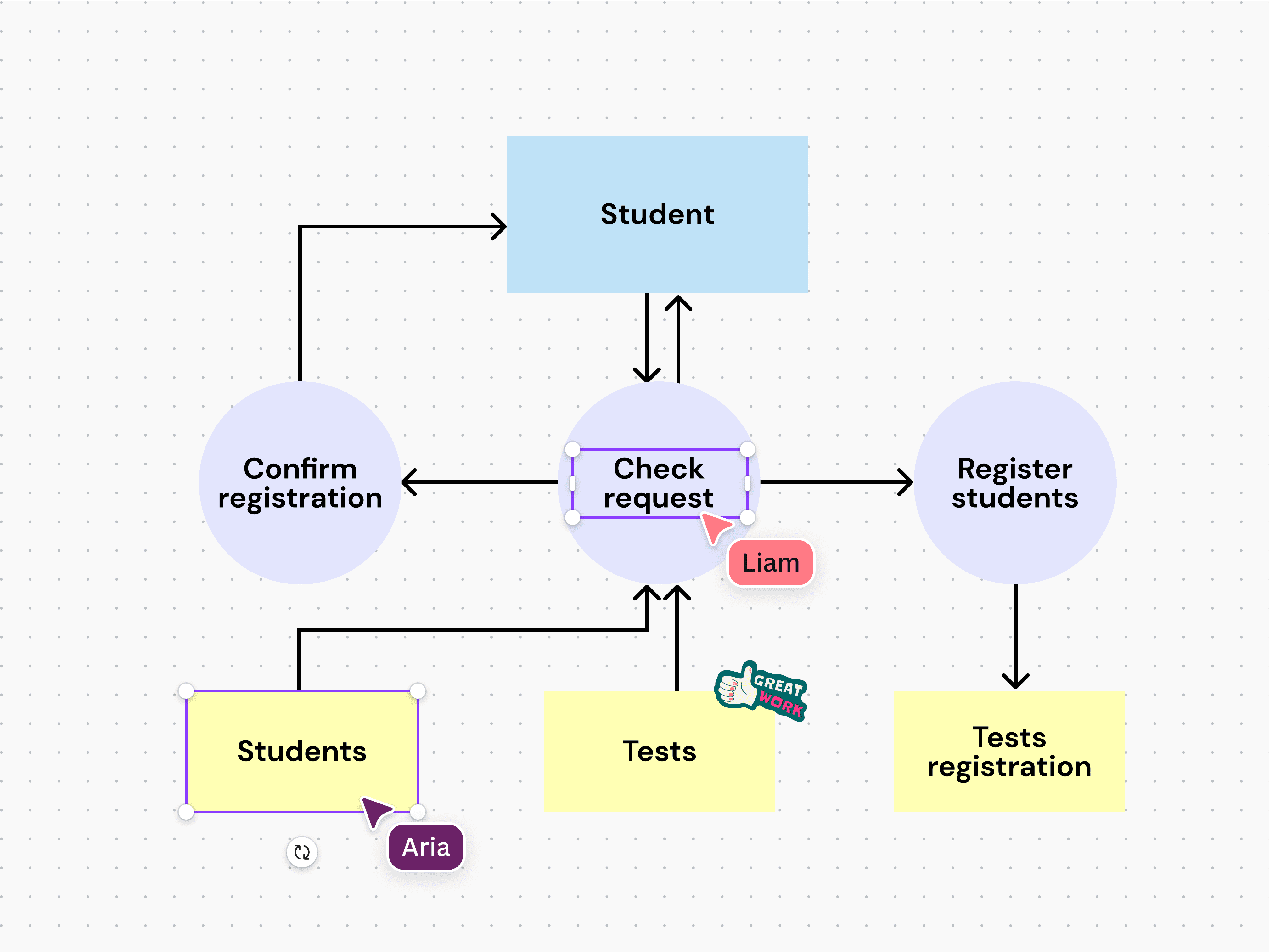
The Joins relationship indicates that a student can enroll or join a course. This relationship connects the Student and Course entities.

**4. Data Flow Diagram (DFD)**

A DFD represents how data flows through the system. It shows where data comes from, how it is processed, and where it goes.

**Key Components of a DFD:**

1. External Entities:
   * These are sources or destinations of data that interact with the system but are outside its boundaries.
   * Represented by rectangles.
   * Example: In a student registration system, a student could be an external entity.
2. **Processes:**
   * These show activities that transform incoming data into outgoing data.
   * Represented by circles or rounded rectangles.
   * Example: A process might be to check requests, register students, or confirm registration.
3. **Data Stores:**
   * These represent where data is stored for later use, like a database or a file.
   * Represented by open-ended rectangles.
   * Example: Data stores in a student system could be students and tests databases.
4. **Data Flows:**
   * Arrows that show the direction of data movement between processes, data stores, and external entities.
   * Represented by arrows.
   * Example: Data flows might show how student registration details flow from the student to the register student’s process.



In the **Booking System**:

* Data (student booking request) flows from the "Student" to the "System", then is sent to the "Officer" for approval, and finally returns as a confirmation email to the student.

# 5. Wireframe or Mockup Diagram

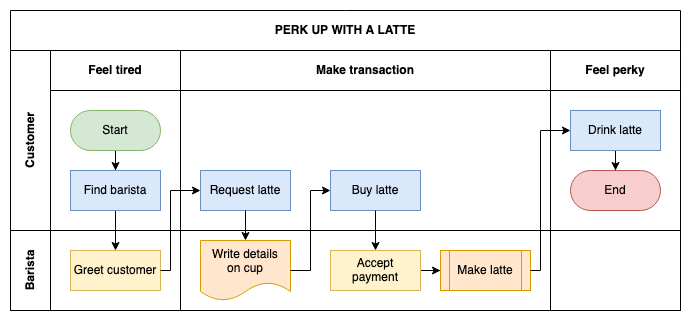
A Wireframe or Mockup diagram is a visual guide that represents the skeletal framework of a digital interface, such as a website or app. It's typically used in the early stages of design to showcase the layout and structure of elements on a screen without including the detailed design or branding aspects.

Wireframes and mockups are essential in the early stages of UI/UX design, helping ensure that both the layout and visual aesthetics align with user needs and project goals.

**Example**:  


# 6. Swimlane Diagram (Cross-Functional Flowchart)

A Swimlane Diagram shows who is responsible for different tasks in a process. Each "lane" represents a different role or department.

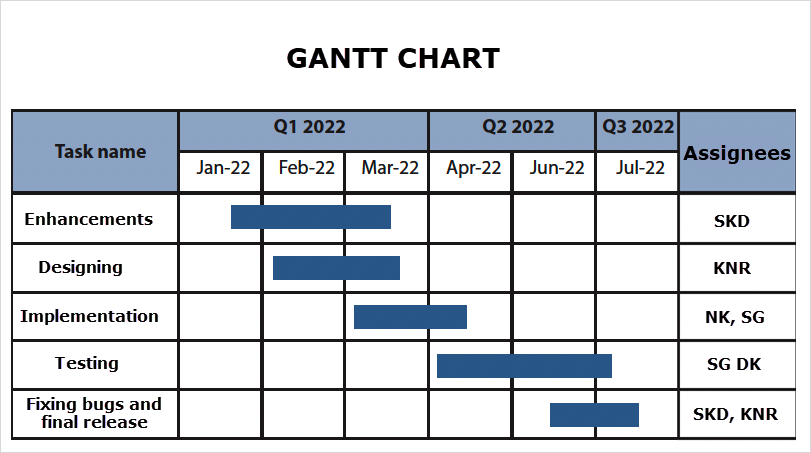


This above Swimlane Diagram illustrates the customer and barista's roles in the process of purchasing a latte, separating their responsibilities into clear lanes. The process is broken down into simple steps that show how the customer’s actions flow into the barista’s actions and vice versa.  
For the **Perk Up with a Latte** process:

* **Lanes**: "Customer", "Barista"
  + **Customer feels tired** (in the Customer lane).
  + **Customer finds the barista** (in the Customer lane).
  + **The barista greets the customer** (in the Barista lane).
  + **Customer requests a latte** (in the Customer lane).
  + **The barista writes details on the cup** (in the Barista lane).
  + **Customer buys the latte** (in the Customer lane).
  + **The Barista accepts payment** (in the Barista lane).
  + **The barista makes the latte** (in the Barista lane).
  + **Customer drinks the latte and feels refreshed** (in the Customer lane, end event).

# 7. Gantt Chart

A Gantt Chart is a project management tool that shows tasks along a timeline. It visually represents when each task starts and ends, making it easy to track the project’s progress and see task dependencies.

**Example**:  


In the given Gantt chart:

* **Enhancements**: Runs from Feb to Mar 2022, assigned to **SKD**.
* **Designing**: Takes place Feb to Mar 2022, assigned to **KNR**.
* **Implementation**: Scheduled for Mar to Apr 2022, handled by **NK** and **SG**.
* **Testing**: Occurs from Apr to Jun 2022, assigned to **SG** and **DK**.
* **Fixing Bugs and Final Release**: Runs from Jun to Jul 2022, handled by **SKD** and **KNR**.

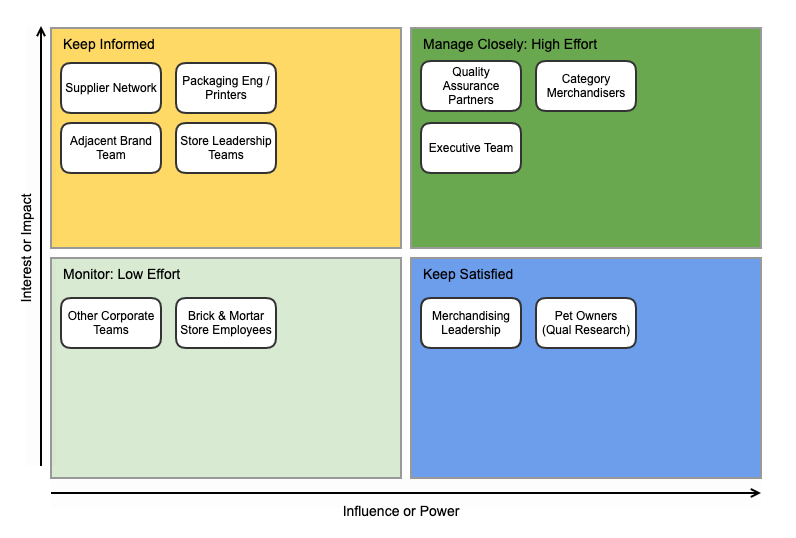
It helps track task durations, responsibilities, and ensures the project stays on schedule.

# 8. Stakeholder Map

A Stakeholder Map visually represents the key stakeholders involved in a project and their level of influence or interest. It Helps in identifying and managing relationships with stakeholders, understanding who to prioritize in decision-making.

**Components of a Stakeholder Map:**

1. **Interest or Impact** (Y-axis): This indicates how much the stakeholder is affected by or interested in the project's outcome.
2. **Influence or Power** (X-axis): This measures how much control or influence the stakeholder has over the project.



Stakeholder Groups in the above diagram:

* **Manage Closely (High Effort):** These are high-power, high-interest stakeholders (e.g., Quality Assurance Partners, Executive Team). They require close management and regular communication.
* **Keep Informed:** These are high-interest, low-power stakeholders (e.g., Supplier Network, Store Leadership Teams). They need regular updates but don’t require heavy engagement.
* **Keep Satisfied:** These are high-power, low-interest stakeholders (e.g., Merchandising Leadership). They should be kept satisfied but with less frequent communication.
* **Monitor (Low Effort):** These are low-power, low-interest stakeholders (e.g., Other Corporate Teams). They require minimal attention but should still be monitored.

This map helps prioritize efforts and strategies for managing stakeholder relationships based on their influence and interest in the project.

# 9. State Diagram

A **State Diagram** is a visual representation of the different states an object (or system) goes through during its lifecycle in response to events.

**Key Components**:

Represents the condition or situation of an object at a specific point in time (e.g., "Student enrolled").

* States: Conditions of the object (e.g., "Student enrolled").
* Transitions: Arrows showing changes from one state to another.
* Initial State: The starting point (filled circle).
* Final State: The endpoint (circle with a dot).



In this **Student Enrollment State Diagram**:

* The process starts with Student enrolled.
* The student is assigned a StudentID and a Supervisor.
* A decision leads to Subjects selected, followed by Timetable sent.
* The process ends at the final state (circle with a dot).

This state diagram effectively models the different stages a student goes through after enrolling, from assigning an ID to receiving their timetable.

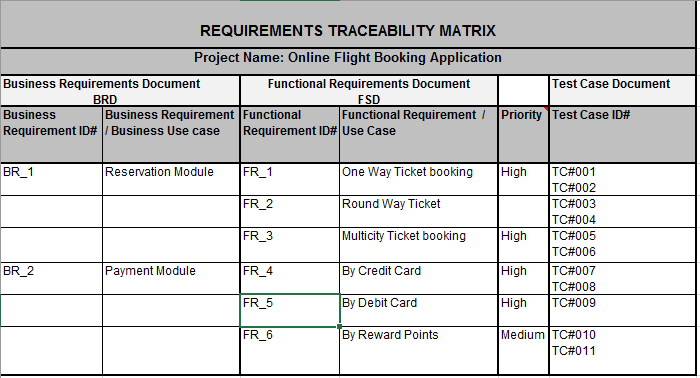
# 10. Requirement Traceability Matrix (RTM)

A Requirements Traceability Matrix (RTM) is a document that helps ensure that all requirements defined for a system are tested throughout the development lifecycle. Ensures all requirements are tested, making sure nothing is missed during testing.

**Key Components of RTM:**

1. **Business Requirements Document (BRD)**: Lists the high-level business requirements or use cases.
2. **Functional Requirements Document (FSD)**: Breaks down the business requirements into specific functional requirements.
3. **Test Case Document**: Maps test cases to each functional requirement, ensuring that each requirement is properly tested.
4. **Priority**: Indicates the importance of each requirement (e.g., High, Medium, Low).

Here is the A Requirements Traceability Matrix (RTM) for Online Flight Booking Application.

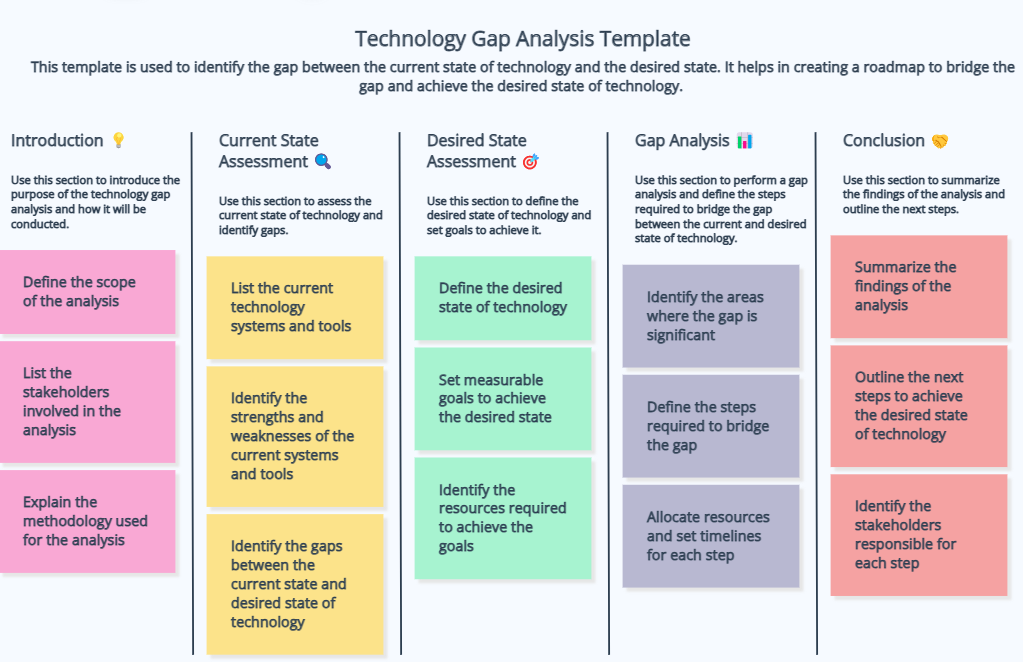


# 11. Gap Analysis Diagram

A Gap Analysis Diagram is a tool used to compare the current state of a process, system, or organization with the desired future state. It identifies the gaps or differences between where you are now (current state) and where you want to be (future state), helping to pinpoint areas that need improvement or action**.**  
The given image is a **Technology Gap Analysis Template** that outlines the steps needed to bridge the gap between the current state of technology and the desired future state. It helps in identifying areas for improvement and setting a roadmap for achieving the desired technology goals.

**Key Components:**

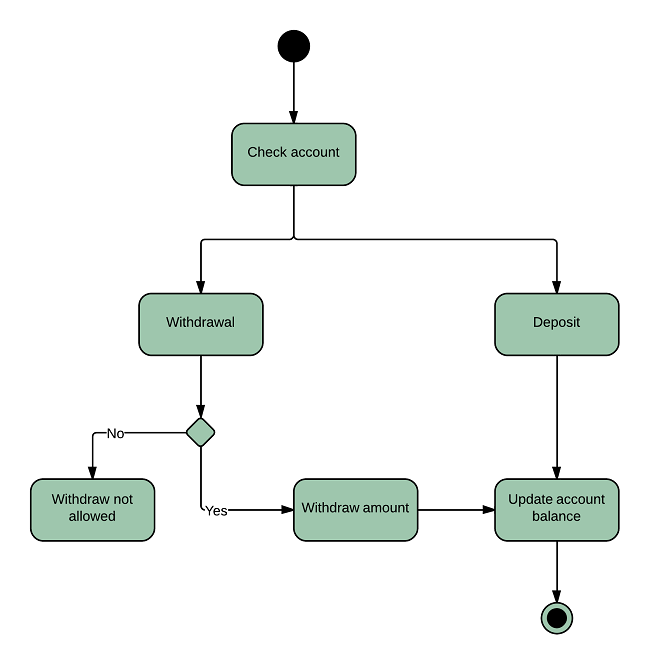
1. **Introduction**: Defines scope, stakeholders, and methodology.
2. **Current State Assessment**: Reviews current technology and identifies gaps.
3. **Desired State Assessment**: Defines goals and resources for the future state.
4. **Gap Analysis**: Identifies gaps and steps to bridge them.
5. **Conclusion**: Summarizes findings and outlines next steps.

****

# 12. Activity Diagram

**Key Components of an Activity Diagram:**

1. **Start Node:** A filled black circle that indicates the beginning of the process.
2. **Activities:** Represent tasks or steps in the process, typically shown as rounded rectangles.
3. **Transitions/Activity Edge:** Arrows that show the flow from one activity to the next.
4. **Decision Nodes:** A diamond shape where branching occurs based on conditions or decisions.
5. **Fork:** A thick horizontal or vertical line indicating parallel activities (the process splits into multiple paths).
6. **Join:** A thick line where multiple parallel activities converge into a single path.
7. **End Node:** A filled circle with an outer ring, indicating the end of the process.



**Key Steps in the Diagram:**

1. **Start Node** (Black circle):
   * The process begins with the start node, indicating the beginning of the flow.
2. **Check Account**:
   * The account is checked to see whether the user wants to make a withdrawal or deposit.
3. **Decision Node** (Forks into Withdrawal and Deposit):
   * The process splits into two potential actions:
     + **Withdrawal**: The user selects to withdraw money from the account.
     + **Deposit**: The user selects to deposit money into the account.
4. **Withdrawal Path**:
   * After selecting **withdrawal**, a decision point checks whether the withdrawal is allowed:
     + **No**: If the withdrawal is not allowed (due to insufficient funds, for example), the process ends with "Withdraw not allowed".
     + **Yes**: If the withdrawal is allowed, the withdrawal amount is processed, and the balance is updated.
5. **Deposit Path**:
   * If the user selects **Deposit**, the deposit amount is added, and the system proceeds to update the account balance.
6. **Update Account Balance**:
   * Whether it’s a successful withdrawal or a deposit, the system updates the account balance as the final step.
7. **End Node** (Circle with inner dot):
   * The process concludes at the end node, marking the end of the workflow.