Name of Student: Pushkar Sane		
Roll Number: 45		**Tutorial Number: 7**
Title of Tutorial: UML Diagrams (Sequence Diagram)		
DOP: 25-09-2023		**DOS: 26-09-2023**
CO Mapped: **CO1, CO2, CO3**	**PO Mapped:** **PO1, PO4, PO6**	**Signature:**

Tutorial No. 7

Aim: UML Diagrams (Sequence Diagrams)

Description:

- **Introduction to Sequence Diagrams in UML**
  What Are Sequence Diagrams in UML?
  Sequence diagrams are a fundamental type of diagram in the Unified Modelling Language (UML), a widely used notation for visualizing and documenting software systems. UML sequence diagrams are particularly valuable for representing the dynamic aspects of a system. They provide a visual way to illustrate how different components, objects, or actors interact with each other over time.

- **Purpose and Importance of Sequence Diagrams**
  Sequence diagrams serve several crucial purposes in software engineering:
  1. **Modeling Behavior:** They model the behavior of a system, capturing the flow of interactions and messages exchanged between objects or components during the execution of a particular use case or scenario.
  2. **Communication:** Sequence diagrams facilitate communication among project stakeholders. They provide a common visual language that developers, designers, business analysts, and other team members can use to discuss and understand system behavior.
  3. **Validation and Testing:** Sequence diagrams can be used to validate the correctness of a system's behavior and are valuable for designing test cases and ensuring that the system functions as intended.
  4. **Documentation:** They are a form of documentation that helps in comprehending and maintaining the software system. They serve as a reference for future development, troubleshooting, and enhancements.

- **Elements of Sequence Diagrams**

  Key Elements of a Sequence Diagram are mentioned below:

  1. **Lifelines:** Lifelines represent objects or actors participating in the sequence of interactions. Each lifeline is depicted as a vertical dashed line and is labelled with the name of the object or actor it represents. Lifelines show the existence and life cycle of these entities during the sequence.
  2. **Messages:** Messages are horizontal arrows that connect lifelines, indicating the flow of communication between objects. There are two primary types of messages:
     a. **Synchronous Messages:** Represented by solid arrows, synchronous messages indicate that the sender waits for a response from the receiver before proceeding.
     b. **Asynchronous Messages:** Represented by open arrows, asynchronous messages indicate that the sender does not wait for an immediate response from the receiver and continues with its own activities.
  3. **Activation Bars:** Activation bars, also known as execution occurrences, are boxes or rectangles that appear on lifelines. They depict the duration during which an object is actively processing a message or performing an action. Activation bars provide a visual representation of the timeline of message execution.

- **Guidelines for Creating Sequence Diagrams**

  1. **Identify the Actors:** Begin by identifying the key actors or components involved in the scenario you want to model. These will become the lifelines in your sequence diagram.

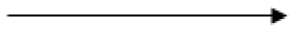
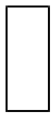
  2. **Define the Scenario:** Clearly define the scenario or use case you are modeling. Understand the specific interactions and the order in which they occur.

  3. **Sequence Messages:** Sequence the messages between lifelines to show the chronological order of interactions. Use synchronous and asynchronous messages appropriately based on the scenario.

4. **Include Control Flow:** Sequence diagrams can also include control flow elements like loops, conditionals, and alternative paths to capture complex behavior.

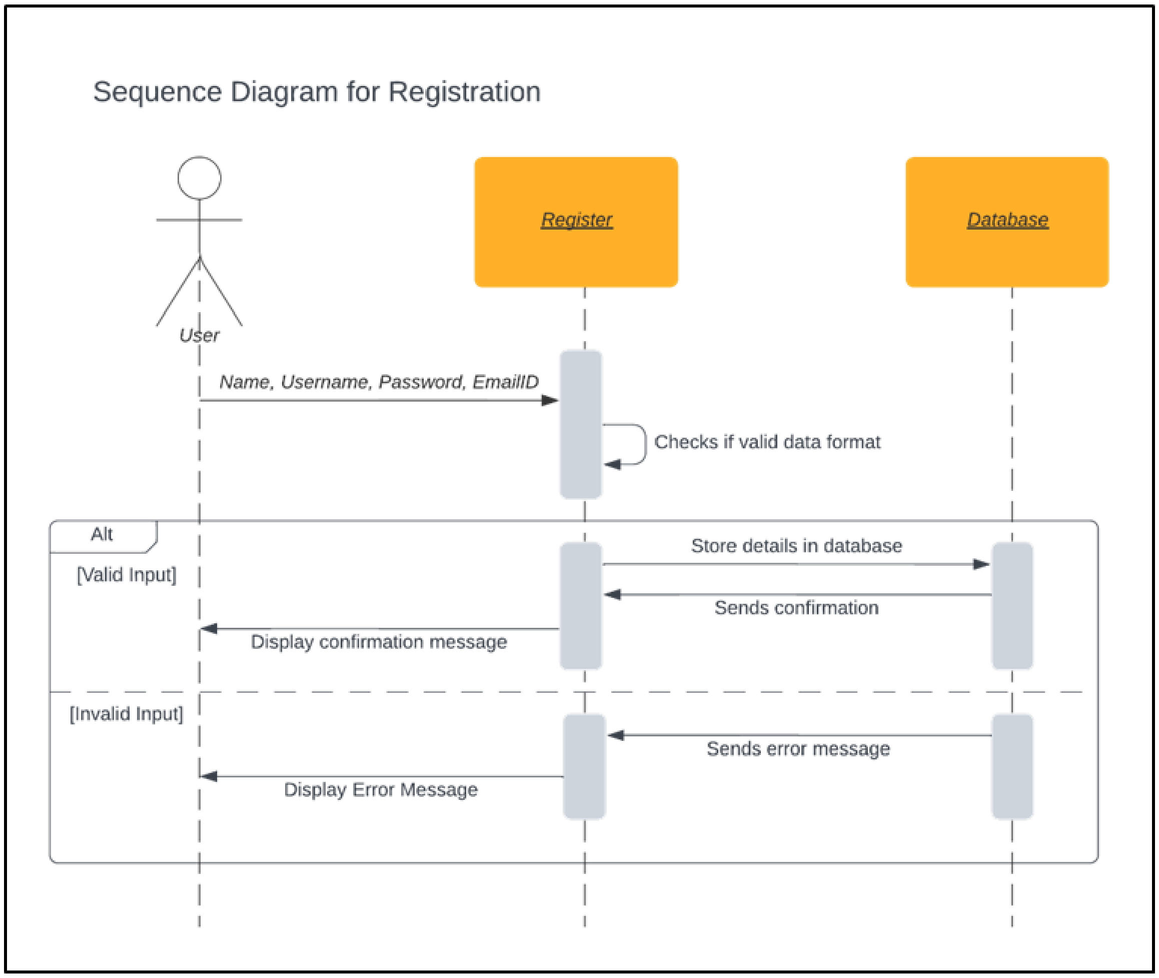
5. **Use Activation Bars Wisely:** Utilize activation bars to illustrate the duration of message processing or method execution. Avoid unnecessary clutter by keeping them concise.

- **Notations and Symbols**

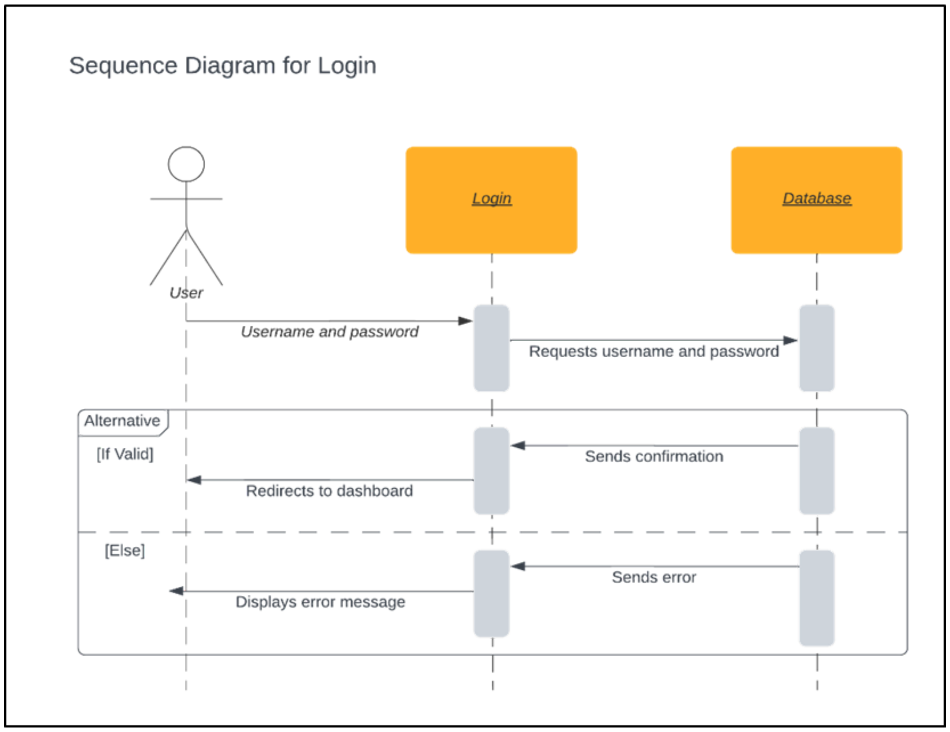
Name	Symbol	Description
Synchronous Message		An instantaneous communication between objects that conveys information, with the expectation that an action will be initiated as a result.
Activation Box		The period during which an object is performing an action.
Object		An object that is created, performs actions, and/or is destroyed during the lifeline

- **Problem Statement**
  To create a live cricket scoring app. Here, various types of functions are provided such as ball-by-ball scoring, professional scorecard etc.

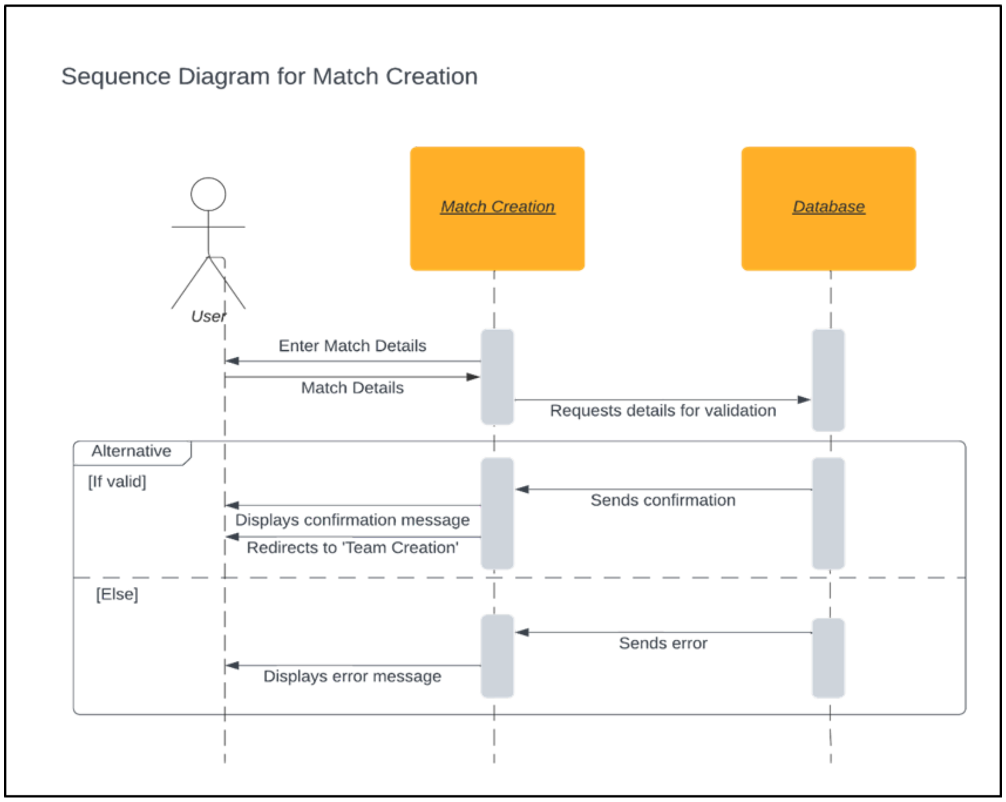
1) Sequence diagram for registration


Sequence Diagram for Registration
User
Register
Database
Name, Username, Password, EmailID
Checks if valid data format
Alt
[Valid Input]
Store details in database
Sends confirmation
Display confirmation message
[Invalid Input]
Sends error message
Display Error Message

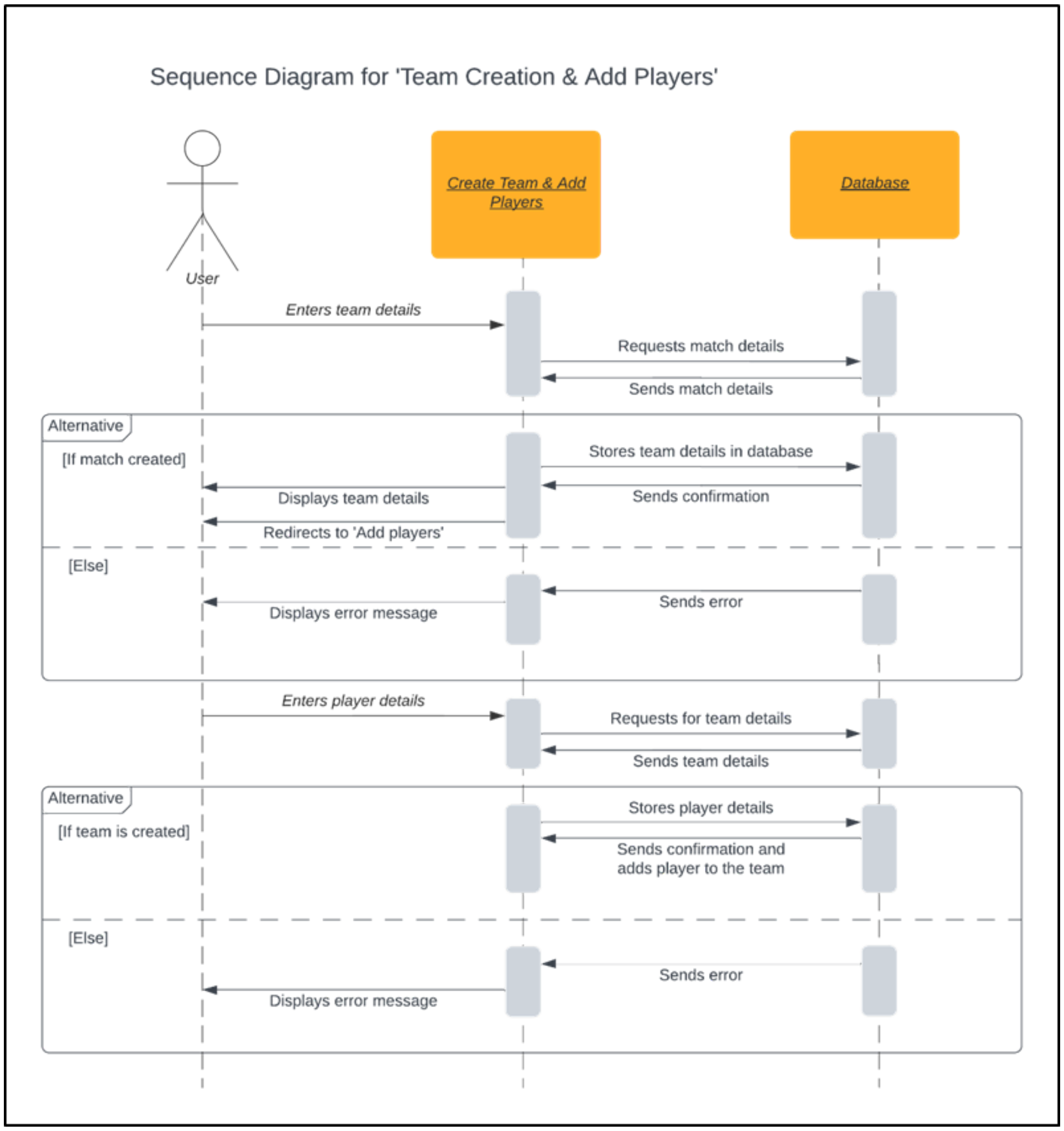
2) Sequence diagram for Login



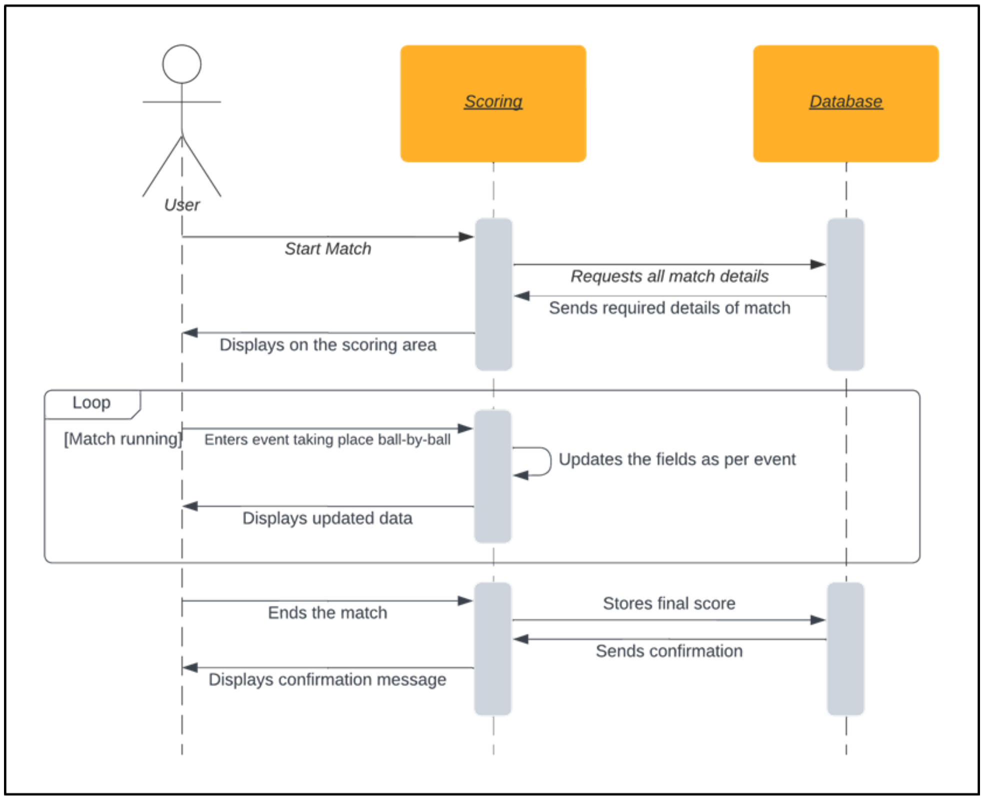
3) Sequence diagram for Match Creation



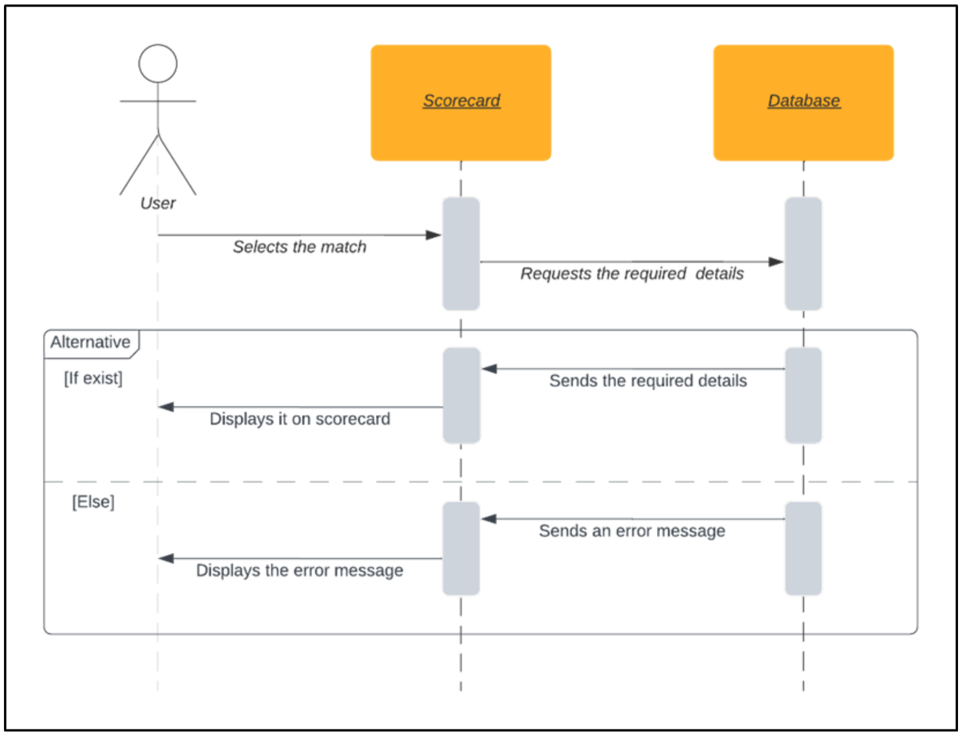
4) Sequence diagram for Creation of teams and players



5) Sequence diagram for Live Scoring


User
Scoring
Database
Start Match
Requests all match details
Sends required details of match
Displays on the scoring area
Loop
[Match running]
Enters event taking place ball-by-ball
Updates the fields as per event
Displays updated data
Ends the match
Stores final score
Sends confirmation
Displays confirmation message

6) Sequence diagram for Scorecard



- **Advanced Sequence Diagram Concepts**
    1. **Combined Fragments:** Combined fragments are used to express complex interactions and conditions in a sequence diagram. They allow you to represent loops, conditionals, parallelism, and alternatives within the sequence.
    2. **Interaction Use:** Interaction uses are a way to reuse parts of a sequence diagram within another diagram, promoting modular design and reducing redundancy.

- **Additional Considerations**
    1. **Tool Support:** Many UML modeling tools provide features for creating and editing sequence diagrams, making it easier to generate and maintain them as your system evolves.
    2. **Scalability:** For large and complex systems, consider breaking down sequence diagrams into smaller, more manageable parts to maintain readability.
    3. **Consistency:** Ensure consistency between your sequence diagrams and other UML diagrams, such as class diagrams, to maintain a holistic view of your system.
    4. **Documentation:** Sequence diagrams, like all UML diagrams, should be well documented. Include explanations, annotations, and notes to clarify any ambiguities.

Conclusion:

In conclusion, sequence diagrams are a powerful tool in UML for modelling and documenting the dynamic behavior of software systems. When used effectively, they enhance communication among stakeholders, aid in system understanding, and contribute to the overall success of software development projects. By mastering the creation and interpretation of sequence diagrams, software professionals can improve the quality and efficiency of their work.