Activity based

Project Report on

Software Engineering

Project Module - II

Submitted to Vishwakarma University, Pune

Under the Initiative of

Contemporary Curriculum, Pedagogy, and Practice (C2P2)

By

Shravan Meshram

SRN No : 202101425

Roll No : 31

Div : E

Third Year Engineering

Department of Computer Engineering

Faculty of Science and Technology

Academic Year

2023-2024

Software Engineering: Project Module II

Project Name : SRS Plan and UML Diagram for Telegram

Part I : Write a Software Requirement Specification (SRS) Plan for Telegram.

Explanation of each technique and its applicability in mentioned project

❖ Introduction

Telegram is a versatile and feature-rich messaging application that has gained widespread popularity for its robust functionality, security features, and user-friendly interface. Launched in 2013 by brothers Nikolai and Pavel Durov, Telegram offers a platform for seamless communication through text, voice, and multimedia messages, with a focus on speed and privacy.

As a cloud-based messaging app, Telegram allows users to exchange messages and media files across various devices, including smartphones, tablets, and desktop computers, while ensuring synchronization and accessibility from anywhere with an internet connection. Its cross-platform compatibility and synchronization capabilities make it a convenient choice for users seeking a unified messaging experience.

Telegram's feature set extends beyond basic messaging, offering a range of functionalities to cater to diverse user needs. Users can create and join groups and channels to connect with communities and share interests, while channel owners can broadcast messages to large audiences. The platform also supports bots, which are automated programs that can perform various tasks and provide services within the app, enriching the user experience with additional features and capabilities.

In addition to its user-oriented features, Telegram provides extensive documentation and resources for developers, enabling them to build custom bots, integrate external services, and develop third-party applications using the Telegram API. This openness and flexibility have contributed to Telegram's growing ecosystem of third-party tools and services, further enhancing its appeal to users and developers alike.

With its combination of advanced features, security measures, and developer-friendly platform, Telegram continues to attract millions of users worldwide, serving as a leading messaging solution for personal, professional, and community-based communication needs.

❖ Functional Requirements

Functional requirements define the specific actions and functionalities that a system or software application must perform to satisfy the needs of its users. In the case of Telegram, functional requirements would encompass the various features and capabilities that the messaging application offers to its users. Here are some examples of functional requirements for Telegram:

User Registration and Authentication:

- Users should be able to create an account on Telegram using their mobile number or email address.
- The application should verify user identities and authenticate them securely.

Messaging:

- Users should be able to send text messages, photos, videos, audio files, and other multimedia content to individuals or groups.
- The messaging feature should support real-time communication and synchronization across devices.

Group Chats:

- Users should be able to create, join, and participate in group chats with multiple participants.
- Group chat functionality should include features such as adding and removing members, changing group settings, and administering group permissions.

Channels:

- Users should be able to create and manage channels for broadcasting messages to large audiences.
- Channel owners should have the ability to customize channel settings, invite subscribers, and moderate content.

Security and Privacy:

- Telegram should provide end-to-end encryption for Secret Chats to ensure secure communication between users.
- Users should have control over their privacy settings, including options for blocking users, hiding online status, and controlling who can add them to groups or channels.

Notifications:

- The application should provide notifications for new messages, mentions, replies, and other relevant activities.
- Users should be able to customize notification settings based on their preferences.

Search and Discovery:

- Users should be able to search for other users, groups, channels, and messages within the application.
- Telegram should provide discovery features to help users find and join relevant groups and channels based on their interests.

Bots and Integrations:

- Telegram should support bots, which are automated programs that can interact with users and perform various tasks.
- Users should be able to integrate third-party services and applications with Telegram to extend its functionality.

File Sharing and Cloud Storage:

- Users should be able to share files of various types and sizes, including documents, images, and videos.
- Telegram should offer cloud-based storage for storing and accessing media files, documents, and chat history across devices.

❖ Non-Functional Requirements

Non-functional requirements specify the qualities or attributes of a system or software application, rather than its specific functionalities. These requirements define how well the system performs certain functions rather than what functions it performs. Here are some examples of non-functional requirements for Telegram:

Performance:

- The application should provide fast and responsive user interactions, with minimal latency in sending and receiving messages.
- It should be able to handle a large volume of concurrent users and messages without significant degradation in performance.

Reliability:

- Telegram should be highly reliable, with minimal downtime and service interruptions.
- Messages should be delivered reliably, and users should not experience data loss or message duplication.

Scalability:

- The application should be scalable to accommodate a growing user base and increasing message traffic.
- It should be able to scale horizontally by adding more servers or instances to handle increased load.

Security:

- Telegram should prioritize the security and privacy of user data, with robust encryption mechanisms to protect messages and user information.
- The application should implement security best practices to prevent unauthorized access, data breaches, and cyber attacks.

Availability:

- The application should be available 24/7, with high uptime and minimal scheduled maintenance windows.
- It should have redundant systems and failover mechanisms to ensure continuous availability in case of hardware or software failures.

Usability:

- Telegram should have a user-friendly interface that is intuitive and easy to navigate.
- It should provide clear feedback to users on their actions, such as message delivery status and notification alerts.

Compatibility:

- The application should be compatible with a wide range of devices and operating systems, including smartphones, tablets, and desktop computers.
- It should support multiple platforms and browsers to ensure accessibility for users.

Regulatory Compliance:

- Telegram should comply with relevant data protection and privacy regulations, such as GDPR (General Data Protection Regulation) in Europe.

- It should implement measures to ensure user data is handled and stored securely, and user privacy is respected.

Performance Efficiency:

- The application should optimize resource utilization, such as CPU, memory, and network bandwidth, to minimize resource consumption and maximize efficiency.
- It should use caching and compression techniques to reduce data transfer and improve performance.

❖ System Requirements

System requirements specify the hardware, software, and other infrastructure components needed to support the operation and deployment of a software application like Telegram. Here are some examples of system requirements for Telegram:

Operating System:

- Telegram is compatible with various operating systems, including:
- Android: Versions 5.0 (Lollipop) and above
- iOS: Versions 9.0 and above
- Windows: Windows 7 and above
- macOS: macOS 10.12 (Sierra) and above
- Linux: Ubuntu 12.04 and above, Fedora, Debian, etc.

Hardware Requirements:

- Minimum hardware specifications for devices running Telegram:
- Android/iOS Devices: Modern smartphones or tablets with sufficient memory and storage space.
- Desktop Computers: Standard hardware configuration with adequate RAM and disk space.

Internet Connection:

- A stable and high-speed internet connection is required for sending and receiving messages, media files, and other data within Telegram.

Supported Browsers:

- For accessing Telegram via web browser:
- Latest versions of popular browsers such as Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, etc.

Network Requirements:

- Telegram requires access to the internet for connecting to its servers and transmitting messages securely.
- It should support both Wi-Fi and mobile data connections for seamless communication.

Software Dependencies:

- Telegram may require certain software dependencies or libraries to be installed on the device or system for proper functioning.
- This may include runtime environments, encryption libraries, multimedia codecs, etc.

Storage Space:

- Telegram may require a certain amount of storage space on the device or system for caching data, storing media files, and maintaining chat history.
- Users should have sufficient free space available to store downloaded media files and chat backups.

Access Permissions:

- Users may need to grant permissions to the Telegram application to access certain device features such as camera, microphone, contacts, location, etc., depending on the platform.

Updates and Maintenance:

- Users should ensure that their devices and operating systems are up to date with the latest security patches and updates to ensure compatibility and security when using Telegram.

❖ User Requirements

User requirements outline the needs, expectations, and preferences of the users regarding the functionality, usability, and features of a software application like Telegram. These requirements serve as the foundation for designing and developing a system that meets the users' needs effectively. Here are some examples of user requirements for Telegram:

Messaging Features:

- Users should be able to send and receive text messages, multimedia files (photos, videos, audio), stickers, and emojis.
- The messaging interface should support real-time communication with minimal latency.

Group Communication:

- Users should be able to create and participate in group chats with multiple participants.
- Group chat functionality should include features such as adding/removing members, changing group settings, and mentioning users.

Channel Subscriptions:

- Users should be able to discover and subscribe to channels to receive updates, news, and content from various sources.
- Channel owners should have the ability to broadcast messages to their subscribers.

Privacy and Security:

- Users should have control over their privacy settings, including options for blocking/unblocking users, hiding online status, and controlling who can add them to groups or channels.
- The application should provide end-to-end encryption for Secret Chats to ensure secure communication.

Notifications and Alerts:

- Users should receive notifications for new messages, mentions, replies, and other relevant activities.
- Notification settings should be customizable to allow users to manage their notification preferences.

Search and Discovery:

- Users should be able to search for other users, groups, channels, and messages within the application.
- Telegram should provide discovery features to help users find and join relevant groups and channels based on their interests.

Accessibility:

- The application should be accessible to users with disabilities, with support for features such as screen readers, keyboard navigation, and alternative input methods.
- The user interface should be intuitive and easy to navigate for users of all skill levels.

Cross-Platform Compatibility:

- Telegram should be available on multiple platforms, including Android, iOS, Windows, macOS, and web browsers, to ensure seamless access across devices.

Performance and Reliability:

- Users expect Telegram to be fast, responsive, and reliable, with minimal downtime and service interruptions.
- Messages should be delivered reliably, and users should not experience data loss or message duplication.

Customization and Personalization:

- Users should have options to customize their profiles, chat backgrounds, themes, and notification sounds according to their preferences.
- The application should provide features for organizing and managing chats, contacts, and media files effectively.

❖ Constraints

Constraints are limitations or restrictions that may affect the design, development, or operation of a software application like Telegram. These constraints can arise from various factors such as technical limitations, regulatory requirements, resource constraints, or organizational policies. Here are some examples of constraints that may apply to Telegram:

Security and Privacy Regulations:

- Telegram must comply with data protection and privacy regulations such as GDPR (General Data Protection Regulation) in Europe and other relevant laws and regulations in different regions.
- The application must implement security measures to protect user data and ensure secure communication, including encryption and authentication mechanisms.

Platform Limitations:

- Telegram's features and functionality may be limited by the capabilities and restrictions of the platforms it operates on, such as mobile operating systems (Android, iOS), web browsers, and desktop environments.
- The application must adhere to platform-specific guidelines and restrictions imposed by app stores (e.g., Google Play Store, Apple App Store) for distribution and deployment.

Resource Constraints:

- Telegram's performance and scalability may be constrained by limitations in hardware resources (e.g., server capacity, network bandwidth) and software resources (e.g., memory, disk space).
- The application must optimize resource utilization to ensure efficient operation and minimize resource consumption.

Compatibility Requirements:

- Telegram may face constraints related to compatibility with older devices, operating systems, and web browsers that may not support the latest features or standards.
- The application must maintain backward compatibility with a wide range of devices and software versions to ensure accessibility for all users.

Network Connectivity:

- Telegram's functionality may be constrained by the availability and reliability of internet connectivity, especially in regions with limited or unstable network infrastructure.
- The application must support offline functionality and provide mechanisms for handling intermittent connectivity and network failures.

Localization and Internationalization:

- Telegram must support multiple languages and cultural preferences to cater to a diverse global user base.
- The application must adhere to localization and internationalization standards to provide a consistent user experience across different languages and regions.

Legal and Regulatory Constraints:

- Telegram may face legal and regulatory constraints related to content moderation, censorship, intellectual property rights, and compliance with government regulations in various jurisdictions.
- The application must implement policies and mechanisms to enforce legal and regulatory requirements while upholding freedom of expression and user privacy rights.

Organizational Policies and Business Requirements:

- Telegram's design and development may be constrained by organizational policies, business objectives, budgetary constraints, and strategic priorities.
- The application must align with the goals and priorities of the organization and stakeholders while meeting user needs and expectations

❖ Assumptions and Dependencies

Assumptions and dependencies are factors that are assumed to be true or external conditions that a software application like Telegram relies on for its operation. These assumptions and dependencies may impact the design, development, and deployment of the application. Here are some examples of assumptions and dependencies for Telegram:

Assumptions:

- Users have access to compatible devices (smartphones, tablets, computers) and a stable internet connection to use Telegram.
- Users are familiar with basic messaging and communication concepts and can navigate the Telegram interface with minimal guidance.
- Users have consented to the terms of service and privacy policy of Telegram and are willing to share their personal information to use the application.
- Telegram's servers and infrastructure are operational and available to handle user requests and message traffic.
- Users trust Telegram to protect their privacy and data security through encryption and other security measures.
- Telegram's development team will continue to update and maintain the application to address bugs, security vulnerabilities, and evolving user needs.

Dependencies:

- Telegram depends on third-party services and APIs for various functionalities, such as cloud storage (e.g., Amazon Web Services, Google Cloud Platform), push notifications (e.g., Firebase Cloud Messaging, Apple Push Notification Service), and analytics (e.g., Google Analytics, Firebase Analytics).
- Telegram's compatibility with different platforms (Android, iOS, Windows, macOS, Linux) depends on the availability and support of platform-specific features, libraries, and APIs.
- Telegram may rely on external libraries, frameworks, and development tools for implementing features, optimizing performance, and ensuring compatibility across devices and platforms.
- The availability and reliability of internet connectivity, network infrastructure, and data transmission services are crucial dependencies for Telegram's real-time messaging functionality.
- Telegram's compliance with legal and regulatory requirements (e.g., GDPR, data protection laws) depends on adherence to industry standards, best practices, and guidance from legal experts and regulatory authorities.

❖ Functional Models

Functional models represent the behaviour and functionality of a software system, illustrating how users interact with the system and the responses generated by the system in various scenarios. For a messaging application like Telegram, functional models can include use case diagrams, activity diagrams, and sequence diagrams. Here's how each of these functional models can be applied to Telegram:

Use Case Diagram:

- A use case diagram depicts the interactions between users (actors) and the system (Telegram) to achieve specific goals or tasks.
- Actors in a Telegram use case diagram may include regular users, group administrators, channel owners, and bots.
- Use cases represent specific actions or functionalities that users can perform within the Telegram application, such as sending messages, joining groups, creating channels, etc.
- Relationships between actors and use cases show which actors are involved in each use case.
- Use case diagrams help to identify the scope of the system and clarify user requirements and system functionalities.

Activity Diagram:

- An activity diagram illustrates the flow of activities or processes within the system, showing the sequence of actions and decision points.
- In Telegram, an activity diagram can represent the steps involved in sending a message, joining a group, or managing channel settings.
- Activities represent specific actions or tasks performed by users or the system, while transitions show the flow of control between activities.
- Decision points (branches) in the diagram indicate where different paths or actions may be taken based on conditions or user input.
- Activity diagrams provide a visual representation of the system's behavior and help to identify potential bottlenecks or areas for optimization.

Sequence Diagram:

- A sequence diagram illustrates the interactions between objects or components within the system over time, showing the sequence of messages exchanged between them.

- In Telegram, a sequence diagram can represent the exchange of messages between users, groups, channels, and bots.
- Lifelines represent the objects or components involved in the interaction, while messages indicate the communication between them.
- Sequence diagrams help to visualize the flow of messages and interactions within the system, including asynchronous and parallel activities.
- They can also depict error handling and exception scenarios, showing how the system responds to unexpected events or errors.

❖ Use Cases

Use cases describe the interactions between users (actors) and a system (Telegram) to achieve specific goals or tasks. They outline the steps involved in a particular scenario and the system's responses to user actions. Here are some example use cases for Telegram:

1. **Send Message:**
   - Actor: Regular User
   - Description: This use case describes the process of sending a message to another user or group.
   - Steps:

   1.1. User opens the Telegram application.
   1.2. User selects the recipient (user or group) to whom they want to send a message.
   1.3. User types the message content in the chat input field.
   1.4. User taps the send button to send the message.

   - Alternate Flows:
     - If the user attaches media (photo, video, audio) to the message, the attachment is sent along with the text.
2. **Join Group:**
   - Actor: Regular User
   - Description: This use case describes the process of joining a group chat on Telegram.
   - Steps:

   2.1. User receives an invitation link or searches for the group within the Telegram application.
   2.2. User selects the group from the search results or clicks on the invitation link.
   2.3. User taps the "Join" button to join the group.

   - Alternate Flows:
     - If the group is private, the user may need to wait for approval from the group administrator before joining .
3. **Create Channel:**
   - Actor: Regular User
   - Description: This use case describes the process of creating a new channel on Telegram.
   - Steps:

   3.1. User opens the Telegram application.
   3.2. User navigates to the "Channels" tab and selects the option to create a new channel.
   3.3. User provides a name and description for the channel, as well as selects the privacy settings (public or private).
   3.4. User taps the "Create" button to create the channel.

   - Alternate Flows:
     - If the user selects a private channel, they may need to invite members or approve requests to join.
4. **Send Message to Channel:**
   - Actor: Channel Owner

- Description: This use case describes the process of broadcasting a message to a channel's subscribers.
- Steps:

4.1. Channel owner opens the Telegram application.
4.2. Channel owner selects the channel they want to send a message to.
4.3. Channel owner composes the message content in the chat input field.
4.4. Channel owner taps the send button to broadcast the message to the channel's subscribers.

- Alternate Flows:
    - Channel owner may schedule the message to be sent at a later time or date.

❖ Traceability Matrix.

A traceability matrix is a tool used in software development to ensure that all requirements are addressed and properly implemented throughout the project lifecycle. It establishes a traceable link between various elements of the project, such as requirements, design components, test cases, and other artifacts. Here's how a traceability matrix can be structured for Telegram:

Requirement ID	Use Case ID	Functional Requirement	Non-Functional Requirement	Test Case ID	Design Component
REQ-001	UC-001	Send Message	Performance	TC-001	Messaging Module
REQ-002	UC-002	Join Group	Usability	TC-002	Group Module
REQ-003	UC-003	Create Channel	Security	TC-003	Channel Module
REQ-004	UC-004	Send Message to Channel	Compatibility	TC-004	Broadcast Module
REQ-005	UC-005	Customize Profile	Accessibility	TC-005	Profile Module
...	...	...	...	...	...

❖ Glossary

A glossary is a list of terms and their definitions used within a specific context or domain. In the case of Telegram, a glossary can help stakeholders, users, and developers understand the terminology and concepts associated with the messaging application. Here's a sample glossary for Telegram:

Telegram: A cloud-based messaging application that allows users to send text messages, multimedia files, and other content securely and privately.

Message: A unit of communication sent by a user to another user, group, or channel within the Telegram application.

Chat: A conversation between two or more users within the Telegram application, where messages are exchanged in real-time.

Group Chat: A chat room where multiple users can communicate with each other simultaneously within the Telegram application.

Channel: A broadcast platform within Telegram where users can subscribe to receive updates, news, and content from various sources.

Bot: An automated program or script that interacts with users within the Telegram application to perform various tasks and provide services.

End-to-End Encryption: A security feature in Telegram that encrypts messages between users to ensure privacy and prevent unauthorized access.

Secret Chat: A private conversation between two users in Telegram that offers enhanced security features such as end-to-end encryption, self-destructing messages, and screenshot alerts.

Notification: A message or alert sent by the Telegram application to inform users about new messages, mentions, replies, or other activities.

Profile: A user's personal account information and settings within the Telegram application, including their name, profile picture, status, and privacy settings.

Join: The action of adding a user to a group chat or channel within the Telegram application.

Admin: A user with administrative privileges in a group chat or channel, responsible for managing members, settings, and content moderation.

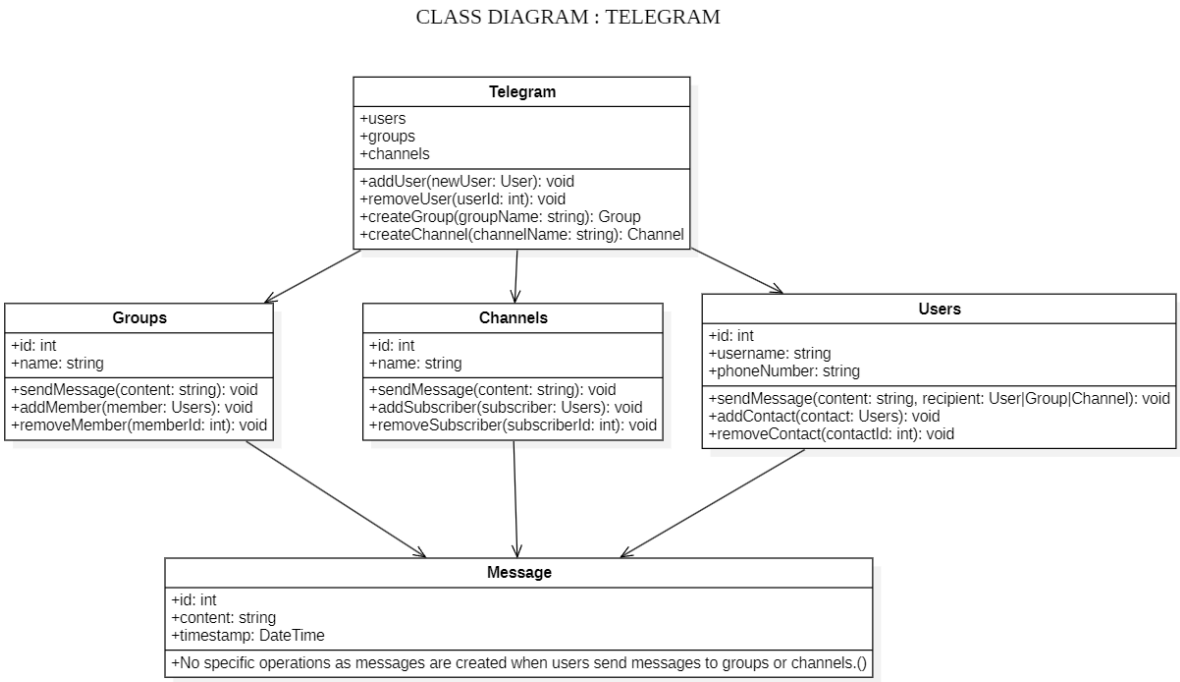
I**nvite Link**: A unique URL or code generated by Telegram that allows users to join a group chat or channel by clicking on the link.

Encryption Key: A cryptographic key used in end-to-end encryption to encode and decode messages between users in Telegram.

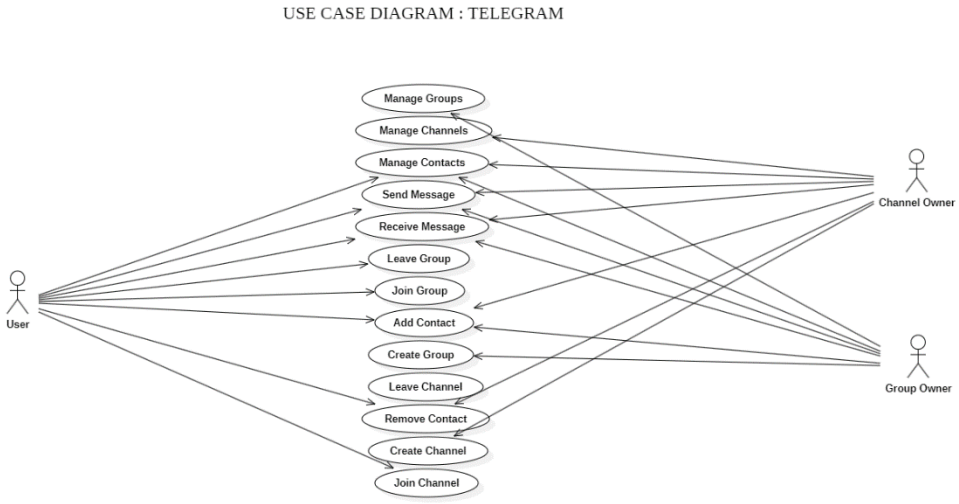
Backup: A copy of user data, including messages, media files, and settings, stored securely in the cloud or locally for recovery and restoration purposes.

Part II : Draw following UML diagram for above mentioned project

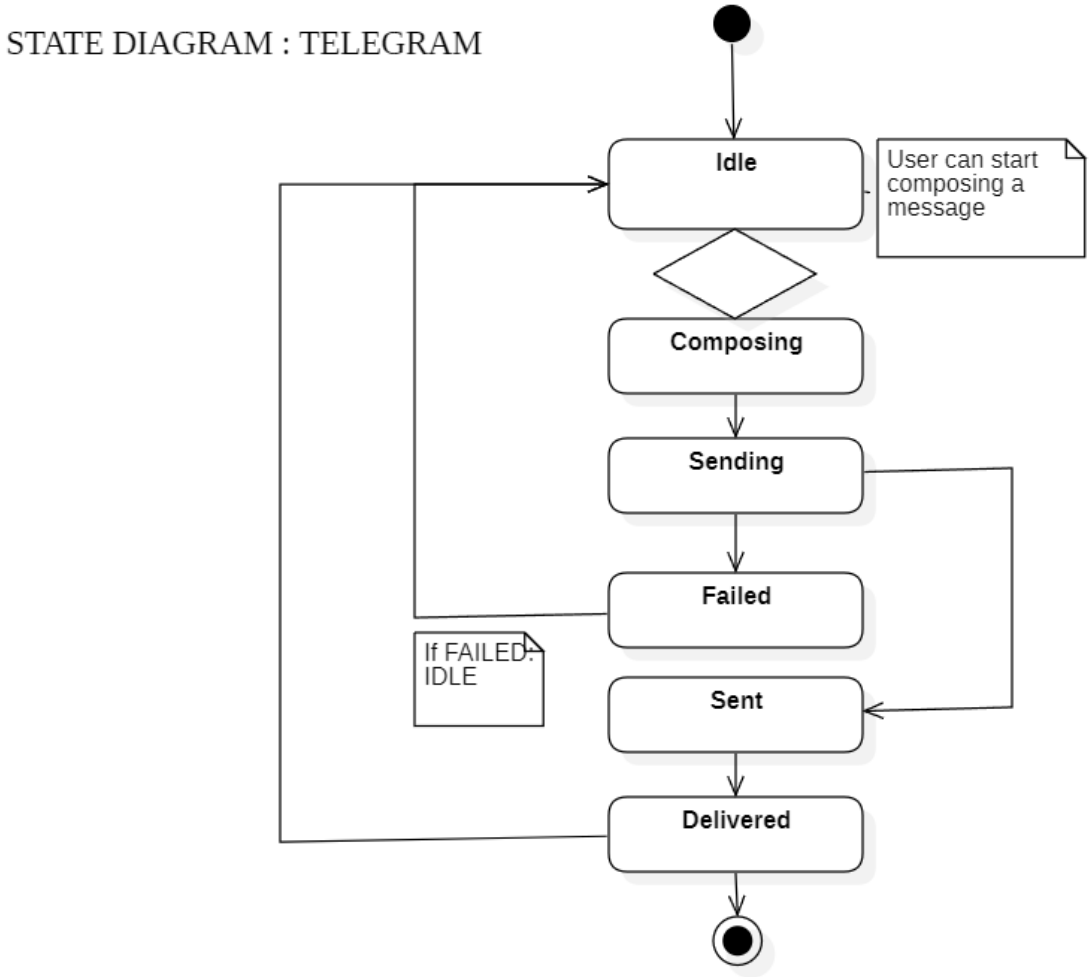
- **Class Diagram**



- **Use Case Diagram**



- **State Chart Diagram**



- **Sequence Diagram**