MESSAGE COMPILATION SYSTEM
SOFTWARRE ENGINEERING CONCEPTS DOCUMENTATION

Submitted by

JAYANEE POOBALARAYAN J	**220701102**
KAAVIYA G	**220701114**
KARUMURY NAGA MADHAVA NIKHIL	**220701120**
KEERTHIGA P	**220701125**
KEERTHIKA B	**220701126**
KIRUTHIGA M	**220701132**

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute
Thandalam
Chennai – 602105
2023-2024

ABSTRACT

The Message Compilation System is designed to streamline and enhance the process of aggregating, organizing, and presenting messages from multiple sources. In today's fast-paced digital environment, businesses and individuals often face challenges in managing communication efficiently, leading to wasted time, errors, and inconsistent outputs. This system addresses these issues by offering an automated, integrated solution that ensures accuracy, consistency, and ease of use.

Key features include a user-friendly interface for inputting and managing messages, advanced algorithms for efficient message compilation, and extensive customization options to meet specific user requirements. The system supports seamless integration with various communication platforms, enabling users to import messages effortlessly. It also provides real-time collaboration tools, allowing multiple users to work on projects simultaneously, thereby enhancing productivity.

Security and data privacy are paramount, with robust measures in place to protect user information and ensure compliance with industry standards. Additionally, an analytics module offers insights into system performance and user engagement, facilitating continuous improvement.

Overall, the Message Compilation System offers a comprehensive solution to the inefficiencies and inconsistencies of traditional message management methods, empowering users to communicate more effectively and professionally.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Introduction

1.2. Problem Statement

1.3. Solution

2. BUSINESS ANALYSIS

2.1 Business Architecture

2.2 How Different From Other Application

2.3 Process Of Working System

2.4 Business Problem

2.5 Solution

3. REQUIREMENTS

3.1. Epics And Features

3.2. User Story

3.3. Poker Planning Estimates

3.4. Non Functional Requirements

3.5 Functional Requirements

4. ARCHITECTURE DIAGRAM

4.1. Overview

4.2. Modules

4.3. Class Diagram

4.4. Sequence Diagram

5. TEST STRATEGY

5.1. Test Cases

5.2. Github Repository Structure

5.3. DevOps Architecture in Azure

6. DEPLOYMENT ARCHITECTURE

6.1 Deployment Diagram

6.2 Explanation

7. CONCLUSION

CHAPTER 1

INTRODUCTION

1.1.INTRODUCTION

A message compilation system is designed to collect, process, and organize messages from various sources into a coherent and structured format. This system typically involves several stages, including message retrieval, data parsing, content analysis, and output generation. The goal is to ensure that the messages are effectively compiled and presented, making it easier for users to access, understand, and utilize the information.

1.2. PROBLEM STATEMENT

In today's digital age, individuals and organizations are inundated with messages from multiple channels, including emails, social media, text messages, and other communication platforms. Managing and organizing this influx of information can be overwhelming and time-consuming. The core problem is the lack of a unified system that can automatically compile and categorize these messages in a meaningful way.

Specifically, the issues include:

- Fragmented Information: Messages are scattered across different platforms, leading to difficulties in accessing and synthesizing information.
- Time-Consuming Manual Compilation: Users spend significant amounts of time manually sorting and compiling messages, which is inefficient and prone to errors.
- Lack of Contextual Organization: Messages often lack contextual organization, making it hard to see the connections between related messages.
- Difficulty in Prioritizing Messages: Important messages can be easily overlooked due to the sheer volume of information.

1.3. SOLUTION

To address these issues, a message compilation system needs to:

- Integrate Multiple Sources: Seamlessly collect messages from various platforms and formats.
- Automate Processing: Use algorithms and artificial intelligence to automatically parse, analyze, and categorize messages.
- Provide Contextual Organization: Organize messages contextually to highlight connections and relevance.
- Enhance User Accessibility: Offer a user-friendly interface that allows users to quickly access and navigate compiled messages.
- Prioritize Effectively: Implement mechanisms to prioritize messages based on their importance and relevance to the user.
- The development of an efficient message compilation system can significantly reduce the burden of managing communication, leading to improved productivity and better information management.

CHAPTER 2
BUSINESS ANALYSIS

2.1. BUSINESS ARCHITECTURE

The business architecture of the message compilation system involves several key components that work together to provide a seamless and efficient experience for users. The main elements include:

- User Interface (UI): The frontend interface where users input their messages and view compiled outputs. This can be web-based, mobile, or desktop application.
- Backend Server: The server that processes user requests, manages data storage, and ensures smooth operation of the system. It handles the compilation logic and interacts with the database.
- Database: A robust database system that stores user data, message templates, compiled messages, and other relevant information securely.
- APIs: Application Programming Interfaces that allow integration with other systems, platforms, or services to enhance functionality, such as importing messages from different sources.
- Security Layer: Measures to ensure the confidentiality, integrity, and availability of data, including encryption, authentication, and authorization protocols.

2.2 HOW DIFFERENT FROM OTHER APPLICATION

The message compilation system stands out from other applications due to the following unique features:

- Customization and Flexibility: Offers extensive customization options for users to create personalized message templates and compile messages according to their specific needs.
- Integration Capabilities: Seamlessly integrates with various messaging platforms, social media, email services, and other communication tools to aggregate and compile messages from multiple sources.

- Advanced Compilation Algorithms: Utilizes sophisticated algorithms to compile messages efficiently, ensuring accuracy and coherence in the final output.
- User-friendly Interface: Designed with an intuitive interface that simplifies the message compilation process, making it accessible even for users with limited technical knowledge.
- Real-time Collaboration: Supports real-time collaboration, allowing multiple users to work on message compilation projects simultaneously, enhancing productivity and teamwork.
- Security and Compliance: Adheres to industry standards for data security and privacy, ensuring that user information is protected and compliant with relevant regulations.

2.3. PROCESS OF WORKING SYSTEM

The process of the working system involves several stages:

- User Registration and Authentication: Users sign up and log in to the system using secure authentication methods.
- Message Input: Users input their messages or import them from various sources using integrated APIs.
- Template Selection and Customization: Users select from pre-designed templates or create their own, customizing them to fit their requirements.
- Message Compilation: The system compiles the input messages based on the selected templates and user specifications using advanced algorithms.
- Review and Edit: Users review the compiled messages and make any necessary edits to ensure accuracy and coherence.
- Output Generation: The final compiled message is generated and can be exported, shared, or sent through integrated communication channels.

2.4. BUSINESS PROBLEM

The primary business problem addressed by the message compilation system is the inefficiency and inconsistency in compiling messages from multiple sources. Businesses often face challenges in aggregating, organizing, and presenting information cohesively, leading to wasted time, errors, and miscommunication.

Key issues include:

- Time-Consuming Processes: Manually compiling messages from various platforms is labor-intensive and prone to delays.
- Inconsistent Formatting: Different sources have different formatting styles, making it difficult to maintain a consistent output.
- Human Error: Manual processes increase the risk of errors, which can lead to miscommunication and misunderstandings.
- Lack of Integration: Without proper integration, aggregating messages from multiple platforms can be cumbersome and inefficient.

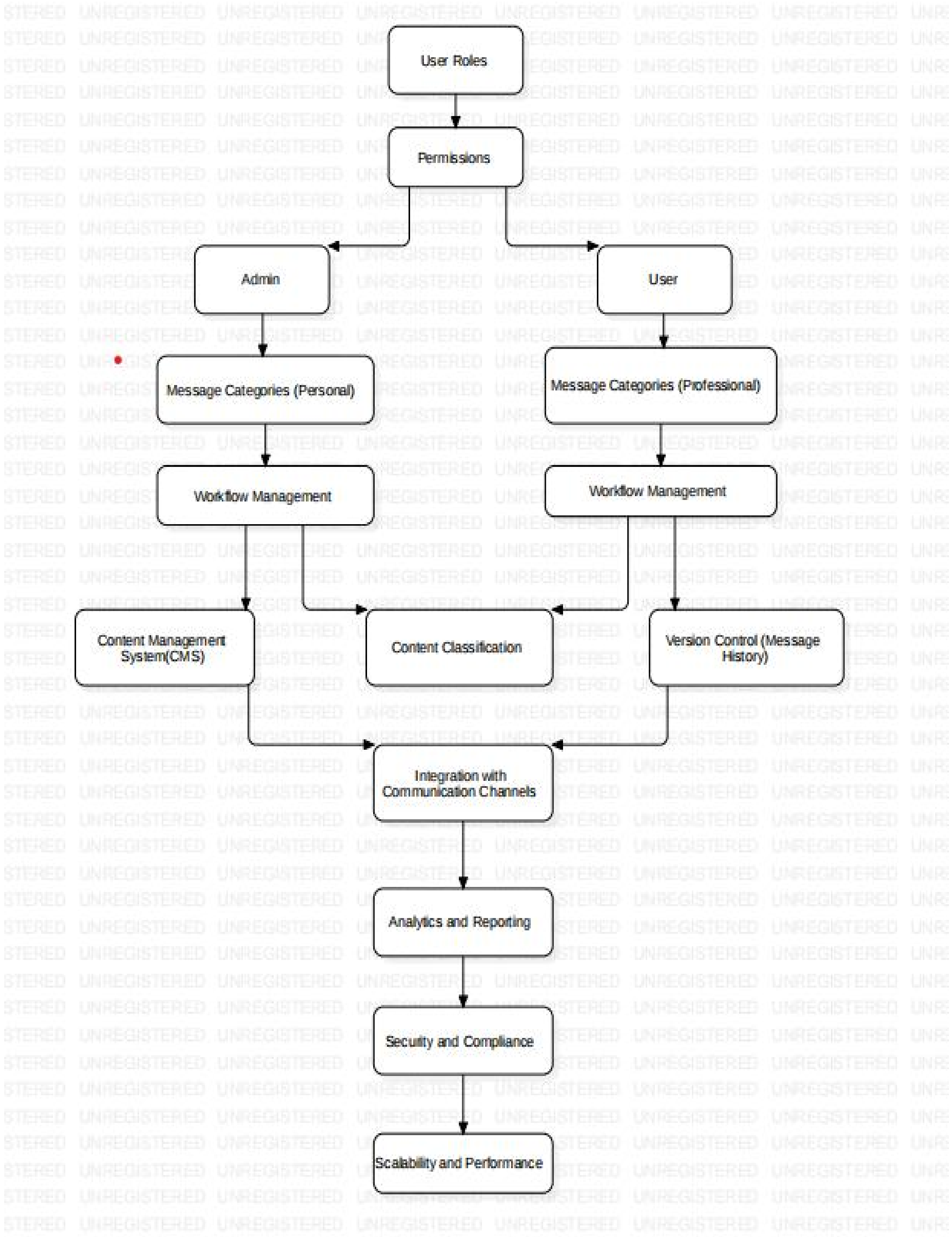
2.5. SOLUTION

The message compilation system provides a comprehensive solution to these problems by:

- Automation: Automating the message compilation process significantly reduces the time and effort required, enhancing productivity.
- Consistency: Ensuring consistent formatting and presentation of compiled messages, improving readability and professionalism.
- Error Reduction: Minimizing the risk of human error through automated processes and advanced algorithms.
- Integration: Offering seamless integration with various platforms and services, making it easy to import and compile messages from multiple sources.
- Customization: Allowing users to create and use customized templates, ensuring that the compiled messages meet their specific needs and preferences.

- Collaboration: Enabling real-time collaboration, allowing teams to work together efficiently on message compilation projects.
- Security: Implementing robust security measures to protect user data and ensure compliance with industry standards and regulations.

BUSINESS ARCHITECTURE DIAGRAM :


User Roles
Permissions
Admin
User
Message Categories (Personal)
Message Categories (Professional)
Workflow Management
Workflow Management
Content Management System(CMS)
Content Classification
Version Control (Message History)
Integration with Communication Channels
Analytics and Reporting
Security and Compliance
Scalability and Performance

CHAPTER 3
REQUIREMENTS
3.1. EPICS AND FEATURES

Epic : Unified Messaging Hub

Description: Seamlessly integrate Instagram and SMS messages into one platform, allowing users to efficiently manage their communications.

Features:

- Message Segregation: Automatically categorize messages as personal or professional based on user-defined criteria.
- Authorization Prompt: Prompt users to authorize their messaging accounts to ensure accurate segregation.
- Category Organization: Sort messages into customizable categories such as finance, travel, and technology for easy access.

Epic : Smart Message Classifier

Description: Utilize advanced algorithms to classify messages from Instagram and SMS into relevant categories, optimizing user experience.

Features:

- Machine Learning Integration: Employ machine learning models to analyze message content and categorize them accordingly.
- Contextual Understanding: Recognize contextual clues within messages to accurately determine their relevance to personal or professional contexts.
- Real-time Updates: Continuously update message classifications based on user interactions and feedback for improved accuracy.

Epic : Personalized Communication Platform

Description: Provide users with a personalized messaging experience by intelligently organizing their Instagram and SMS conversations.

Features:

- User Preference Settings: Allow users to set preferences for message categorization and organization according to their individual needs.

- Conversation Threads: Group related messages together to facilitate easy tracking and management of ongoing conversations.
- Customizable Filters: Enable users to create custom filters to further refine message segregation based on specific keywords or sender profiles.

Epic : Effortless Message Management

Description: Streamline message handling tasks by centralizing Instagram and SMS communications within a single, user-friendly interface.

Features:

- Unified Inbox: Aggregate messages from multiple sources into a single inbox for convenient access and management.
- Bulk Action Tools: Enable users to perform bulk actions such as archiving, deleting, or marking messages as read with ease.
- Notification Controls: Customize notification settings to receive alerts for important messages while minimizing distractions from non-urgent ones.

Epic : Intelligent Message Prioritization

Description: Prioritize important messages and streamline communication workflows by implementing intelligent prioritization algorithms.

Features:

- Priority Inbox: Automatically identify and highlight high-priority messages based on user-defined criteria such as sender importance or message urgency.
- Urgent Message Alerts: Notify users of critical messages via push notifications or SMS alerts to ensure timely responses.
- Adaptive Learning: Continuously refine message prioritization algorithms based on user feedback and interaction patterns for enhanced accuracy.

Epic : Secure Communication Hub

Description: Safeguard user privacy and data integrity by implementing robust security measures across the messaging platform.

Features:

- End-to-End Encryption: Encrypt message content to prevent unauthorized access or interception during transmission.

- Two-Factor Authentication: Enhance account security by requiring users to authenticate their identity using two-factor authentication methods.
- Data Privacy Controls: Empower users with granular control over their data by implementing privacy settings and consent mechanisms for message handling and storage.

3.2. USER STORY

User Story : User Registration

As a new user, I want to register for the Message Compilation System so that I can access its features.

Acceptance Criteria :

- User can register using email or social media accounts (e.g., Google, Facebook).
- User must provide required information such as name, email, and password.
- After successful registration, the user receives a confirmation email or notification.
- User can choose to skip registration and explore the app with limited functionality.

User Story : User Login

As a registered user, I want to log in to the Message Compilation System to access my account.

Acceptance Criteria :

- User can log in using email and password or social media accounts.
- User receives an error message if login credentials are incorrect.
- User can reset their password if forgotten.
- Upon successful login, the user is directed to the app's home screen.

User Story : Connect Instagram Account

As a user, I want to connect my Instagram account to the Message Compilation System to integrate Instagram messages.

Acceptance Criteria :

- User can navigate to the settings or profile section to connect their Instagram account.
- User must authorize the app to access their Instagram messages.
- Upon successful connection, the app displays a confirmation message.
- App fetches Instagram messages and categorizes them based on predefined criteria.

User Story : Connect SMS Messages

As a user, I want to connect my SMS messages to the Message Compilation System to integrate SMS messages.

Acceptance Criteria :

- User can grant permissions to the app to access SMS messages.
- App fetches SMS messages and categorizes them based on predefined criteria.
- User has the option to enable/disable SMS message integration.
- App displays a confirmation message upon successful connection.

User Story : Categorize Messages

As a user, I want the Message Compilation System to categorize my messages into personal and professional based on predefined criteria.

Acceptance Criteria :

- App analyzes message content from both Instagram and SMS.
- Messages are categorized into predefined categories such as finance, travel, technology, etc.
- App distinguishes between personal and professional messages based on user settings and context.
- User can manually override categorization if necessary.

User Story : View Segregated Messages

As a user, I want to view my messages segregated into different categories for easy access.

Acceptance Criteria :

- App displays segregated messages in a user-friendly interface.
- User can navigate between categories (e.g., finance, travel) to view messages.
- Messages are displayed chronologically within each category.
- User can mark messages as read or unread, and archive or delete them as needed.

3.3. POKER PLANNING ESTIMATES

1. User Login :

- **Description**: Implementing user login functionality, including registration, authentication, and session management.
- **Effort**: 3 points
- **Reason**: This is a standard feature in most applications. It involves setting up user accounts, password hashing, and managing sessions. Libraries and frameworks make this task easier, but attention to security is crucial

2. User Authorization :

- **Description:** Implementing role-based access control (RBAC) and ensuring that users have appropriate permissions.
- **Effort**: 5 points
- **Reason**: RBAC involves setting up different roles and permissions, which can get complex depending on the granularity required. Testing and validating access control adds to the complexity.

3. Category Management :

- **Description**: Allowing users to create, edit, and delete categories for message segregation.

- **Effort**: 3 points

- **Reason**: This requires creating a user interface for category management, along with backend support to store and retrieve categories. It is relatively straightforward but requires a well-designed user interface and efficient backend handling.

4. Message Segregation

- **Description**: Implementing the logic to segregate messages into personal and professional, and further categorizing them based on content.

- **Effort**: 8 points

- **Reason**: This is the core functionality of the application and is quite complex. It involves integrating with external APIs (Instagram, SMS), processing message content, and using natural language processing (NLP) to determine the category. This also involves a feedback loop to improve accuracy over time.

Summary :

- **User Login:** 3 points
- **User Authorization:** 5 points
- **Category Management:** 3 points
- **Message Segregation:** 8 points

3.4. NON-FUNCTIONAL REQUIREMENTS

Non functional requirements :

Scalability:

- The system should be able to handle a large volume of messages from various sources (e.g., Instagram, SMS) without degradation in performance.

Security:

- Implement secure authentication mechanisms to ensure that only authorized users can access the system and their messages. Encrypt sensitive data, such as personal information and message content, to prevent unauthorized access.

Reliability:

- Ensure high availability and reliability of the system to prevent data loss or downtime. Implement backup and recovery mechanisms to recover data in case of system failure.

Performance:

- The system should have low latency and fast response times when processing and categorizing messages. Optimize algorithms and data structures to improve performance.

Scalability:

- The system should be able to scale horizontally to accommodate a growing user base and increasing message volume. Implement load balancing and distributed processing techniques to distribute workload efficiently.

Compatibility:

- Ensure compatibility with various social media platforms and SMS protocols to support a wide range of users and devices.

User Experience:

- Design a user-friendly interface that allows users to easily navigate and interact with the system. Provide clear instructions and feedback to guide users through the message categorization process.

Privacy:

- Respect user privacy and comply with data protection regulations. Implement measures to anonymize or pseudonymize personal data when necessary, and provide users with control over their data.

3.5. FUNCTIONAL REQUIREMENTS

Functional Requirements :

User Authentication:

- Users must be able to securely log in to the app using their Instagram and SMS accounts to access their messages.

Message Integration:

- The app should seamlessly integrate with Instagram and SMS platforms to retrieve and organize messages.

Message Categorization:

- Implement a system to automatically categorize messages into predefined categories such as finance, travel, technology, etc., based on content analysis.

User Profiling:

- Develop a user profiling mechanism to determine whether a contact is categorized as personal or professional based on interaction history and user-defined preferences.

Authorization:

- Users should have the ability to authorize the categorization of their contacts as personal or professional, allowing for customization and fine-tuning of the categorization process.

Real-time Message Updates:

- Ensure that the app provides real-time updates for new messages received on both Instagram and SMS platforms.

Customization:

- Allow users to customize category labels and criteria for message categorization to suit their preferences and needs.

Notification Management:

- Implement notification settings that enable users to control how they are alerted about new messages, including sound, vibration, and in-app notifications.

Search and Filter:

- Provide robust search and filtering functionalities to allow users to quickly locate specific messages based on various criteria such as sender, category, and keywords.

CHAPTER 4
ARCHITECTURE DIAGRAM
4.1. OVERVIEW

- The Message Compilation System is designed to address the challenge of managing personal and professional messages from multiple sources. By integrating SMS and Instagram messages into a single platform, the system helps users stay organized and focused. The authentication module ensures secure access to message data, while the data integration module fetches and syncs messages from external APIs.
- The core of the system is the message categorization module, which uses NLP to automatically sort messages into personal and professional categories. Users receive real-time notifications about new messages through the notification module, ensuring they are always up-to-date. The user interface module provides a seamless experience across web and mobile platforms, allowing users to view, search, and manage their messages efficiently.
- Data storage is handled securely, ensuring that all user data and messages are protected. The security module further ensures that data privacy and integrity are maintained, making the Message Compilation System a reliable and secure solution for managing communications. Overall, the system enhances productivity, reduces stress, and provides a comprehensive tool for managing both personal and professional messages.

4.2. MODULES

The Message Compilation System is designed to integrate and categorize messages from SMS and Instagram, providing users with a unified platform to manage their communications efficiently. Below is an overview of the key modules and their functionalities:

1. Authentication Module

- ❖ Overview: Handles the secure authentication of user accounts for both SMS and Instagram.
- ❖ Functions:
  - Verify phone numbers for SMS integration.
  - Authenticate Instagram accounts using OAuth for secure access to direct messages. Manage user sessions and token refresh.

2. Data Integration Module :

- ❖ Overview: Retrieves messages from SMS and Instagram APIs and stores them in the system.
- ❖ Functions:
  - o Connect to SMS and Instagram APIs to fetch messages.
  - o Periodically sync new messages from both sources.
  - o Handle API rate limits and errors gracefully.

3. Message Categorization Module :

- ❖ Overview: Uses Natural Language Processing (NLP) to analyze and categorize messages into personal and professional.
- ❖ Functions:
  - o Analyze message content to determine context and intent.
  - o Automatically categorize messages into personal or professional.
  - o Allow manual overrides and custom labeling by users.

4. Notification Module :

- ❖ Overview: Sends real-time notifications to users about new messages.
- ❖ Functions:
  - o Notify users of new messages as they arrive.
  - o Provide notifications for both web and mobile applications.
  - o Manage notification settings and preferences.

5. User Interface Module :

- ❖ Overview: Provides a responsive and intuitive interface for users to view and manage their messages.
- ❖ Functions:
  - o Display categorized messages in a clear and organized manner.
  - o Offer search functionality to find specific messages quickly.
  - o Allow users to manually categorize and label messages.
  - o Provide access through both web browsers and mobile applications.

6. Data Storage Module :

- ❖ Overview: Manages the storage of user data, message details, and categorization results.
- ❖ Functions:
  - o Store user information and authentication tokens securely.
  - o Save message content and metadata.
  - o Maintain categorization results and user labels.

7. Security Module :

- ❖ Overview: Ensures the security and privacy of user data throughout the system.
- ❖ Functions:
  - o Encrypt data at rest and in transit.
  - o Implement access controls and permissions.
  - o Monitor for security breaches and vulnerabilities.

4.3. CLASS DIAGRAM

1. **User** :

Attributes:

- `userId: string` - A unique identifier for each user.

- `username: string` - The username of the user.

- `password: string` - The password of the user, used for authentication.

Operations:

- `authenticateUser(username, password): boolean` - Verifies if the provided username and password match an existing user in the system.

- `authorizeUser(user): boolean` - Checks if the user has the necessary permissions to perform certain actions.

2. **Personal Message & Professional Message**

Attributes:

- `messageId: string` - A unique identifier for each message.

- `sender: User` - The user who sent the message.

- `content: string` - The text content of the message.

- `timestamp: datetime` - The date and time when the message was sent.

- `category: string` - The category of the message, such as 'personal' or 'professional'.

Operations:

- `setIsPersonal(isPersonal: boolean): void` - Sets the personal status of the user.

- `sendPersonalMessage(isPersonal: boolean): void` - Sends a personal message. The isPersonal parameter indicates whether the message is personal.

- `receivePersonalMessage(): void` - Receives a personal message.

- `setIsProfessional(isProfessional: boolean): void` - Sets the professional status of the user.

- `sendProfessionalMessage(isProfessional: boolean): void` - Sends a professional message. The isProfessional parameter indicates whether the message is professional.

- `receiveProfessionalMessage(): void` - Receives a professional message.

3. **UserManager :**

Attributes :

- `instagramId : string` - The user's Instagram ID.

- `name: string` - The user's name.

- `isPersonal: boolean` - Indicates if the user is a personal user.

- `isProfessional: boolean` - Indicates if the user is a professional user.

- `age: int` - The user's age.

- `phone_number: string` - The user's phone number.

Operations:

- `authenticateUser(username, password): boolean` - Verifies if the provided username and password match an existing user in the system.

- `authorizeUser(user): boolean` - Checks if the user has the necessary permissions to perform certain actions.

4. **Message Receiver** :

Attributes:

- accessToken: string - The access token used to authenticate and receive messages from services like Instagram.

Operations:

- `receiveInstagramMessage(accessToken): void` - Receives messages from Instagram using an access token for authentication.

- `receiveSMSMessage(): void` - Receives SMS messages from the device or a service.

5. **Message Parser** :

Attributes:

- `content: string` - The raw content of the message that needs to be parsed.

Operations:

- \`parseMessage(content): Message\` - Converts raw message content into a structured \`Message\` object.

6. **Message Categorizer** :

Attributes:

- \`message: Message\` - The message object that needs to be categorized.

Operations:

- \`categorizeMessage(message): string\` - Determines the category of a message (e.g., 'personal', 'professional') based on its content.

7. Database Manager :

Attributes:

-\`message: Message\` - The message object that needs to be stored or retrieved from the database.

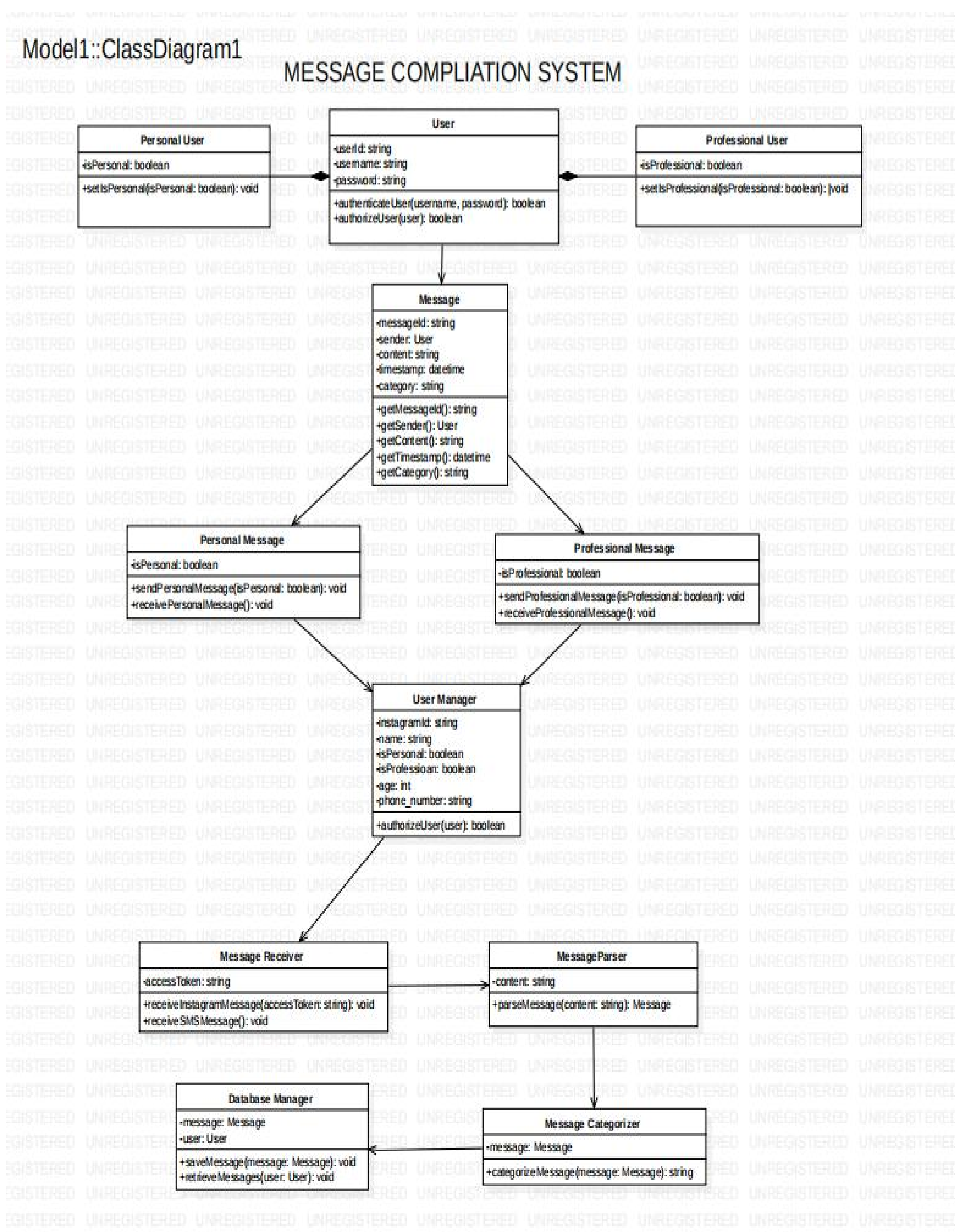
- \`user: User\` - The user associated with the messages being retrieved.

Operations:

- \`saveMessage(message): void\` - Stores a message in the database.

- \`retrieveMessages(user): List<Message>\` - Retrieves all messages associated with a specific user from the database.

CLASS DIAGRAM



4.3. SEQUENCE DIAGRAM

1. **User Authentication:**

- The sequence begins with the User sending an authenticateUser message to the UserManager with a username and password.
- The UserManager processes the authentication and returns a boolean indicating whether the authentication was successful or not.

2. **Message Reception:**

- Once authenticated, the UserManager sends a message to the MessageReceiver to indicate that a message has been received and needs processing.
- The MessageReceiver calls receiveInstagramMessage(accessToken) to receive a message from Instagram using an access token.
- Additionally, the MessageReceiver calls receiveSMSMessage() to receive any SMS messages.

3. **Message Parsing:**

- The MessageReceiver sends the message content to the MessageParser by calling parseMessage(content).
- The MessageParser processes the raw content and returns a parsedMessage object.

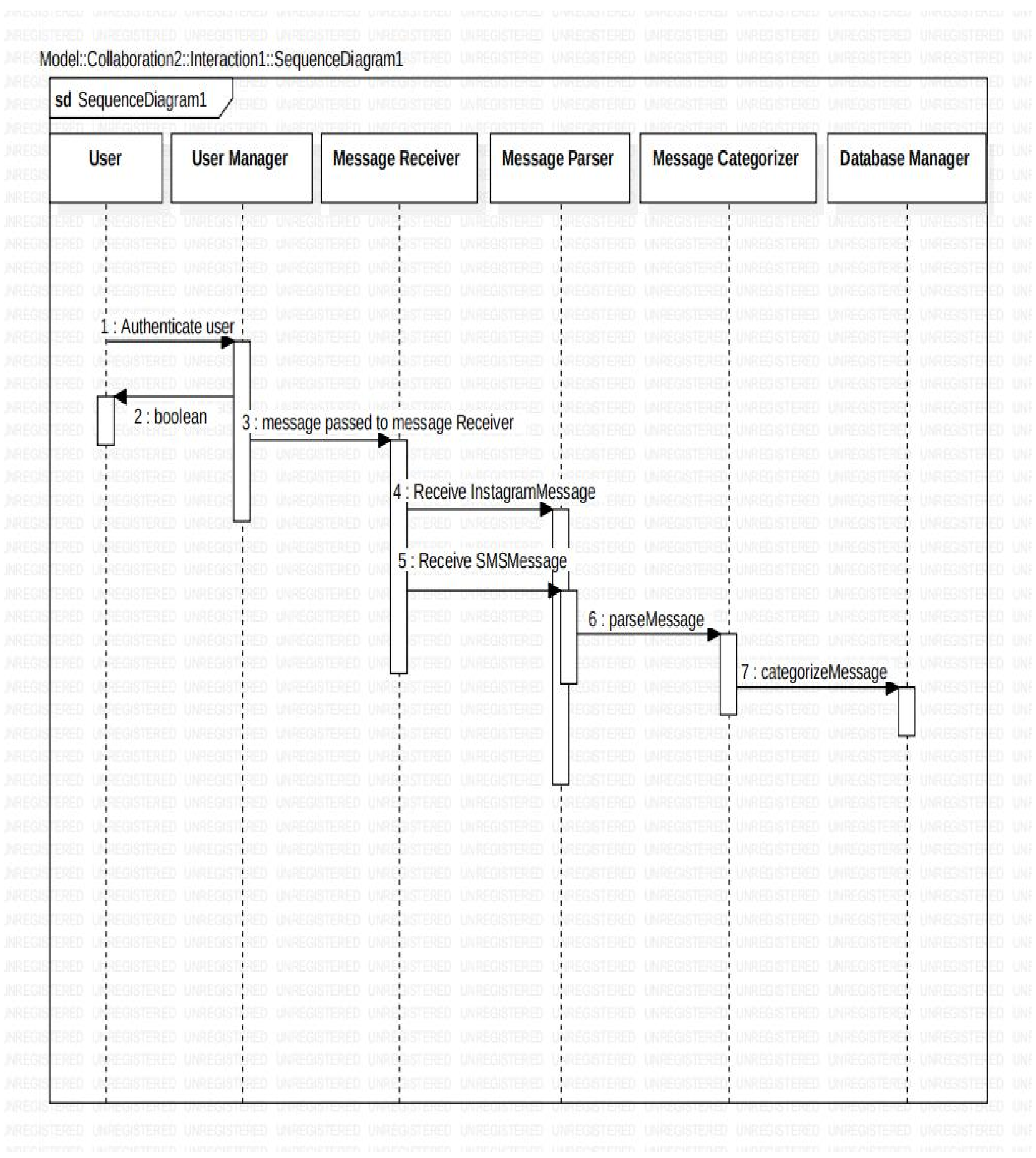
4. **Message Categorization:**

- The MessageReceiver sends the parsedMessage to the MessageCategorizer by calling categorizeMessage(parsedMessage).
- The MessageCategorizer determines the category of the message and returns this category information to the MessageReceiver.

5. **Message Storage:**

- Finally, the MessageReceiver sends the parsedMessage and its category to the DatabaseManager by calling saveMessage(parsedMessage, category).
- The DatabaseManager stores the message and its category in the database and returns a void response to indicate that the operation was completed successfully.

SEQUENCE DIAGRAM :



CHAPTER 5

TEST STRATEGY

1.1 TEST CASES

Introduction:

- The Test Strategy document outlines the approach, resources, and schedule for the testing activities of the Message Compilation System
- The goal is to ensure that the system meets its requirements and functions correctly under all specified conditions.

Objectives:

- Verify that the system meets functional and non-functional requirements.
- Identify and resolve defects.
- Ensure the system's usability, performance, security, and compatibility.

Scope:

- Functional Testing
- Performance Testing
- Security Testing

Test Approach:

- Manual Testing: Focus on exploratory testing, usability, and edge cases.
- Automated Testing: Focus on regression testing and repetitive tasks.

Test Environment:

- Operating Systems: Windows, macOS, Linux
- Browsers: Chrome, Firefox, Safari, Edge
- Devices: Desktop, Tablet, Mobile
- Tools: Selenium, JIRA, JMeter

Test Plan :

Test Cases for User Stories :

User Story 1: Create a Message

Description: As a user, I want to create a message so that I can compile it later.

Happy Path

1. Test Case ID: TC_001

Description: Verify that a user can create a message successfully.

Preconditions: User is logged in.

Steps:

Navigate to the message creation page.

Enter valid message details.

Click "Create".

Expected Result: Message is created and displayed in the list.

Error Scenario

2. **Test Case ID**: TC_002

Description: Verify that an error is shown when mandatory fields are empty.

Preconditions: User is logged in.

Steps:

Navigate to the message creation page.

Leave mandatory fields empty.

Click "Create".

Expected Result: Error message indicating mandatory fields are required.

User Story 2: Edit a Message

Description: As a user, I want to edit an existing message to update its content.

Happy Path

3. **Test Case ID**: TC_003

Description: Verify that a user can edit a message successfully.

Preconditions: User is logged in, message exists.

Steps:

Navigate to the message list.

Click on the "Edit" button of a message.

Update message details.

Click "Save".

Expected Result: Message is updated with new details.

Error Scenario

4. **Test Case ID**: TC_004

Description: Verify that an error is shown when invalid data is entered.

Preconditions: User is logged in, message exists.

Steps:

Navigate to the message list.

Click on the "Edit" button of a message.

Enter invalid data (e.g., excessively long text).

Click "Save".

Expected Result: Error message indicating invalid data.

User Story 3: Delete a Message

Description: As a user, I want to delete a message so that it is no longer available.

Happy Path

5. **Test Case ID**: TC_005

Description: Verify that a user can delete a message successfully.

Preconditions: User is logged in, message exists.

Steps:

Navigate to the message list.

Click on the "Delete" button of a message.

Confirm deletion.

Expected Result: Message is deleted and no longer displayed.

Error Scenario

6. **Test Case ID**: TC_006

Description: Verify that an error is shown when trying to delete a non-existing message.

Preconditions: User is logged in.

Steps:

Attempt to delete a non-existing message (simulate the scenario).

Expected Result: Error message indicating the message does not exist.

User Story 4: Compile Messages

Description: As a user, I want to compile multiple messages into one document.

Happy Path

7. **Test Case ID**: TC_007

Description: Verify that a user can compile multiple messages successfully.

Preconditions: User is logged in, multiple messages exist.

Steps:

Navigate to the message list.

User Story 5: As a user, I want to search for messages by title.

Happy Path:

7. **Test Case ID**: TC_008

Description: Verify that a user can search for messages by title.

Steps:

Navigate to the message list page.

Enter a search term in the search bar.

Click on the "Search" button.

Verify that messages matching the search term are displayed.

Expected Result: Messages matching the search term are displayed.

5.2. GITHUB REPOSITORY STRUCTURE

GitHub Repository Structure:

```
MessageCompilationSystem/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── example/
│   │   │           └── messagecompilation/
│   │   │               ├── controllers/
│   │   │               ├── models/
│   │   │               ├── repositories/
│   │   │               ├── services/
│   │   │               └── utils/
│   └── test/
│       ├── java/
│       │   └── com/
│       │       └── example/
│       │           └── messagecompilation/
│       │               ├── controllers/
│       │               ├── models/
│       │               ├── repositories/
│       │               ├── services/
│       │               └── utils/
├── resources/
```

```
│   ├── static/
│   ├── templates/
│   └── application.properties
├── scripts/
│   ├── init.sql
│   ├── test_data.sql
│   └── migration/
├── .gitignore
├── README.md
├── pom.xml (or build.gradle)
└── docs/
    ├── design/
    ├── requirements/
    └── test/
        └── test_plan.md
        └── test_cases.md
```

Naming Conventions:

Packages and Folders:

- Follow standard naming conventions such as lowercase letters, and meaningful names.

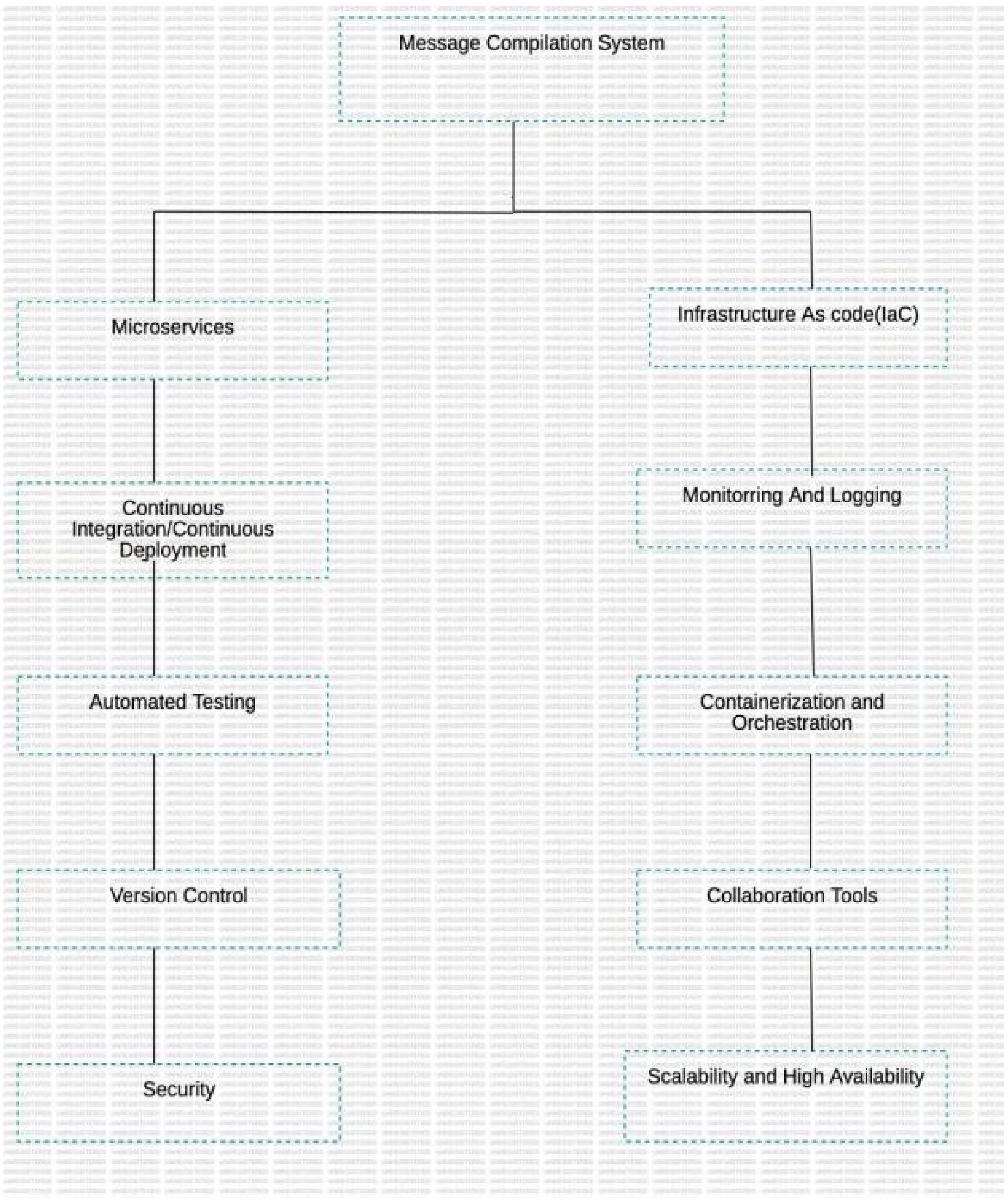
Classes:

- Use CamelCase for class names (e.g., MessageController, MessageService).

Files:

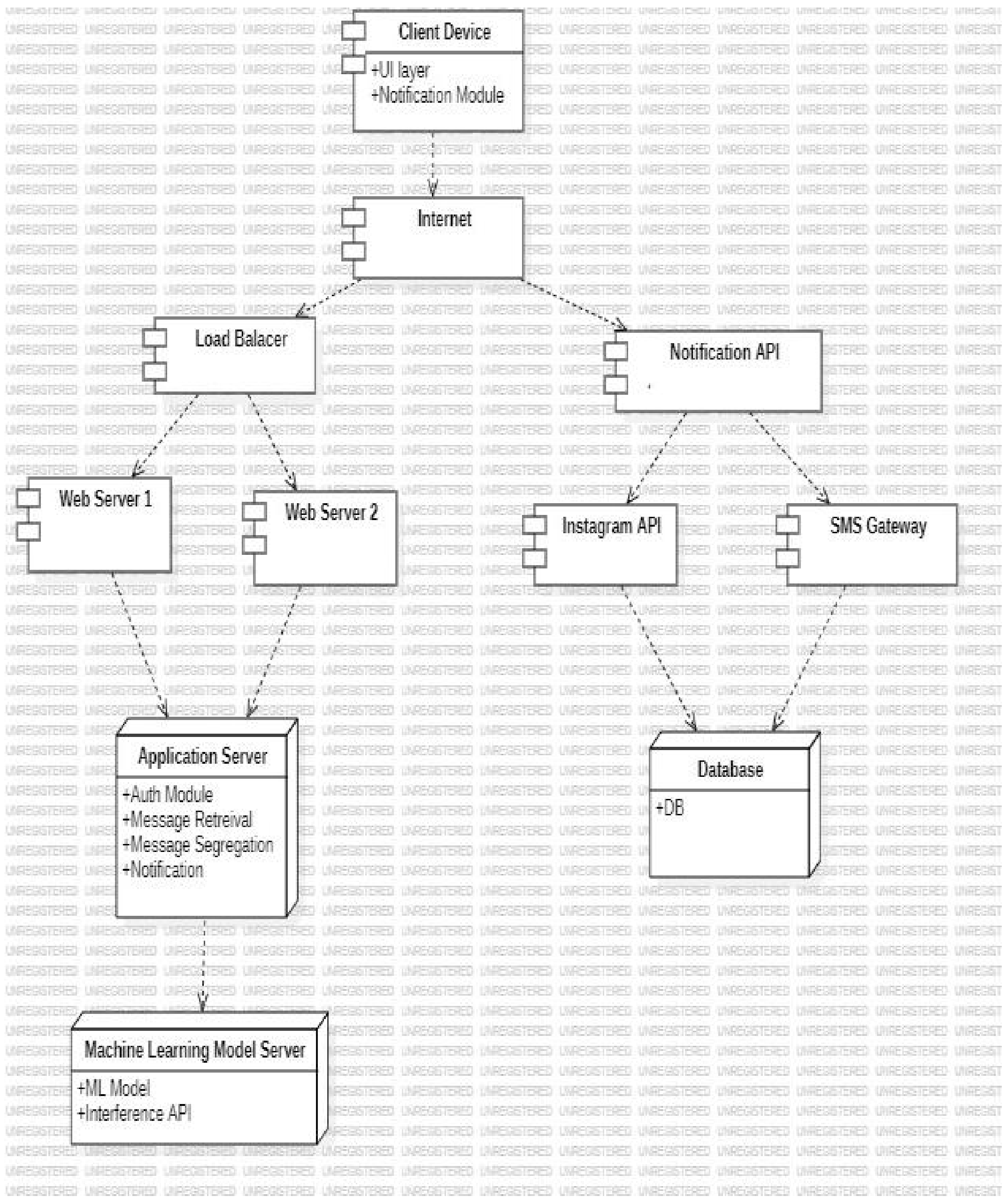
- Use descriptive names reflecting the purpose (e.g., init.sql for database initialization, application.properties for configuration).

5.3. DEVOPS ARCHITECTURE IN AZURE


Message Compilation System
Microservices
Continuous Integration/Continuous Deployment
Automated Testing
Version Control
Security
Infrastructure As code(IaC)
Monitorring And Logging
Containerization and Orchestration
Collaboration Tools
Scalability and High Availability

CHAPTER 6 : DEPLOYMENT ARCHITECTURE

6.1 DEPLOYMENT DIAGRAM



6.2 EXPLANATION

1. **Client Device:**
   - This is the user's device (mobile or web) where the user interacts with the application.

2. **Frontend Application:**
   - This component represents the UI/UX part of the system. It could be a mobile app or a web application where users can view and interact with their segregated messages.
   - It communicates with the Backend API Server to fetch and display messages.

3. **Backend API Server:**
   - The core of the system's logic. It handles requests from the frontend, interacts with the Social Media Integration Service, SMS Gateway, and Message Classification Engine.
   - It manages user authentication, data processing, and communication between different components.

4. **Notification Server:**
   - This server is responsible for sending notifications to the users when new messages are received. It can use push notifications for mobile devices or web notifications for web applications.

5. **Social Media Integration Service:**
   - This service handles the integration with social media platforms like Instagram. It uses the Instagram API to fetch messages.
   - It ensures that the messages from Instagram are collected and passed on to the Backend API Server.

6. **Instagram API:**
   - The external API provided by Instagram that allows access to user messages. The Social Media Integration Service interacts with this API to retrieve messages.

7. SMS Gateway:

- This component is responsible for receiving and sending SMS messages. It integrates with the Backend API Server to pass on received SMS messages for processing.

8. Message Classification Engine:

- The heart of the categorization system. This engine uses machine learning or rule-based algorithms to classify messages into categories such as finance, travel, tech, etc.
- It also segregates messages into personal and professional categories based on predefined criteria.

9. Message Database:.

- A storage system (like a relational database or NoSQL database) where all messages are stored after classification.
- It supports queries to fetch messages based on user requests, such as filtering by category or type.

DETAILED WORKFLOW :

1. Message Retrieval:

- The Social Media Integration Service retrieves messages from Instagram using the Instagram API.
- The SMS Gateway receives SMS messages and forwards them to the Backend API Server.

2. Message Classification:

- The Backend API Server sends the collected messages to the Message Classification Engine.
- The Message Classification Engine analyzes the content of the messages to determine their category (e.g., finance, travel) and type (personal or professional).

3. Storage and Notification:

- Classified messages are stored in the Message Database with metadata indicating their category and type.
- The Notification Server sends notifications to the user's device indicating that a new message has been received and categorized.

4. User Interaction:

- The user interacts with the Frontend Application to view their messages, which are fetched from the Backend API Server.
- The user can filter and browse messages based on categories and types (personal/professional).

CHAPTER 7
CONCLUSION

The Message Compilation System is an innovative solution designed to streamline and organize communications across different platforms by categorizing messages into personal and professional segments. The primary objective of this system is to enhance productivity and ensure that users can easily navigate through their communications based on context and relevance.

Key Objectives:

Integration of Messaging Platforms:

- Integrate Instagram and SMS into the app to read and fetch messages.

User Authentication and Authorization:

- Authorize users to differentiate between personal and professional contexts.

Data Segregation and Organization:

- Design an intuitive interface that displays categorized messages for easy access and management.

Privacy and Security:

- Ensure the privacy and security of user data during integration and storage.

By achieving these objectives, the Message Compilation System will offer an efficient solution for managing and organizing messages from different sources, enhancing productivity and communication management for users.