International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 12 | Dec 2021 www.irjet.net p-ISSN: 2395-0072

An object-Oriented Approach towards Building WhatsApp Inspired Platform-Independent Chat Application (WOJ) through Java without using Database Management System

Vineesh Cutting[1], Syed Aeliya Mahdi Taqvi[2]

[1]*Assistant Professor, Department of Computer Science & IT, SHUATS, India.*
[2]*Developer, Amiti Software Technology Pvt. Ltd., India.*

Abstract – *Chat application is a program that utilizes internet to communicate directly among other internet users who are online or who are using the internet. It allow users to communicate over great distance. Therefore, chat application needs to be real-time and platform-independent to be used by many users. The development of proposed chat application (WOJ) begins with the collection of relevant data. All the modules of application are defined using Java following pure object-oriented concepts thereby making it highly independent of other technologies. Serialization has been used to save the state of objects. WOJ provides very neat and clean chatting interface through which one client/user can interact with others in a very simple way with very less complexity. WOJ supports file sharing (for: jpeg, png, mp4, pdf, etc), which makes this application very handy for sharing multi-media content while chatting.*

Key Words: WOJ, Object-Oriented Programming, Java, Chat Application, Client-Server, No-Database, Serialisation.

1. INTRODUCTION

Communication is a means for people to exchange information. It has started since the beginning of human creation. Interestingly enough, telephonic communications stand out as the fastest growing technology, from fixed line to wireless mobile, from voice call to data driven digital calls.

The emergence of digital computer network and telecommunication technologies bears the same objective that is to allow people to communicate. All this while, much effort has been drawn towards consolidating the devices into one and therefore combining the services. Today digital chatting is one of the best ways to bring people and ideas together despite geographical barriers.

There are lot of chat applications present in the digital marketplace, but it takes lot of storage for installation and memory space to run. Moreover, chat applications are very complicated, because of which non-technical users find it difficult to use.

The proposed chat application (WOJ) is implemented with custom-communication mechanism for interaction between chat-client and chat-server, intended to be lightweight, with easy-to-use GUI interface, so the user with less technical knowledge can also use it.

Any device with java virtual machine (JVM) gets to register as a client and start communication instantly. Chat-client application should be present on user's device to contact server application over internet to enable chat with other registered users.

The Object Oriented (OO) pattern makes sure that every module and class is analysed, tested, and updated separately.

2. METHODOLOGY

Since the invention of the computer, many approaches of program development have evolved. Due to the popularity of C language, structured programming became very popular and was the main technique of the 1980s. Later this technique failed to show the desired performance in terms of maintainability, reusability and reliability. As a result of this realization, a new methodology known as Object-Oriented programming emerged.[1]

2.1 Programming Language Paradigm

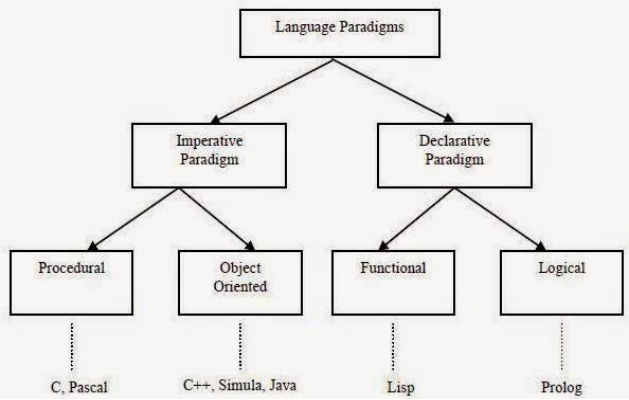


Fig -1: Language Paradigm

1. **Imperative:** programmer instructs the machine how to change its state.[2]

a. *Procedural* - groups instructions into procedures.

b. *Object-Oriented* - groups instructions with the part of the state they operate on.[2]

2. **Declarative:** programmer merely declares properties of the desired result, but not how to compute it.

a. *Functional* - the desired result is declared as the value of a series of function applications.

b. *Logic* - which the desired result is declared as the answer to a question about a system of facts and rules.[2]

Error free code with proper control can be achieved over each function making the development process fast and efficient yet Data were given lower priority while the emphasis was on "doing things" in Structured Programming. Unfortunately, functions and data structures failed to model the real world very well [1].

2.2 Object-Oriented Programming

The approach emphasis on writing programs having modules that can relate to real-world entity based upon how people perceive the world. The program is organized around the data being operated upon rather than the operations performed [1]. The data is combined with the set of functions that operate on the data to form a single unit called "object". With large number of cooperating objects, a work network can be formed to achieve the defined task. The definition of the object (data and functions) is specified in class according to which the object is formed.

Data is a main element in the development and is local to an object. From outside the object the data cannot be accessed directly as it remains encapsulated within an object.

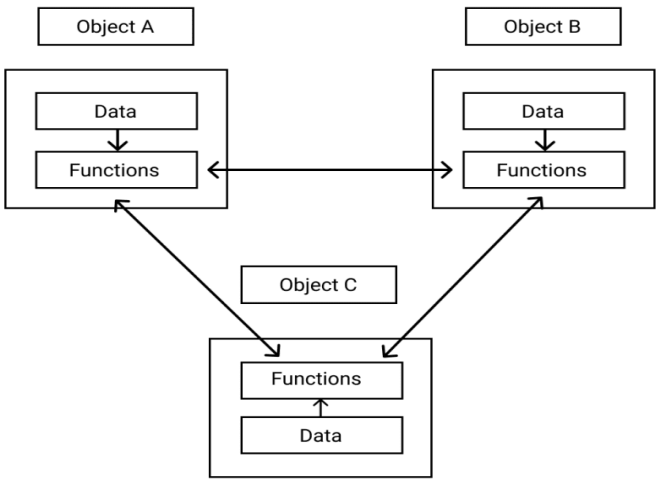


Fig -2: Object Oriented Approach

Features of Object-Oriented Programming:

1. Abstraction: represents only the relevant data and hides irrelevant data i.e. specifies necessary and sufficient descriptions rather than implementation details. It results in separation of interface and implementation [3].

2. Encapsulation: keeps both data and functions safe from outside interference and misuse. The advantage of encapsulated code is that user knows how to access it, there is no need of implementation details [4].

3. Inheritance: allows the extension and reuse of existing code, without having to repeat or rewrite the code from scratch. Inheritance involves the transfer of the parent's attribute to the children [3].

4. Polymorphism: the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object [5].

5. Concurrency: multi-threading a way to achieve concurrency in OOP like Java. A process can be divided into multiple threads that can run concurrently and each part can handle a different task at the same time, consuming available hardware resources efficiently specially when computer has multiple cores [6].

6. Event Handling: technique to handle an event (interrupt), monitor the event and along with code for what action to be performed if the event occurs. Since, objects are connected to one another therefore can pass the information and control in any defined order to another object, making handling easy.

7. Object Cloning: to create exact copy of an object by replicating all data functions to the cloned object thereby saves the extra processing task for creating the exact copy of an object [9]. Can be achieved by using method defined within object class.

3. WOJ

The chat mechanism implemented requires the constant interaction between a chat server and a chat client from the initial phase of user registration to sending text, data, etc. therefore the complete development is done in two parts: 1. Server Application development 2. Client Application development.

The java.net package in the Java platform provides a class, Socket, that implements one side of a two-way connection between one Java program and another program present on the network. The Socket class sits on top of a platform-dependent implementation, hiding the details of any particular system from your Java program

[7]. The Socket is used to create connection between chat server and chat client.

3.1 Server-Side

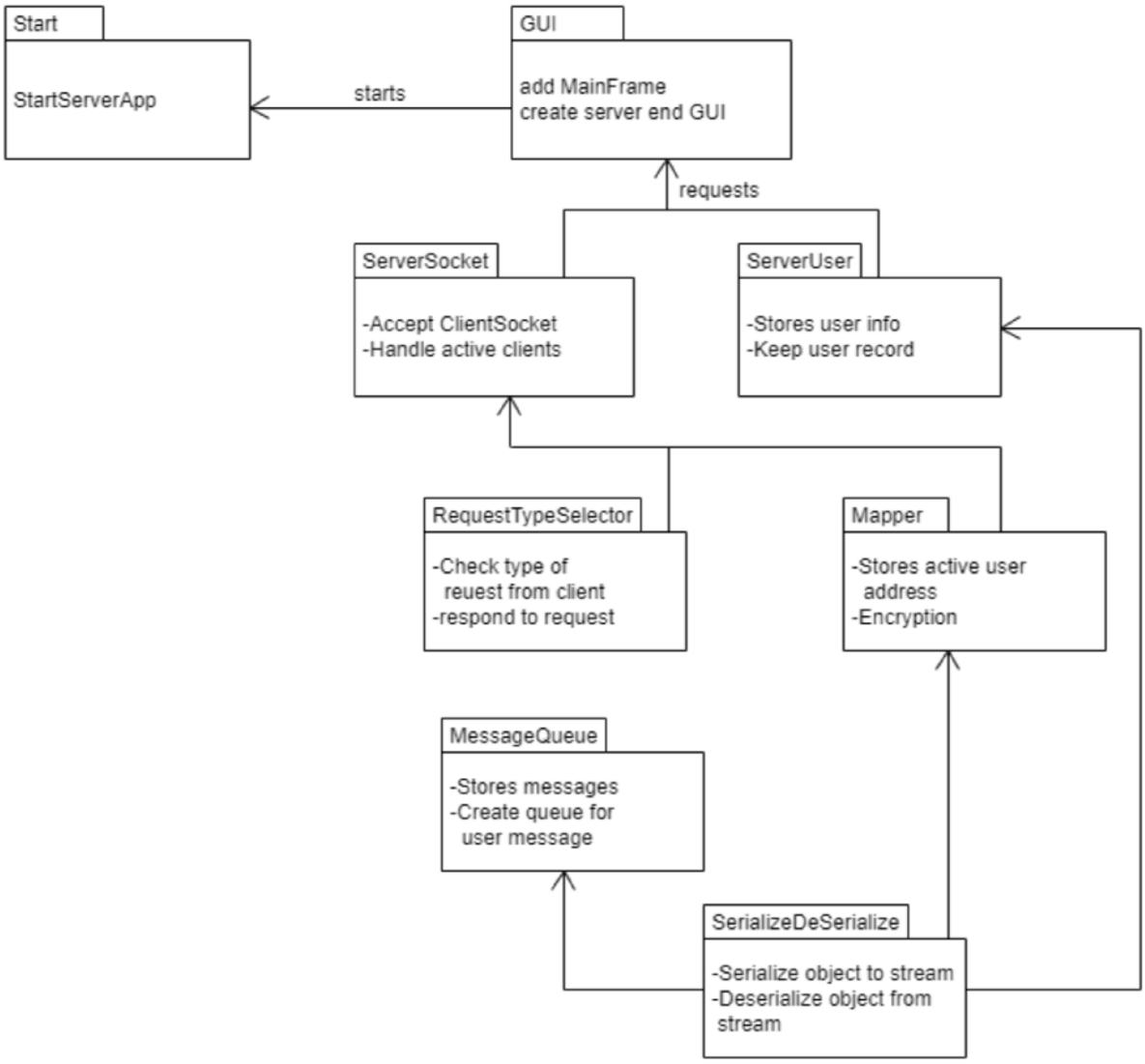


Fig -3: Server End Package

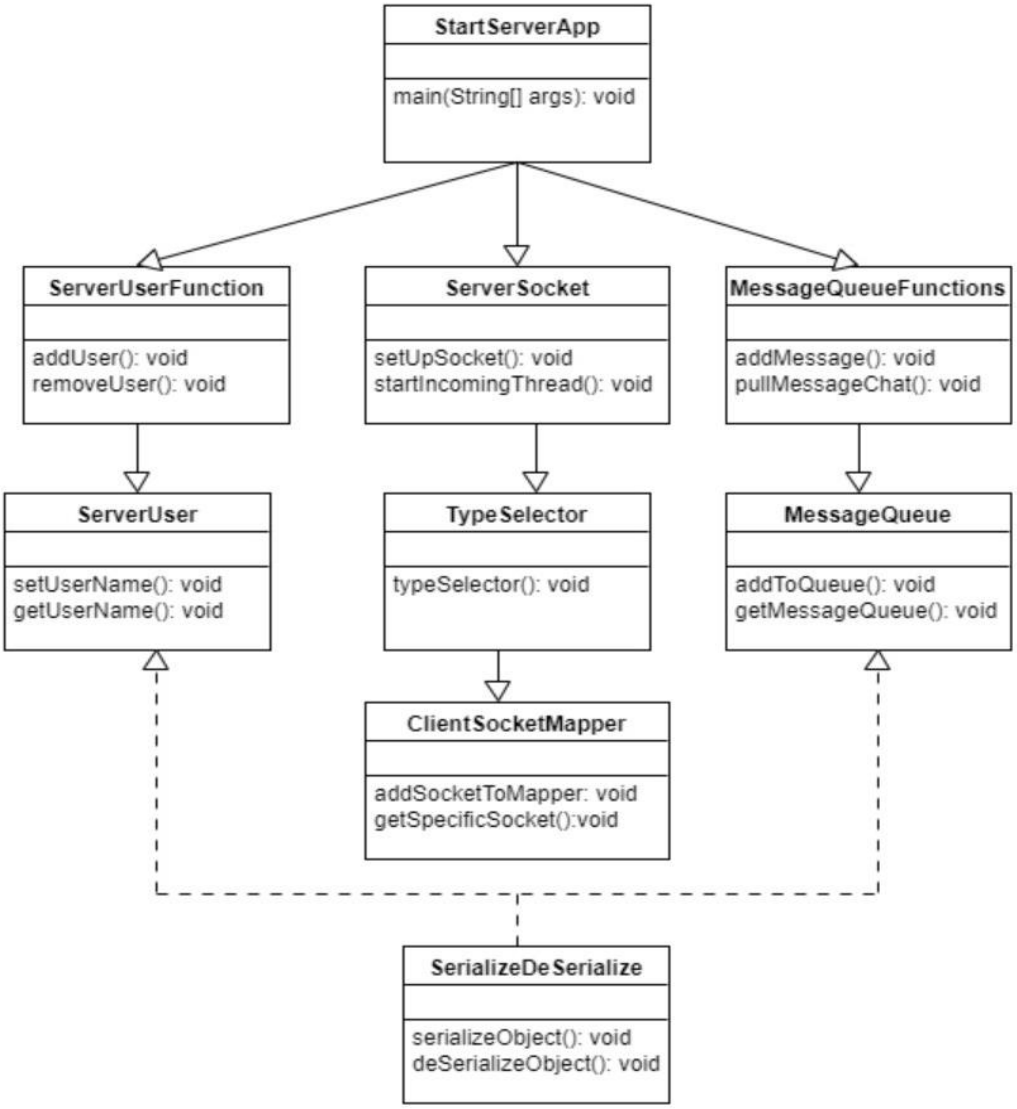


Fig -4: Server End Class Diagram

3.1.1 ServerSocket

This package contains server socket through which client can interact with server and this is where all the request of client gets processed and data of all the clients are stored at server. All the functionalities to give response to requests are defined in this package.

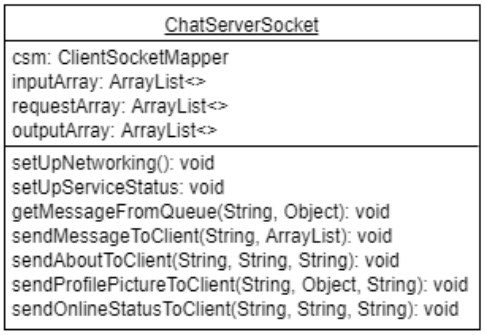


Fig -5: ChatServerSocket Class Diagram

3.1.2 MessageQueue

In this package all the messages and files (which are sent while chatting through socket) are stored in queue

data structure and functionality class contains all the functions to fetch messages from message queue.

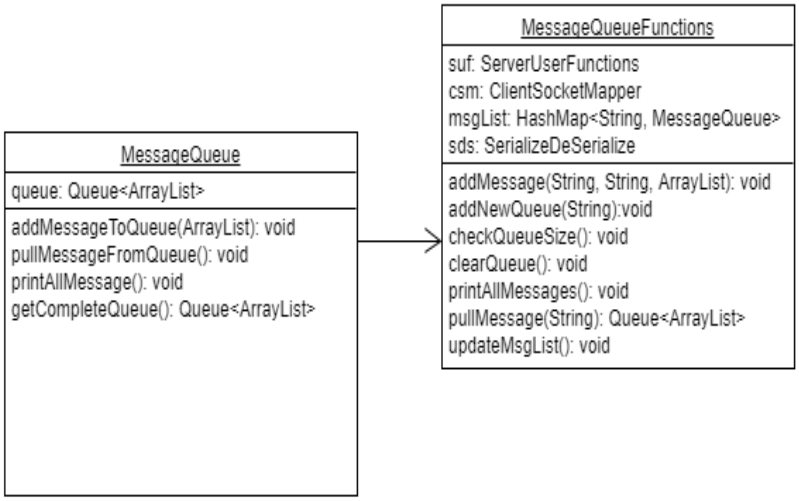


Fig -6: MessageQueue and MessageQueueFunctions Class Diagram

3.2 Client-Side

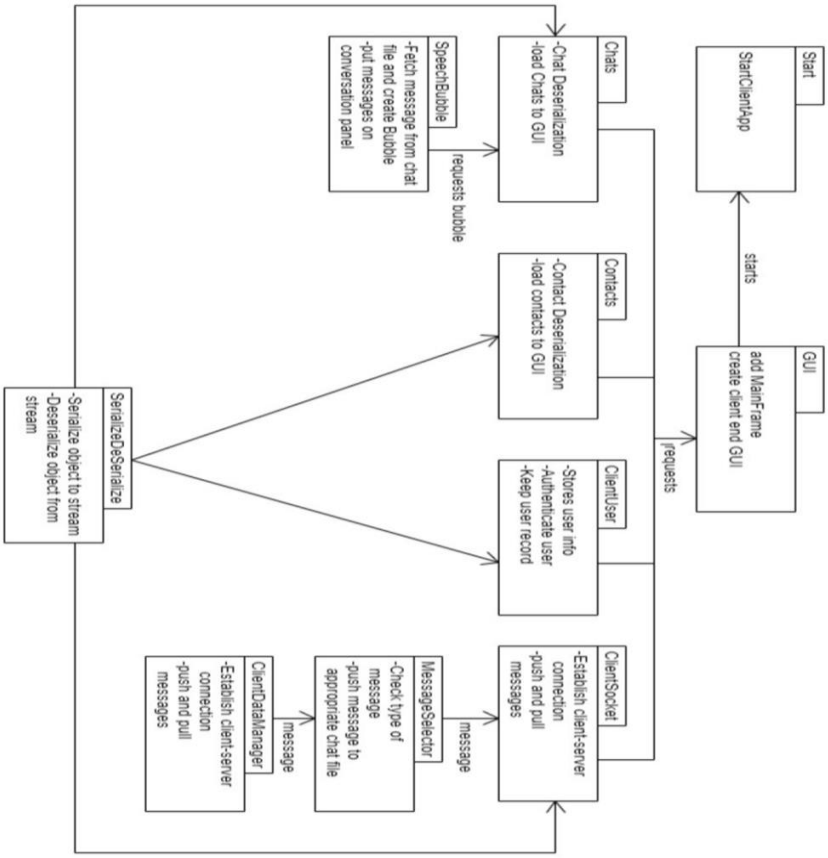


Fig -7: Client End Package

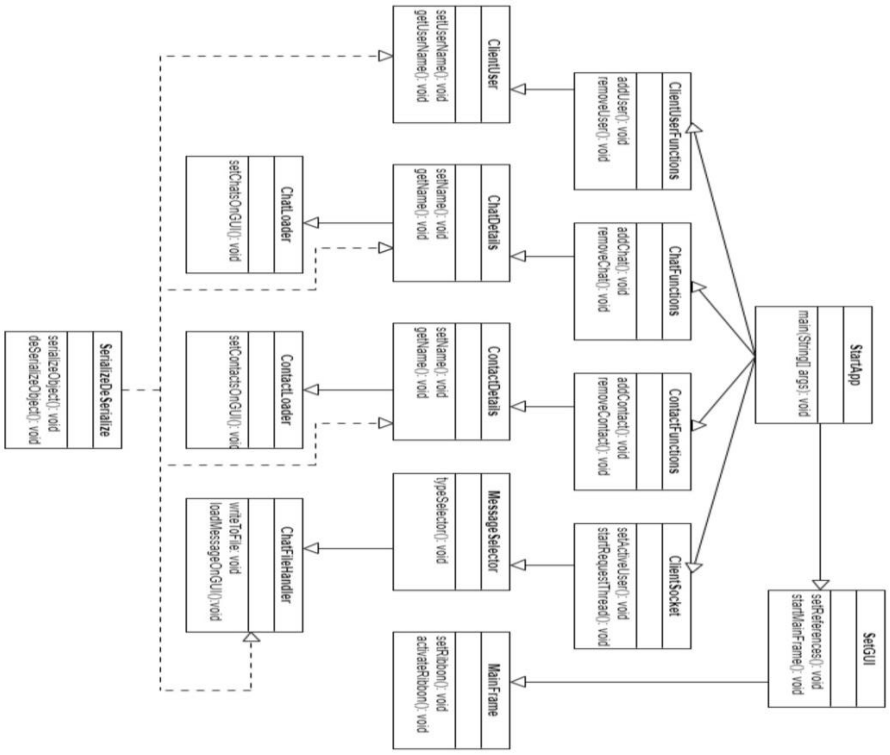


Fig -8: Client End Class Diagram

3.2.1 User

In this package all the records of registered users are maintained and all the functionalities to fetch, update, and delete user details are defined.

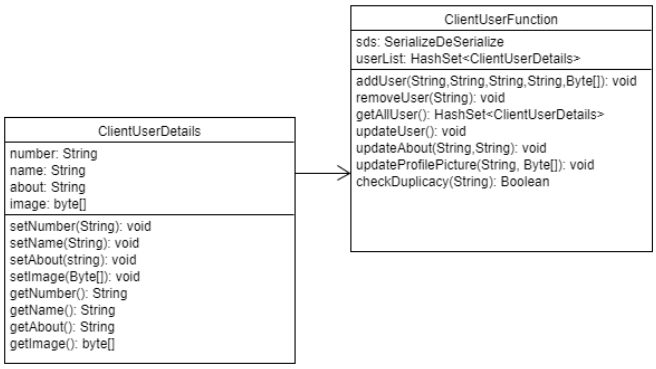


Fig -9: ClientUserDetails and ClientUserFunction Class Diagram

3.2.2 Chats

In this package all chats messages and information of chat user is maintained and functionalities to fetch, update, and delete chat is defined.

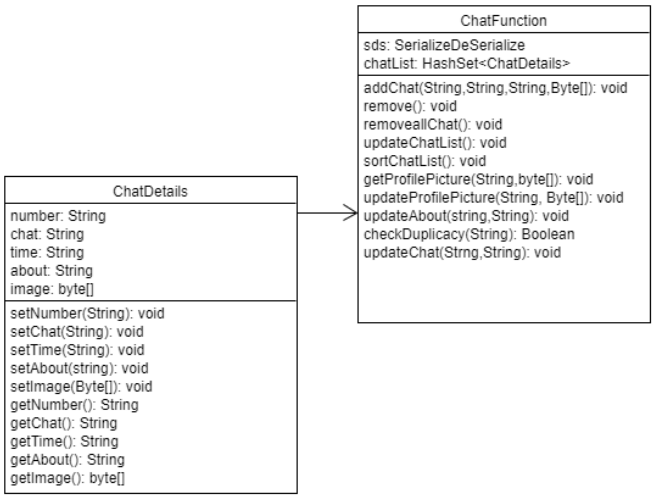


Fig -10: ChatDetails and ChatFunction Class Diagram

3.2.3 Contacts

In this package all the contact details are maintained. And functionalities to fetch, update, and delete contacts are defined.

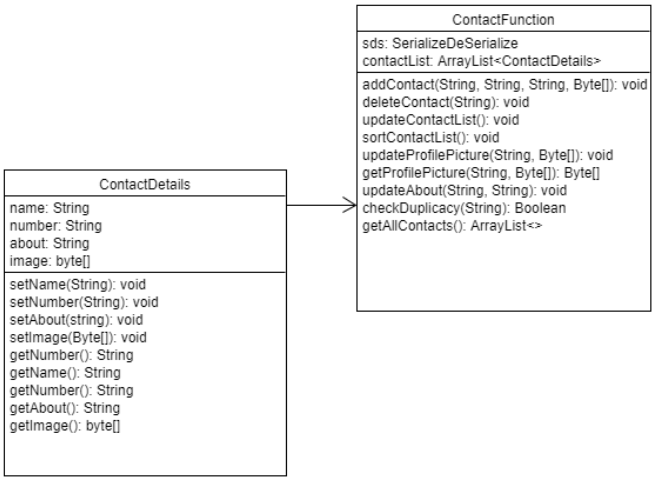


Fig -11: ContactDetails and ContactFunction Class Diagram

3.2.4 MessageQueue

In this package user messages are stored in Queue data structure. And functionalities to fetch and remove messages are defined.

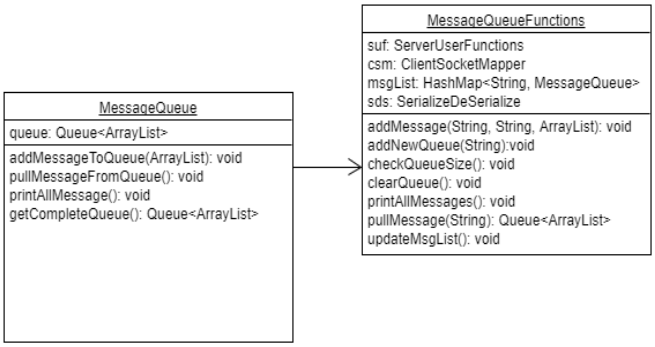


Fig -12: MessageQueue and MessageQueueFunction Class Diagram

3.2.5 ClientSocket

This package contains the client end socket through which client can interact with server. Functions for different kind of requests are defined such as: Registering client on server, uploading profile picture on server, and message fetching request.

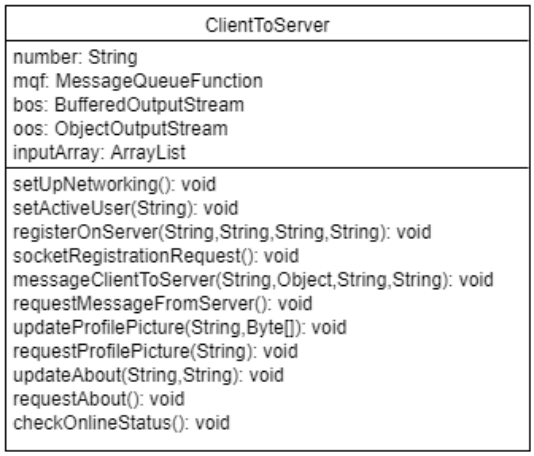


Fig -13: ClientToServer Class Diagram

3.3 Custom Request/Response Message

The communication between Server and Client application is maintained by passing custom ArrayList containing information of sender, receiver, message, message type, etc. in a custom defined sequence which then converted to Byte Array in order to be sent over the network.

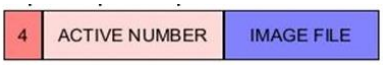
Here, zero (0) means the message sent from client to server is a request for new user registration, therefore, proper method defined to handle registration task will be executed accepting Number, Name, Email, About as arguments.

After successful registration, based upon message type (1), the server will register the socket object of the client, in order to provide service to the client.

Depending upon the message type (2), the file with its name, extension, date along with the destination information is sent to server from client.

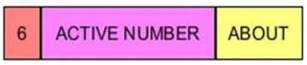
The other clients on getting online, generate the request to server for chat/conversation data via message type (3). In response, the server replies with message type (3.1)

containing the chat data which is processed at client application.

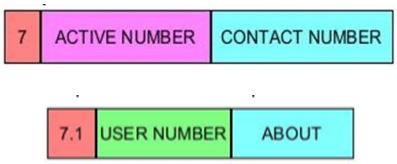


Profile picture can be updated via message type (4), while the image file is sent in the form of Byte Array.

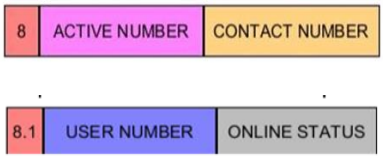
The request for profile picture is made with message type (5) from client to server. In response, the server sends back the image file in form of Byte Array in message type (5.1) which is processed at client application.



Message type (6), sent from client to server to update "About" section information.



Message type (7), sent from client to server to request "About" section information for a particular client using contact information. In response, server sends message type (7.1), containing "About" information to client.



Message type (8), to request online status from server. In response, server replies with message type (8.1), containing online status as Boolean data.

4. Result

The complete implementation of "WOJ" done using Java as a language of choice. None of the primitive data types used so as to keep the program completely object oriented. Alternatively, Wrapper classes (Byte, Integer, etc.) used. No DBMS used, the generated data is handled completely using custom algorithms, user defined methods along with built-in methods that comes with standard JDK installer. System Configuration used for development: Intel® Core™ i7-4700MQ CPU @ 2.40GHz, 8GB RAM, 240 GB SSD, Windows 10 64-bit, JDK 14.2, Eclipse IDE. Configuration used for execution of Server-Application: Intel® Core™ i7-6700MQ CPU @ 3.40GHz, 8GB RAM, 1TB HDD, Windows 10 64-bit, JVM 15, 100 Mbps connection speed between the network and the computer. Configuration used for execution of Client-Application(s): Intel® Core™ i7-6700MQ CPU @ 3.40GHz, 8GB RAM, 1TB HDD, Windows 10 64-bit, JVM 15, 100 Mbps connection speed between the network and the computer.

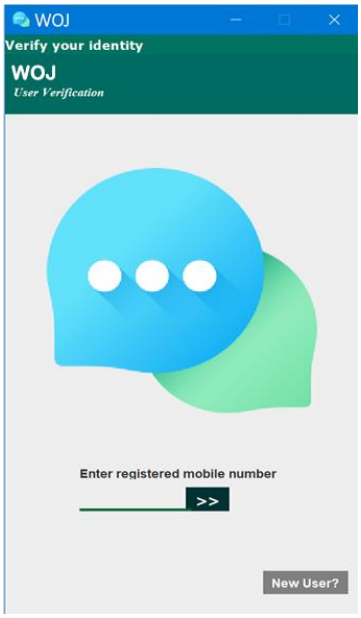


Fig -14: Login Screen

Enter mobile number on login screen, if registered chat panel will appear else click on new user button in the bottom.

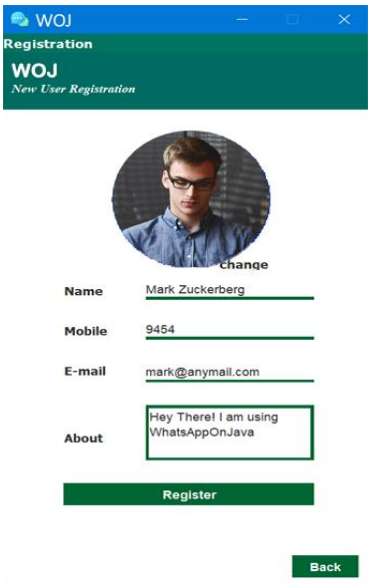


Fig -15: Registration Screen

If not registered, create new account by entering all the details on Registration Screen. After filling all the details click on Register button

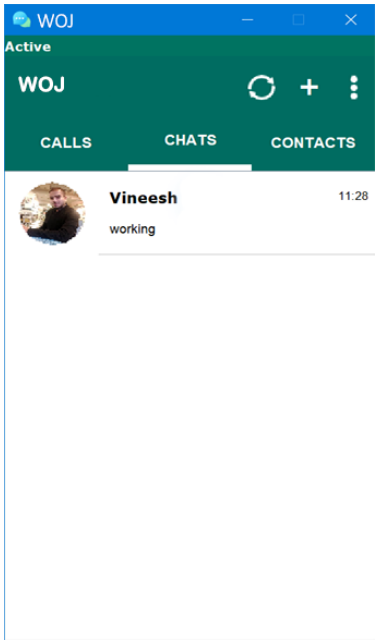


Fig -16: Chat Panel

If registered- Chat panel will appear, click on listed chats to start conversation.

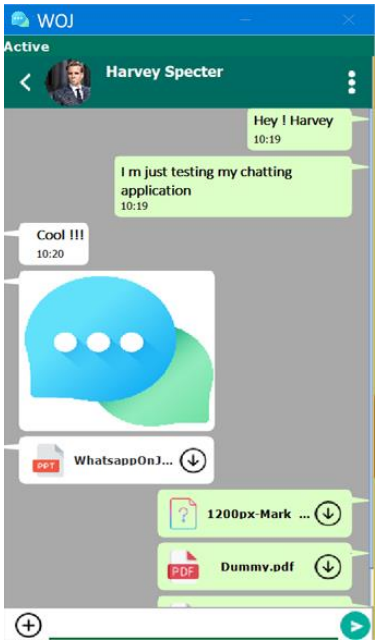


Fig -17: Conversation Panel

Conversation panel loads all the chat data of specific client as per the selection. Any message or file can be sent by clicking on plus button, JFileChooser will pop-up through which a selection can be made [10].

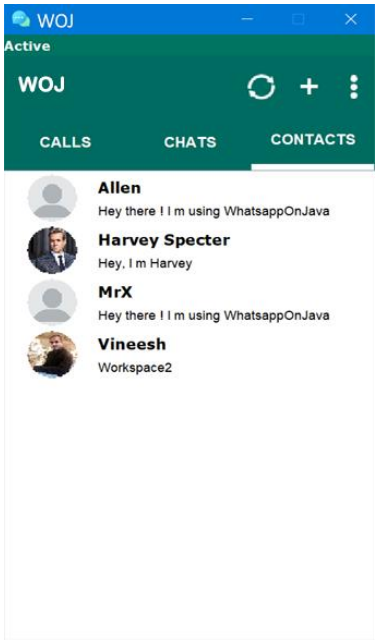


Fig -18: Contacts Panel

Any fresh chat can be initiated by clicking "CONTACTS" tabs that will load all the contacts. Clicking on any name will again open Conversation Panel.

Fig -19: About Panel

About Panel displays the information of the already registered client. Status text and number of the client is shown here, along with the profile picture.

5. CONCLUSIONS

This paper presents an easy object-oriented approach to build a platform independent chat application. All the OOPs concept presented are build using Java as a language of choice [8]. After complete implementation and execution, it can be concluded that so many difficulties and problems arise while designing fully functional chat server which holds the user information. "WOJ" is an efficient and light-weight chatting solution with easy-to-use GUI so that user with less technical knowledge can also use it. Object oriented design makes application easy to develop, maintain and update any module, class as per requirement.

Designing all functionalities in java makes it, highly independent of external/other technologies thereby extremely reducing the probability of getting the exceptions.

No matter how good anything can be there is always scope for improvement. Primary area of improvement is the fresh implementation of encryption techniques to secure the data when sent and received over network. Secondary, cloud based online backup for chat data can be added. Moreover, server-side application can be redesigned to support multiple instances working together in parallel in-order to handle the large requests when the number of client increases.

REFERENCES

[1] "Unit-1: Object Oriented Methodology-1", IGNOU, https://egyankosh.ac.in/bitstream/123456789/10090/1/Unit-1.pdf, © 2018 Copyright: IGNOU.

[2] "Programming paradigm", Wikipedia,

https://en.wikipedia.org/wiki/Programming_paradigm

[3] A. I. Isaiah, A. C. Odi, A. U. Rita, A. C. Verginia and O. H. Anaya, "Technological Advancement in Object Oriented Programming Paradigm for Software Development", International Journal of Applied Engineering Research, ISSN 0973-4562, Volume: 14, Number 8 (2019), pp. 1835-1841.

[4] R. S. Raut, "Research Paper on Object-Oriented Programming (OOP)", International Research Journal of Engineering and Technology (IRJET), ISSN: 2395-0056, Volume: 07 Issue: 10, Oct 2020.

[5] "Java - Polymorphism", Tutorialspoint,

https://www.tutorialspoint.com/java/java_polymorphism.htm

[6] "Java Concurrency Tutorial", Tutorialspoint,

https://www.tutorialspoint.com/java_concurrency/index.htm

[7] "What is a Socket?", The Java Tutorials, Oracle,

https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

[8] Pellet, Jean-Philippe, Amaury Dame, and Gabriel Parriaux. "How beginner-friendly is a programming language? A short analysis based on Java and Python examples." (2019).

[9] Fatima, N., and S. Arabia. "Performance comparison of most common high level programming languages." International Journal of Computing Academic Research (IJCAR) 5.5 (2016): 246-258.

[10] Kendal S., (2209), Object Oriented Programming using Java, Bookboon eBook Company, ISBN 978-87-7681-501-1, 16P.