Anjuman-I-Islam's
M.H. Saboo Siddik Polytechnic

SUBJECT NAME: ADVANCED JAVA PROGRAMMING (AJP) - 22517

DEPARTMENT: COMPUTER ENGINEERING

SEMESTER: FIFTH

MICRO PROJECT TITLE: CHAT APPLICATION

YEAR: 2023-24

PREPARED BY:

Names of Team Members with Roll Nos.

1. Abdurrahman Qureshi - 210451
2. Oaish Qazi - 210455
3. Shaikh Mohammed Hussain - 220486

UNDER THE GUIDANCE OF: Prof. Zaibunnisa Malik

Maharashtra State
Board of Technical Education

Certificate

This is to certify that Mr. Abdurrahman Qureshi of Fifth Semester of Diploma in Computer Engineering of Institute M.H. Saboo Siddik Polytechnic has successfully completed Micro-project work in subject Advanced Java Programming (22517) for the academic year 2023- 2024 as prescribed in the I-Scheme Curriculum.

Place: …… Enrollment no: ……..

Date: …… Exam seat no: ……....

Signature	Signature	Signature
Project Guide	H. O. D	Principal

Seal of Institute

Maharashtra State
Board of Technical Education

Certificate

This is to certify that Mr. Oaish Qazi of Fifth Semester of Diploma in Computer Engineering of Institute M.H. Saboo Siddik Polytechnic has successfully completed Micro-project work in subject Advanced Java Programming (22517) for the academic year 2023-2024 as prescribed in the I-Scheme Curriculum.

Place: …… Enrollment no: ……..

Date: …… Exam seat no: ……....

Signature	Signature	Signature
Project Guide	H. O. D	Principal

Seal of Institute

Maharashtra State
Board of Technical Education

Certificate

This is to certify that Mr. Shaikh Mohammed Hussain of Fifth Semester of Diploma in Computer Engineering of Institute M.H. Saboo Siddik Polytechnic has successfully completed Micro-project work in subject Advanced Java Programming (22517) for the academic year 2023-2024 as prescribed in the I-Scheme Curriculum.

Place: …… Enrollment no: ……..

Date: …… Exam seat no: ……....

Signature	Signature	Signature
Project Guide	H. O. D	Principal

Seal of Institute

ACKNOWLEDGMENT

We wish to express our profound gratitude to our guide Ms. Zaibunnisa Malik who guided us endlessly in the framing and completion of the micro project. He guided us on all the main points in that micro project. We are indebted to his constant encouragement, cooperation, and help. It was his enthusiastic support that helped us in overcoming various obstacles in the micro-project.

We are also thankful to our Principal, HOD, faculty members and classmates of Computer Engineering department for extending their support and motivation in the completion of this micro-project.

Names of Team Members with Roll Nos.

1. Abdurrahman Qureshi - 210454
2. Oaish Qazi - 210455
3. Shaikh Mohammed Hussain - 220486

Annexure-I

Micro-Project Proposal

Chat Application

1.0 Aims/Benefits of the Micro-Project

The aim of this micro-project is to create a real-time chat application using web technologies, with a specific focus on learning and applying Servlets and JSP for server-side programming. By undertaking this project, you can gain valuable hands-on experience in developing interactive web applications. The benefits of this endeavor include enhancing your understanding of Java-based web development, improving your proficiency in Servlets, JSP, and web application architecture, and achieving the creation of a functional chat application for personal or educational use. Additionally, this project presents the opportunity to potentially publish your application online, serving as a showcase of your web development skills.

2.0 Course Outcomes Addressed

d) Develop java programs using networking concept.
e) Develop programs using database.
f) Develop program using Servlet

3.0 Proposed Methodology

- Discussion of topic with guide and group members.
- Dividing the work and Gathering information about the project.
- Submission of project proposal. (Annexure I).
- Collection and Analysis of data / information from various sources.
- Set up the development environment (e.g., IntelliJ IDEA, Tomcat server).
- Design the user interface for the chat application using JSP.
- Implement Servlets to handle user registration, login, and chat functionality.
- Utilize AJAX or WebSockets for real-time communication between users.
- Store chat messages in a database for persistence.
- Apply security measures to protect against common web vulnerabilities.
- Test the Project for Bugs and Errors.
- Preparation of the project report (Annexure II).
- Microproject Submission and Viva.

Annexure-I

4.0 Action Plan

Sr. No.	Week	Details of activity	Planned Start date	Planned Finish date	Name of Responsible Team Members
1	1 & 2	Discussion and finalization of the			All
2	3 & 4	Dividing the work among group members.			All
3	5	Submission of micro project proposal (Annexure I)			All
4	6	Collection of the information on the Topic.			All
6	7	Collection of all relevant content / materials for the execution of the project.			All
7	8 & 9	Execution of collected data and preparing layout of the web app and building servlets and websockets.			Oaish / Abdurrahman
8	10	Integration of frontend with backend			Oaish / Abdurrahman
9	11	Testing the project for bugs and errors			Oaish / Abdurrahman
10	12	Preparation of the project report (Annexure II)			All
11	13	Microproject Submission and Viva			All

5.0 Resources Required

Sr. No.	Name of Resource/material	Specifications	Quantity	Remarks
1	Software	Tomcat Server 9, MySQL Server, Git	1 of Each	
2	CASE Tools	IntelliJ, VS Code, Notepad	1 of Each	
3	Browser	Opera GX, Firefox, Chrome, Edge	1 of Each	

Names of Team Members with Roll Nos.

1. Abdurrahman Qureshi - 210454
2. Oaish Qazi - 210455
3. Shaikh Mohammed Hussain - 220486

(To be approved by the concerned teacher)

****

Annexure - I I

Micro-Project Report

Chat Application

1.0 Rationale

This project helps students or developers understand the practical application of Java-based web technologies. It allows for the implementation of real-time features, demonstrating the power of web applications. Knowledge gained can be valuable for creating interactive web applications and enhancing employability.

2.0 Aims/Benefits of the Micro-Project:

The aim of this micro-project is to create a real-time chat application using web technologies, with a specific focus on learning and applying Servlets and JSP for server-side programming. By undertaking this project, you can gain valuable hands-on experience in developing interactive web applications. The benefits of this endeavor include enhancing your understanding of Java-based web development, improving your proficiency in Servlets, JSP, and web application architecture, and achieving the creation of a functional chat application for personal or educational use. Additionally, this project presents the opportunity to potentially publish your application online, serving as a showcase of your web development skills.

3.0 Course Outcomes Achieved

d) Develop java programs using networking concept.
e) Develop programs using database.
f) Develop program using Servlet

4.0 Literature Review

- Research existing chat applications and their architectures.
- Study Java Servlets and JSP documentation and tutorials.
- Review best practices for securing web applications.
- Explore AJAX or WebSocket communication in web development.
- Learn about database integration for web applications (e.g., JDBC).

5.0 Actual Methodology Followed

- Discussed the topic with guide and group members.
- Divided the work and Gathered information about the project.
- Submitted project proposal. (Annexure I).
- Collected and Analyzed data / information from various sources.
- The development environment set up (e.g., IntelliJ IDEA, Tomcat server).
- Designed the user interface for the chat application using JSP.
- Implemented Servlets to handle user registration, login, and chat functionality.
- Utilize AJAX or WebSockets for real-time communication between users.
- Store chat messages in a database for persistence.
- Tested the Project for Bugs and Errors.
- Prepared the project report (Annexure II).
- Microproject Submitted with Viva.

6.0 Actual Resources Used (Mention the actual resources used).

Sr. No.	Name of Resource/material	Specifications	Quantity	Remarks
1	Software	IntelliJ IDEA, FireFox, MySQL, Tomcat Server	1	
2	Websites	web.whatsapp.com, youtube.com, geeksforgeeks.com	1	

7.0 Outputs of the Micro-Projects

Java Swing

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

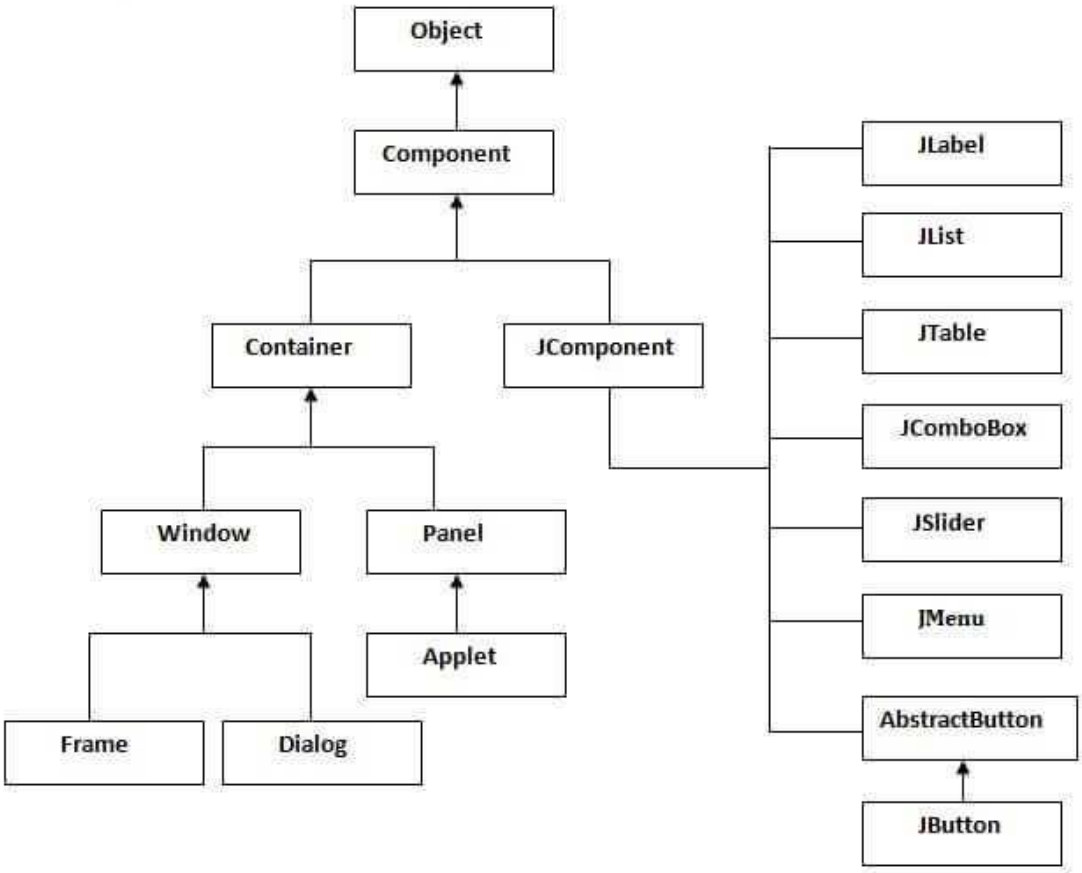
The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

- **Difference between AWT and Swing**

There are many differences between java awt and swing that are given below.

No.	Java AWT	Java Swing
1)	AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
2)	AWT components are **heavyweight**.	Swing components are **lightweight**.
3)	AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
4)	AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing **follows MVC**.

- **Hierarchy of Java Swing classes**
- The hierarchy of java swing API is given below**.**



- **Commonly used Methods of Component class**

The methods of Component class are widely used in java swing that are given below.

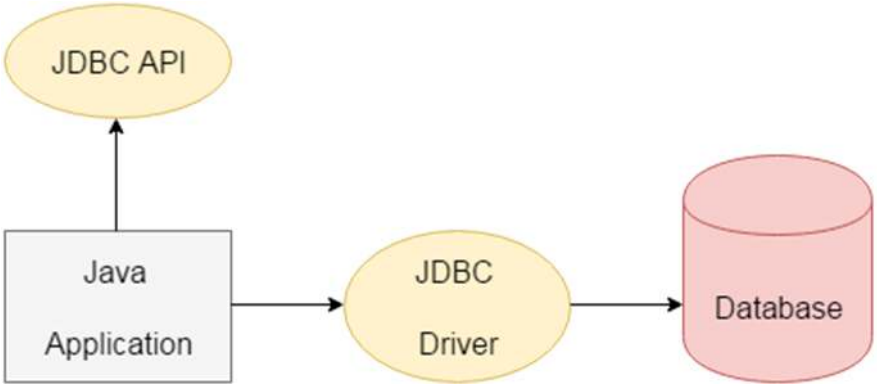
Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

- **Java JDBC**

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface
    - A list of popular *classes* of JDBC API are given below:
- DriverManager class
- Blob class
- Clob class
- Types class

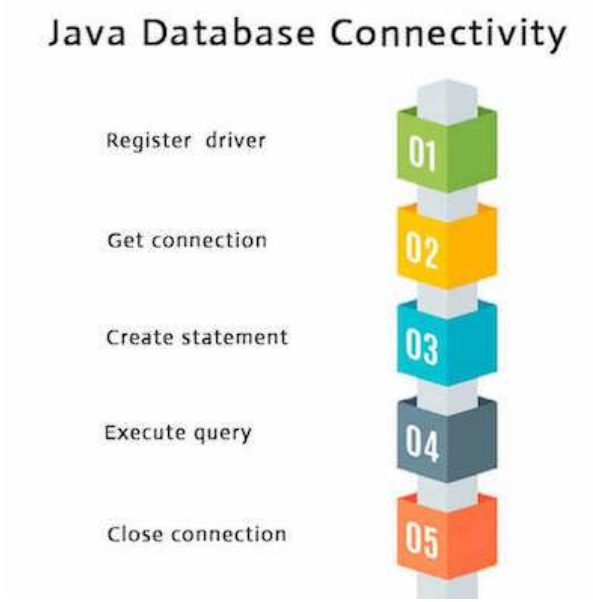
- **Why Should We Use JDBC**

  Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

  We can use JDBC API to handle database using Java program and can perform the following activities:

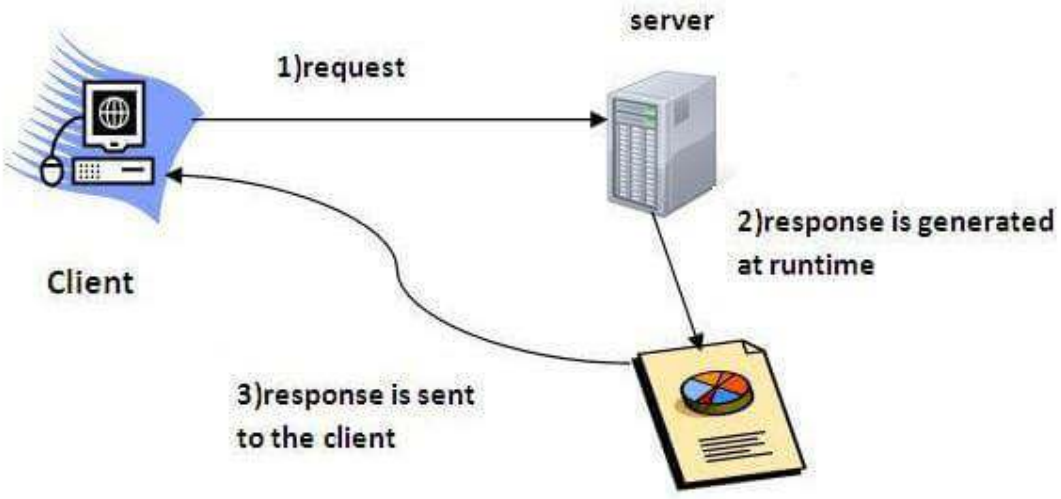
  1. Connect to the database
  2. Execute queries and update statements to the database
  3. Retrieve the result received from the database.

  There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:



- **Servlets**
  Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).
  Servlet technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below.
  There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

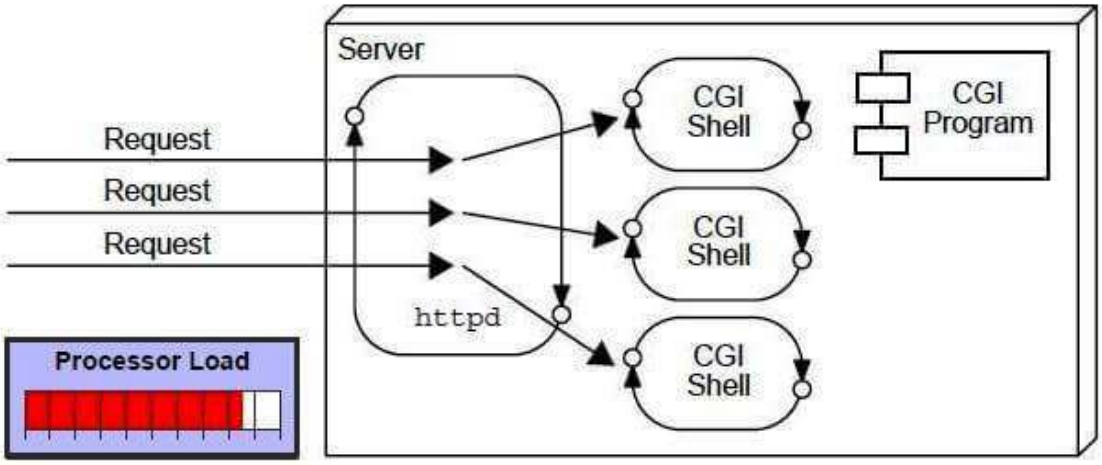
- **What is a Servlet?**
  Servlet can be described in many ways, depending on the context.
  - Servlet is a technology which is used to create a web application.
  - Servlet is an API that provides many interfaces and classes including documentation.
  - Servlet is an interface that must be implemented for creating any Servlet.
  - Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
  - Servlet is a web component that is deployed on the server to create a dynamic web page.



- **What is a web application?**
  A web application is an application accessible from the web. A web application is composed of web components like Servlet, JSP, Filter, etc. and other elements such as HTML, CSS, and

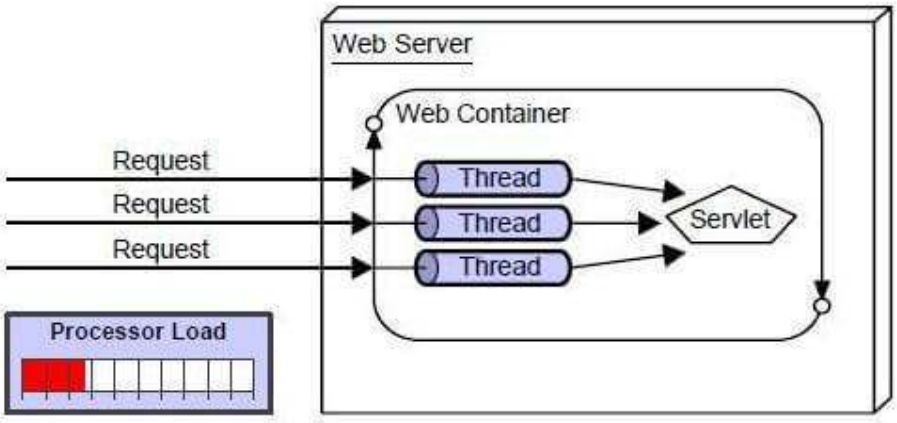
JavaScript. The web components typically execute in Web Server and respond to the HTTP request.

- **CGI (Common Gateway Interface)**
  CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.



- **Disadvantages of CGI**
  There are many problems in CGI technology:
  1. If the number of clients increases, it takes more time for sending the response.
  2. For each request, it starts a process, and the web server is limited to start processes.
  3. It uses platform dependent language e.g. C, C++, perl.

- **Advantages of Servlet**



There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

1. Better performance: because it creates a thread for each request, not process.
2. Portability: because it uses Java language.
3. Robust: JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. Secure: because it uses java language.

Backend Code:

JSPServlet:

```
package com.talkwave;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/jsp")
public class JSPServlet extends HttpServlet {
   @Override
   protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
      RequestDispatcher dispatcher = req.getRequestDispatcher("auth.jsp");
      dispatcher.forward(req, resp);
   }
}
```

WebSocketServlet:

```
package com.talkwave;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.talkwave.handler.MessageHandler;
import com.talkwave.handler.StatusHandler;

import javax.servlet.ServletException;
import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.ServerEndpoint;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import java.util.logging.Logger;

@ServerEndpoint("/websocket")
public class WebSocketServlet {
   Logger logger = Logger.getLogger(WebSocketServlet.class.getName());
   private static final Map<String, Session> sessions = new ConcurrentHashMap<>();
   @OnOpen
   public void onOpen(Session session) {}
   @OnMessage
   public void onMessage(String message, Session session) throws IOException,
SQLException, ClassNotFoundException {
      if (message.contains("$id:")) {
         String id = message.substring(4);
         sessions.put(id, session);
         handleStatus(id, "online");
```

```
    return;
}

if (message.contains("$offline:")) {
    String id = message.substring(9);
    handleStatus(id, "offline");
    return;
}

if (message.contains("$chat-active:")) {
    String[] chatID = message.substring(13).split("&");
    StatusHandler statusHandler = new StatusHandler();
    /* 0: Sender; 1: Receiver */
    statusHandler.setMsgStatus(chatID[0], chatID[1]);
    Session msgSenderSession = sessions.get(chatID[0]);
    try {
        msgSenderSession.getAsyncRemote().sendText(message);
    } catch (NullPointerException e) {
        session.getAsyncRemote().sendText("NullPointerException: " + e.getMessage());
    }
    return;
}

if (message.contains("$chat-inactive:")) {
    String[] chatID = message.substring(15).split("&");
    /* 0: ChatUser; 1: ChatViewer */
    Session chatUserSession = sessions.get(chatID[0]);
    try {
        chatUserSession.getAsyncRemote().sendText(message);
    } catch (NullPointerException e) {
        logger.info("NullPointerException: " + e.getMessage());
        session.getAsyncRemote().sendText("NullPointerException: " + e.getMessage());
    }
    return;
}

if (message.contains("$add-friend:")) {
    String[] chatID = message.substring(12).split("&");
    Session chatUserSession = sessions.get(chatID[0]);
    chatUserSession.getAsyncRemote().sendText(message);
    return;
}

MessageHandler msgHandler = new MessageHandler();
msgHandler.insertMessage(message);
msgHandler.updateLastMessage(message);
msgHandler.close();

ObjectMapper mapper = new ObjectMapper();

JsonNode msg = mapper.readTree(message);
String receiverID = msg.get("receiverID").toString().replace("\"", "");

Session recipientSession = sessions.get(receiverID);

try {
```

```
            recipientSession.getAsyncRemote().sendText(message);
        } catch (NullPointerException e) {
            e.printStackTrace();
            session.getAsyncRemote().sendText("RecipientNullException: " + e);
        }

    }

    @OnClose
    public void onClose(Session session) {
        String closedSessionId = null;
        for (Map.Entry<String, Session> entry : sessions.entrySet()) {
            if (entry.getValue().equals(session)) {
                closedSessionId = entry.getKey();
                break;
            }
        }

        if (closedSessionId != null)
            sessions.remove(closedSessionId);
    }

    public void handleStatus(String id, String status) throws SQLException,
ClassNotFoundException, IOException {
        StatusHandler statusHandler = new StatusHandler();
        statusHandler.setUserStatus(id, status);
        List<String> list = statusHandler.getFriendsList(id);
        for (String item : list) {
            Session s = sessions.get(item);
            if (s != null) {
                s.getBasicRemote().sendText("$" + status + ":" + id);
            }
        }
        statusHandler.close();
    }

}
```

AddFriendServlet:

```
package com.talkwave.api;

import java.io.*;
import java.sql.*;
import java.util.logging.Logger;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

import com.talkwave.Env;

@WebServlet(name = "addFriendServlet", value = "/api-add-friend")
public class AddFriendServlet extends HttpServlet {
    public void init() {}
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String senderID = request.getParameter("senderID");
        String receiverID = request.getParameter("receiverID");
```

```
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(Env.DB_URL,
Env.DB_USERNAME, Env.DB_PASSWORD);
            PreparedStatement ps = con.prepareStatement("INSERT INTO friends VALUES
(DEFAULT, ?, ?, ''), (DEFAULT, ?, ?, '')");
            ps.setString(1, senderID);
            ps.setString(2, receiverID);
            ps.setString(3, receiverID);
            ps.setString(4, senderID);
            ps.execute();

            out.println("{\"status\":\"OK\"}");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            out.println("{\"error\":\""+e.getMessage()+"\"}");
        }
    }

    public void destroy() {
    }
}
```

AuthenticationServlet:

```
package com.talkwave.api;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;
import com.talkwave.Env;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.util.logging.Logger;

@WebServlet(name = "authenticationServlet", value = "/api-authenticate")
public class AuthenticationServlet extends HttpServlet {
    Connection con = null;
    PreparedStatement ps;
    ResultSet rs;
    String jsonMsgData;
    Logger logger = Logger.getLogger(AuthenticationServlet.class.getName());
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setContentType("application/json");
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        PrintWriter out = resp.getWriter();
```

```
        ObjectMapper objectMapper = new ObjectMapper();
        int rowCount = 0;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            con = DriverManager.getConnection(Env.DB_URL, Env.DB_USERNAME,
Env.DB_PASSWORD);
            if (password == null) {
                ps = con.prepareStatement("SELECT * FROM users WHERE username = ?");
                ps.setString(1, username);
                rs = ps.executeQuery();
                while (rs.next()) {
                    rowCount++;
                }
                jsonMsgData = "{\"isValid\":\"true\"}";
            } else {
                ps = con.prepareStatement("SELECT * FROM users WHERE username = ? AND
password = ?");
                ps.setString(1, username);
                ps.setString(2, password);

                rs = ps.executeQuery();

                while (rs.next()) {
                    rowCount++;
                    ObjectNode userNode = objectMapper.createObjectNode();
                    userNode.put("id", rs.getString(1));
                    userNode.put("username", rs.getString(2));
                    userNode.put("profileName", rs.getString(4));
                    userNode.put("image", rs.getString(6));
                    userNode.put("isValid", "true");
                    jsonMsgData = userNode.toString();
                }
            }

            if (rowCount < 1) {
                jsonMsgData = "{\"isValid\":\"false\"}";
            }

            out.println(jsonMsgData);
        } catch (ClassNotFoundException | SQLException e) {
            logger.info("Error: " + e.getMessage());
        }

    }
}
```

GetChatMsgServlet:

```
package com.talkwave.api;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ArrayNode;
import com.fasterxml.jackson.databind.node.ObjectNode;
import com.talkwave.Env;
import com.talkwave.JSPServlet;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.util.logging.Logger;

@WebServlet(name = "getChatMsgServlet", value = "/api-get-chat-msg")
public class GetChatMsgServlet extends HttpServlet {
   Connection con = null;
   PreparedStatement ps;
   ResultSet rs;
   String jsonMsgData;
   Logger logger = Logger.getLogger(JSPServlet.class.getName());
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
      resp.setContentType("application/json");
      String senderID = req.getParameter("senderID");
      String receiverID = req.getParameter("receiverID");
      PrintWriter out = resp.getWriter();

      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         con = DriverManager.getConnection(Env.DB_URL, Env.DB_USERNAME,
Env.DB_PASSWORD);
         ps = con.prepareStatement("SELECT * FROM messages WHERE (sender_id = ? AND
receiver_id = ?) OR (sender_id = ? AND receiver_id = ?) ORDER BY message_id");

         ps.setInt(1, Integer.parseInt(senderID));
         ps.setInt(2, Integer.parseInt(receiverID));
         ps.setInt(3, Integer.parseInt(receiverID));
         ps.setInt(4, Integer.parseInt(senderID));

         rs = ps.executeQuery();

         ObjectMapper mapper = new ObjectMapper();
         ArrayNode arrayNode = mapper.createArrayNode();

         while (rs.next()) {
            ObjectNode userNode = mapper.createObjectNode();
            userNode.put("id", rs.getString(1));
            userNode.put("senderID", rs.getString(2));
            userNode.put("receiverID", rs.getString(3));
            userNode.put("content", rs.getString(4));
            userNode.put("timestamp", rs.getString(5));
            userNode.put("readReceipt", rs.getString(6));
            arrayNode.add(userNode);
         }

         jsonMsgData = arrayNode.toString();

         out.println(jsonMsgData);
      } catch (ClassNotFoundException | SQLException e) {
```

```
            logger.info("Error: " + e.getMessage());
        }

    }
}
```

GetFriendsServlet:

```
package com.talkwave.api;

import java.io.*;
import java.sql.*;
import java.util.logging.Logger;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ArrayNode;
import com.fasterxml.jackson.databind.node.ObjectNode;
import com.talkwave.Env;

@WebServlet(name = "getFriendsServlet", value = "/api-get-friends")
public class GetFriendsServlet extends HttpServlet {
    public void init() {}

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String senderID = request.getParameter("senderID");
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(Env.DB_URL,
Env.DB_USERNAME, Env.DB_PASSWORD);
            PreparedStatement ps = con.prepareStatement("SELECT user_id, username, password,
profile_name, status, last_msg, image FROM friends f JOIN users u ON u.user_id = f.y_id
WHERE x_id = ?");
            ps.setString(1, senderID);
            ResultSet rs = ps.executeQuery();

            ObjectMapper objectMapper = new ObjectMapper();

            ArrayNode arrayNode = objectMapper.createArrayNode();

            while (rs.next()) {
                ObjectNode userNode = objectMapper.createObjectNode();
                userNode.put("id", rs.getString(1));
                userNode.put("username", rs.getString(2));
                userNode.put("password", rs.getString(3));
                userNode.put("profileName", rs.getString(4));
                userNode.put("status", rs.getString(5));
                userNode.put("lastMsg", rs.getString(6));
                userNode.put("image", rs.getString(7));
                arrayNode.add(userNode);
            }
```

```
            String json = arrayNode.toString();

            out.println(json);
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            out.println("{\"error\":\""+e.getMessage()+"\"}");
        }
    }

    public void destroy() {
    }
}
```

GetSuggestedServlet:

```
package com.talkwave.api;

import java.io.*;
import java.sql.*;
import java.util.logging.Logger;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ArrayNode;
import com.fasterxml.jackson.databind.node.ObjectNode;
import com.talkwave.Env;

@WebServlet(name = "getSuggestedServlet", value = "/api-get-suggested")
public class GetSuggestedServlet extends HttpServlet {
    public void init() {}
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String senderID = request.getParameter("senderID");
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(Env.DB_URL,
Env.DB_USERNAME, Env.DB_PASSWORD);
            PreparedStatement ps = con.prepareStatement("SELECT user_id, username, password,
profile_name, image from users WHERE user_id NOT IN (SELECT y_id FROM friends
WHERE x_id = ?) AND user_id <> ? ORDER BY user_id DESC LIMIT 10");
            ps.setString(1, senderID);
            ps.setString(2, senderID);
            ResultSet rs = ps.executeQuery();

            ObjectMapper objectMapper = new ObjectMapper();
            ArrayNode arrayNode = objectMapper.createArrayNode();

            while (rs.next()) {
                ObjectNode userNode = objectMapper.createObjectNode();
                userNode.put("id", rs.getString(1));
                userNode.put("username", rs.getString(2));
                userNode.put("password", rs.getString(3));
                userNode.put("profileName", rs.getString(4));
```

```
            userNode.put("image", rs.getString(5));
            arrayNode.add(userNode);
        }

        String json = arrayNode.toString();
        out.println(json);
    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
        out.println("{\"error\":\""+e.getMessage()+"\"}");
    }
  }
  public void destroy() {}
}
```

RegisterUserServlet:

```
package com.talkwave.api;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.talkwave.Env;
import com.talkwave.JSPServlet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.util.logging.Logger;

@WebServlet(name = "registerUserServlet", value = "/api-register-user")
public class RegisterUserServlet extends HttpServlet {
  Connection con = null;
  PreparedStatement ps;
  ResultSet rs;
  String jsonMsgData;
  Logger logger = Logger.getLogger(JSPServlet.class.getName());

  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    resp.setContentType("application/json");
    PrintWriter out = resp.getWriter();
    ObjectMapper objectMapper = new ObjectMapper();
    JsonNode jsonNode = objectMapper.readTree(req.getInputStream());
    String username = jsonNode.get("username").asText();
    String password = jsonNode.get("password").asText();
    String profileName = jsonNode.get("profileName").asText();
    String image = jsonNode.get("image").asText();
    try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      con = DriverManager.getConnection(Env.DB_URL, Env.DB_USERNAME,
Env.DB_PASSWORD);
      ps = con.prepareStatement("INSERT INTO
users(username,password,profile_name,image)VALUES(?,?,?,?)");
```

```
            ps.setString(1, username);
            ps.setString(2, password);
            ps.setString(3, profileName);
            ps.setString(4, image);
            ps.execute();
            out.println("{\"status\":\"OK\"}");
        } catch (ClassNotFoundException | SQLException e) {
            logger.info("Error: " + e.getMessage());
            out.println("{\"status\":\"Error\"}");
        }
    }
}
```

MessageHandler:

```
package com.talkwave.handler;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.talkwave.Env;

import java.sql.*;
import java.util.logging.Logger;

public class MessageHandler {
    Connection con;
    PreparedStatement ps;
    Logger logger = Logger.getLogger(MessageHandler.class.getName());

    public MessageHandler() throws ClassNotFoundException, SQLException {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection(Env.DB_URL, Env.DB_USERNAME,
Env.DB_PASSWORD);
    }

    public void insertMessage(String jsonData) {
        try {
            ps = con.prepareStatement("INSERT INTO messages (sender_id, receiver_id, content,
timestamp, read_receipt) VALUES (?,?,?,?,?)");

            ObjectMapper mapper = new ObjectMapper();
            JsonNode jsonNode = mapper.readTree(jsonData);

            String senderID = jsonNode.get("senderID").asText();
            String receiverID = jsonNode.get("receiverID").asText();
            String message = jsonNode.get("message").asText();
            String timestamp = jsonNode.get("timestamp").asText();
            String readReceipt = jsonNode.get("readReceipt").asText();

            ps.setString(1, senderID);
            ps.setString(2, receiverID);
            ps.setString(3, message);
            ps.setString(4, timestamp);
            ps.setString(5, readReceipt);

            ps.execute();
```

```
        } catch (SQLException | JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    public void updateLastMessage(String jsonData) {
        try {
            ObjectMapper mapper = new ObjectMapper();
            JsonNode jsonNode = mapper.readTree(jsonData);

            String sender = jsonNode.get("senderID").asText();
            String receiver = jsonNode.get("receiverID").asText();
            String message = jsonNode.get("message").asText();

            ps = con.prepareStatement("UPDATE friends SET last_msg = ? WHERE (x_id = ?
AND y_id = ?)");

            ps.setString(1, "You: " + message);
            ps.setString(2, sender);
            ps.setString(3, receiver);
            ps.execute();

            ps.setString(1, message);
            ps.setString(2, receiver);
            ps.setString(3, sender);
            ps.execute();

        } catch (SQLException | JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    public void close() throws SQLException {
        con.close();
    }
}
```

StatusHandler:

```
package com.talkwave.handler;

import com.talkwave.Env;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Logger;

public class StatusHandler {
    Connection con;
    PreparedStatement ps;
    ResultSet rs;
    Logger logger = Logger.getLogger(MessageHandler.class.getName());
    public StatusHandler() throws ClassNotFoundException, SQLException {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection(Env.DB_URL, Env.DB_USERNAME,
Env.DB_PASSWORD);
```

```
    }
    public void setUserStatus(String id, String status) throws SQLException  {
        ps = con.prepareStatement("UPDATE users SET status=? WHERE user_id = ?");

        ps.setString(1, status);
        ps.setString(2, id);
        ps.execute();
    }

    public void setMsgStatus(String senderID, String receiverID) throws SQLException  {
        ps = con.prepareStatement("UPDATE messages SET read_receipt = 'read' WHERE
sender_id = ? AND receiver_id = ? AND read_receipt = 'unread'");

        ps.setString(1, senderID);
        ps.setString(2, receiverID);
        ps.execute();
    }

    public List<String> getFriendsList(String id) throws SQLException {
        ps = con.prepareStatement("SELECT y_id FROM friends WHERE x_id = ?");
        ps.setString(1, id);
        rs = ps.executeQuery();
        List<String> list = new ArrayList<>();
        while (rs.next()) {
            list.add(rs.getString(1));
        }
        return list;
    }

    public void close() throws SQLException {
        con.close();
    }
}
```

Env:

```
package com.talkwave;

public class Env {
    public static String DB_URL = "jdbc:mysql://localhost:3306/talkwave";
    public static String DB_USERNAME = "oaish";
    public static String DB_PASSWORD = "Oaish@123";
}
```

Frontend Code:

auth.jsp:

```
<!DOCTYPE html>
<html lang="en">
<head>
<link href="assets/css/main.css" rel="stylesheet">
<link href="assets/css/authentication.css" rel="stylesheet">
<title>TalkWave | Auth</title>
<script src="assets/js/config.js"></script>
<script src="assets/js/authentication.js" defer></script>
<script>
const isLoggedIn = localStorage.getItem("auth")
```

```
if (isLoggedIn === "true") {
window.location.href = "index.jsp"
}
</script>
</head>
<body>

<div class="container">
<div class="card back-xl">
<div class="theme-card">
<img alt="client" src="assets/img/icon/client.svg">
</div>
<div class="card-xl-2">
<button class="btn login-btn login-btn-xl" type="button">Login</button>
<button class="btn login-btn signup-btn-xl" type="button">SignUp</button>
</div>
</div>


<div class="card back">
<div class="auth-text">
<h2>Login</h2>
<img alt="" class="back-btn" id="backBtn" src="assets/img/icon/back_btn.svg">
</div>
<div class="login">
<form action="/login" class="authentication" method="post">
<label>
        <input autocomplete="off" id="usr" name="username"  placeholder="Enter Username"
        spellcheck="false"
                type="text">
</label>
<div class="eye-cont">
        <label class="pass">
        <input autocomplete="off" id="pass" name="password" placeholder="Enter Password"
                spellcheck="false"
                type="password">
        </label>
        <img alt="eye" class="eye" src="assets/img/icon/eye.svg" style="display: none">
</div>
<button class="btn login-auth-btn auth-btn " type="button">Login</button>
</form>
</div>

<div class="signup active">
<form action="/signup" class="authentication" method="post">
<div class="first-page active">
        <label>
        <input autocomplete="off" id="signup-username" placeholder="Enter Username"
        spellcheck="false"
                type="text">
        </label>
        <div class="eye-cont">
        <label class="pass">
                <input autocomplete="off" id="signup-password" placeholder="Enter
                Password" spellcheck="false"
                type="password">
```

```
        </label>
        <img alt="eye" class="eye" src="assets/img/icon/eye.svg" style="display: none">
        </div>
        <div class="eye-cont">
        <label class="pass">
                <input autocomplete="off" id="signup-confirm" placeholder="Confirm
                Password" spellcheck="false"
                type="password">
        </label>
        <img alt="eye" class="eye" src="assets/img/icon/eye.svg" style="display: none">
        </div>
</div>
<div class="second-page">
        <div class="select-image">
        <img alt="" src="assets/img/icon/profile.svg">
        <div class="add-image">
                <img src="assets/img/icon/plus.svg" alt="">
                <label>
                <input id="file" accept=".jpg, .jpeg, .png, .gif" type="file">
                </label>
        </div>
        </div>
        <label>
        <input autocomplete="off" id="signup-profile" placeholder="Enter Profile Name"
        spellcheck="false"
                type="text">
        </label>
</div>
<button class="btn auth-btn" id="next-btn" type="button">Next</button>
</form>
</div>
</div>
<div class="card front">
<h1>Talk Wave</h1>
</div>
</div>

<dialog class="error-container">
<div class="error-msg">Error: Username or Password is Incorrect</div>
<img src="assets/img/icon/close.svg" alt="" class="close-btn">
</dialog>

</body>
</html>
```

index.jsp:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8"/>
   <meta
        name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
minimum-scale=1.0"
   />
   <meta http-equiv="X-UA-Compatible" content="ie=edge"/>
```

```
    <title>TalkWave</title>
    <link rel="stylesheet" href="assets/css/main.css"/>
    <link rel="stylesheet" href="assets/css/loader.css"/>
    <link rel="stylesheet" href="assets/css/qarq90.css"/>
    <link rel="stylesheet" href="assets/css/sidebar.css"/>
    <link rel="stylesheet" href="assets/css/users-container.css"/>
    <link rel="stylesheet" href="assets/css/chat-container.css"/>
    <script src="assets/js/config.js"></script>
    <script src="assets/js/initializer.js" defer></script>
    <script src="assets/js/qarq90.js" defer></script>
    <script src="assets/js/script.js" defer></script>
    <script src="assets/js/websocket.js" defer></script>
    <script>
        const isLoggedIn = localStorage.getItem("auth")

        if (!(isLoggedIn === "true")) {
            window.location.href = "auth.jsp"
        }

        let user = JSON.parse(localStorage.getItem("json"))

        sessionUser = {
            id: user.id,
            username: user.username,
            profileName: user.profileName,
            image: user.image,
        }
    </script>
</head>
<body>
<section class="loader">
    <div class="loader-logo">
        <h2>TalkWave</h2>
        <p>Chats are loading...</p>
    </div>
    <div class="loader-icon">
        <span class="dot dot-1"></span>
        <span class="dot dot-2"></span>
        <span class="dot dot-3"></span>
    </div>
</section>

<main>
    <div class="sidebar">
        <div class="sb-top">
            <div
                    class="chat-icon icon"
                    style="background-image: url('assets/img/icon/chat_room.svg')"
                    onclick="handleTabClick('chat-icon')"
            ></div>
            <div
                    class="user-icon icon"
                    style="background-image: url('assets/img/icon/user.svg')"
                    onclick="handleTabClick('user-icon')"
            ></div>
            <div class="pill"></div>
```

```
    </div>
    <div class="sb-bottom">
      <div
            class="settings-icon icon"
            style="background-image: url('assets/img/icon/settings.svg')"
            onclick="openProfile()"
      ></div>
    </div>
</div>

<div class="my-profile-card slide-hidden" id="myProfile">
    <div class="inner-profile inner-image-container">
      <img id="profile-img" src="" alt="pfp">
    </div>
    <div class="profile-info-div">
      <div class="inner-profile">
        <h4 id="profile-name"></h4>
      </div>
      <div class="inner-profile">
        <h4 id="user-name"></h4>
      </div>
    </div>
</div>

<div class="users-container">
    <div class="title-bar">
      <h2>Chats</h2>
      <div class="logout">
        <img src="assets/img/icon/logout.svg" alt=""/>
      </div>
    </div>
    <div class="users" id="myChats"></div>
    <div class="suggestions hidden" id="mySuggestions"></div>
</div>

<div class="chat-container">
    <div class="chat-mask">
      <img src="assets/img/icon/chat.svg" alt=""/>
    </div>
    <div class="chat-header">
      <div class="chat-profile">
        <img src="assets/img/Default.png" alt=""/>
        <div class="profile-info">
          <div class="chat-name"></div>
          <div class="chat-info"></div>
        </div>
      </div>
    </div>

    <div class="chat-body"></div>

    <div class="chat-down-btn down-btn-hidden">
      <img src="assets/img/icon/down.svg" alt=""/>
    </div>

    <div class="chat-input">
```

```
            <label for="text-box"></label>
            <input class="btn" id="text-box" placeholder="Text a message"/>
            <div
                    class="send-btn btn"
                    style="background-image: url('assets/img/icon/sent.svg')"
                    onclick="sendMsg()"
            ></div>
        </div>
    </div>
</main>
</body>
</html>
```

config.js:

```
const protocol = 'http://';
let domain = ""
if (window.location.hostname === "localhost")
    domain = 'localhost:8080/TalkWave_war_exploded'
else
    domain = '16.170.66.215:8080/TalkWave'

const apiGetFriendsURL = protocol + domain + '/api-get-friends'
const apiGetChatMsgURL = protocol + domain + '/api-get-chat-msg'
const apiAuthenticationURL = protocol + domain + '/api-authenticate'
const apiRegisterUserURL = protocol + domain + '/api-register-user'
const apiAddFriendURL = protocol + domain + '/api-add-friend'
const apiGetSuggestionsURL = protocol + domain + '/api-get-suggested'

let websocket = null
let isChatActive = false
let sessionUser = null
let receiver = null
let friends = {}
let messages = []
let profileB64 = ""
let suggestions = {}
let interval = null
```

qarq90.js:

```
function openChats() {
    myChats.className = "users";
    mySuggested.className = "suggestions hidden";
    titleBar.innerHTML = "Chats";

    fetchFriends().then(() => console.log("friends fetched successfully again"))
}

function openProfile() {
    myProfile.classList.toggle("slide-animate");
    myProfile.classList.toggle("slide-hidden");
    myProfileName.innerHTML = sessionUser.profileName;
    myUserName.innerHTML = "@" + sessionUser.username;
}

document.onclick = function (e) {
    if (!myProfile.contains(e.target) && !settingsBtn.contains(e.target)) {
```

```
        myProfile.classList.remove("slide-animate");
        myProfile.classList.add("slide-hidden");
    }
}

function openSuggested() {
    myChats.className = "users hidden";
    mySuggested.className = "suggestions";
    titleBar.innerHTML = "Add Friends";

    fetchSuggestions().then(() => console.log("suggestions fetched"))
}

function generateSuggestedCard(userData) {
    const suggestedCard = document.createElement("div");
    suggestedCard.className = "user-card";

    suggestedCard.innerHTML =
        `<div class="user-profile">
            <img src="${userData.image}" alt="">
        </div>
        <div class="user-info">
            <div class="user-name">${userData.profileName}</div>
            <div class="user-config"></div>
            <img src="assets/img/icon/add_user.svg" id="add-friend-btn" class="req-button"
alt="plus-icon"/>
        </div>`;

    const addFriendBtn = suggestedCard.querySelector("#add-friend-btn")
    addFriendBtn.onclick = () => {
        suggestedCard.parentNode.removeChild(suggestedCard);
        sendFriendRequest(userData.id).then(() => console.log("req sent"));
        websocket.send(`$add-friend:${userData.id}&${sessionUser.id}`)
    }

    const usersContainer = document.querySelector(".suggestions");
    usersContainer.appendChild(suggestedCard);
}

async function fetchSuggestions() {
    const url = apiGetSuggestionsURL + "?senderID=" + sessionUser.id
    const res = await fetch(url, {method: "GET"})
    suggestions = await res.json()

    if (suggestions.error !== undefined) {
        console.log(suggestions.error);
        return;
    }
    mySuggested.innerHTML = ""

    suggestions.forEach((suggestion) => {
        generateSuggestedCard(suggestion);
    });
}

async function sendFriendRequest(userId) {
```

```
    const url = apiAddFriendURL + "?senderID=" + sessionUser.id + "&receiverID=" + userId;
    const res = await fetch(url, {method: "GET"});

    if (res.status === 200)
        console.log("Friend request sent successfully");
    else
        console.error("Failed to send friend request");
}
```

script.js:

```
myProfileImg.src = sessionUser.image;

function sendMsg() {
    const value = textBox.value
    const time = getCurrentTime()
    textBox.value = ""
    if (value === "" || !isChatActive) return
    let receipt = ""

    if (receiver.canRead)
        receipt = "read";
    else
        receipt = "unread";

    putSenderMsg(value, time, receipt)

    const msgData = {
        senderID: sessionUser.id,
        receiverID: receiver.id,
        message: value,
        timestamp: time,
        readReceipt: receipt,
    }
    receiver.ref.userLastMsg.title = value
    receiver.ref.userLastMsg.textContent = "You: " + value

    receiver.unreadMessages = document.querySelectorAll('.read-receipt.unread')
    console.log(msgData)
    websocket && websocket.send(JSON.stringify(msgData));
}

function getCurrentTime() {
    const now = new Date();
    const options = { hour: '2-digit', minute: '2-digit', hour12: true };
    return now.toLocaleTimeString('en-US', options);
}

textBox.onkeyup = e => { if (e.keyCode === 13) sendMsg() }
function putSenderMsg(msg, time, readReceipt) {
    chatBody.innerHTML +=
        `<div class="row right">
            <div class="right-msg msg">
                <div class="content">${msg}</div>
                    <div class="msg-info">
                        <div class="read-receipt ${readReceipt}">
                            <img src="assets/img/icon/double_tick.svg" alt="">
```

```
                    </div>
                <div class="msg-time">${time}</div>
            </div>
        </div>
    </div>`
    chatBody.scrollTop = chatBody.scrollHeight
}
function putReceiverMsg(msg, time) {
    chatBody.innerHTML +=
        `<div class="row left">
            <div class="left-msg msg">
                <div class="content">${msg}</div>
                <div class="msg-info">
                    <div class="msg-time">${time}</div>
                </div>
            </div>
        </div>`
    chatBody.scrollTop = chatBody.scrollHeight
}
async function getChatHistory() {
    const url = apiGetChatMsgURL + "?senderID=" + sessionUser.id + "&receiverID=" +
receiver.id
    const res = await fetch(url, { method: 'GET' })
    messages = await res.json()
    messages.map((msg) => {
        if (msg.senderID.toString() === sessionUser.id) {
            putSenderMsg(msg.content, msg.timestamp, msg.readReceipt);
        } else {
            putReceiverMsg(msg.content, msg.timestamp);
        }
        return msg;
    });
    receiver.unreadMessages = document.querySelectorAll(".read-receipt.unread");
}
function startChat(user) {
    const chatProfilePic = document.querySelector(".chat-profile img")
    const chatName = document.querySelector(".chat-name")
    const chatInfo = document.querySelector(".chat-info")
    chatProfilePic.src = user.image === '' ? "assets/img/Default.png" : user.image
    chatName.textContent = user.profileName;
    chatBody.innerHTML = ""
    receiver = user
    if (!isChatActive)
        chatMask.style.display = "none";
    isChatActive = true;
    receiver.ref = {
        ...receiver.ref,
        chatInfo
    }
    receiver.ref.chatInfo.innerHTML = receiver.status
    receiver.ref.userLastMsg.style.color = "var(--secondary-text-color)";
    const friend = friends.find(f => f.canRead)
    if (friend)
        websocket.send(`$chat-inactive:${friend.id}&${sessionUser.id}`)
    websocket.send(`$chat-active:${receiver.id}&${sessionUser.id}`)
    interval = setInterval(() => {
```

```
        if (receiver.status === "online")
            clearInterval(interval)
        websocket.send(`$chat-active:${receiver.id}&${sessionUser.id}`)
    }, 1000)
    getChatHistory().then()
}
function generateUserCard(userData) {
    const userCard = document.createElement("div");
    userCard.className = "user-card";
    userCard.onclick = () => {
        friends.forEach(friend => friend.ref.userCard.className = "user-card")
        userCard.className += " user-card-active"
        startChat(userData);
    };
    userCard.innerHTML =
        `<div class="user-profile">
                <img src="${userData.image || 'assets/img/Default.png'}" alt="">
                <div class="status ${userData.status}"></div>
            </div>
            <div class="user-name">${userData.profileName}</div>
            <div class="user-last-msg" title="${userData.lastMsg}">${userData.lastMsg}</div>
            <div class="user-config"></div>`;

    const userStatus = userCard.querySelector(".status")
    const userLastMsg = userCard.querySelector(".user-last-msg")

    const usersContainer = document.querySelector(".users");
    usersContainer.appendChild(userCard);

    return {
        userCard,
        userLastMsg,
        userStatus
    }
}
async function fetchFriends() {
    const res = await fetch(apiGetFriendsURL + "?senderID=" + sessionUser.id, { method:
"GET" })
    friends = await res.json()

    if (friends.error !== undefined) {
        console.log(friends.error);
        return;
    }
    myChats.innerHTML = "";
    friends.forEach((friend) => {
        friend.ref = generateUserCard(friend);
    });
}
fetchFriends().then(() => document.querySelector(".loader").style.display = "none")

/* GENERAL PURPOSE FUNCTIONS */
const downBtn = document.querySelector(".chat-down-btn")

downBtn.onclick = function () {
    chatBody.scrollTop = chatBody.scrollHeight;
```

```
}

logoutBtn.onclick = function () {
    websocket.send("$offline:" + sessionUser.id)
    localStorage.setItem("auth", "false");
    window.location.href = "auth.jsp"
}

chatBody.onscroll = function () {
    const max = chatBody.scrollHeight
    const current = chatBody.scrollTop + chatBody.clientHeight;

    if (current <= max - 100)
        downBtn.className = "chat-down-btn"
    else
        downBtn.className = "chat-down-btn down-btn-hidden"
}

function handleTabClick(tabName) {
    const pill = document.querySelector(".pill");
    const userTab = document.querySelector(".user-icon");
    const chatTab = document.querySelector(".chat-icon");
    const animName = tabName === "user-icon" ? "pill-down" : "pill-up"
    pill.style.animation = animName + " 0.3s ease forwards"

    if (tabName === "user-icon") {
        userTab.classList.add("icon-active");
        chatTab.classList.remove("icon-active");
        openSuggested()
    } else if (tabName === "chat-icon") {
        userTab.classList.remove("icon-active");
        chatTab.classList.add("icon-active");
        openChats();
    }
}

handleTabClick("chat-user")
```

websocket.js:

```
websocket = new WebSocket(`ws://${domain}/websocket`)

setInterval(() => {
    if (!websocket)
        websocket = new WebSocket(`ws://${domain}/websocket`)
}, 100)

websocket.onopen = function (event) {
    console.log('WebSocket connection established')
    websocket.send(`$id:${sessionUser.id}`)
};

websocket.onmessage = function (event) {
    const data = event.data.toString()
    console.log("DATA", data)
    if (data.includes("RecipientNullException")) {
        console.log("Error: " + data)
```

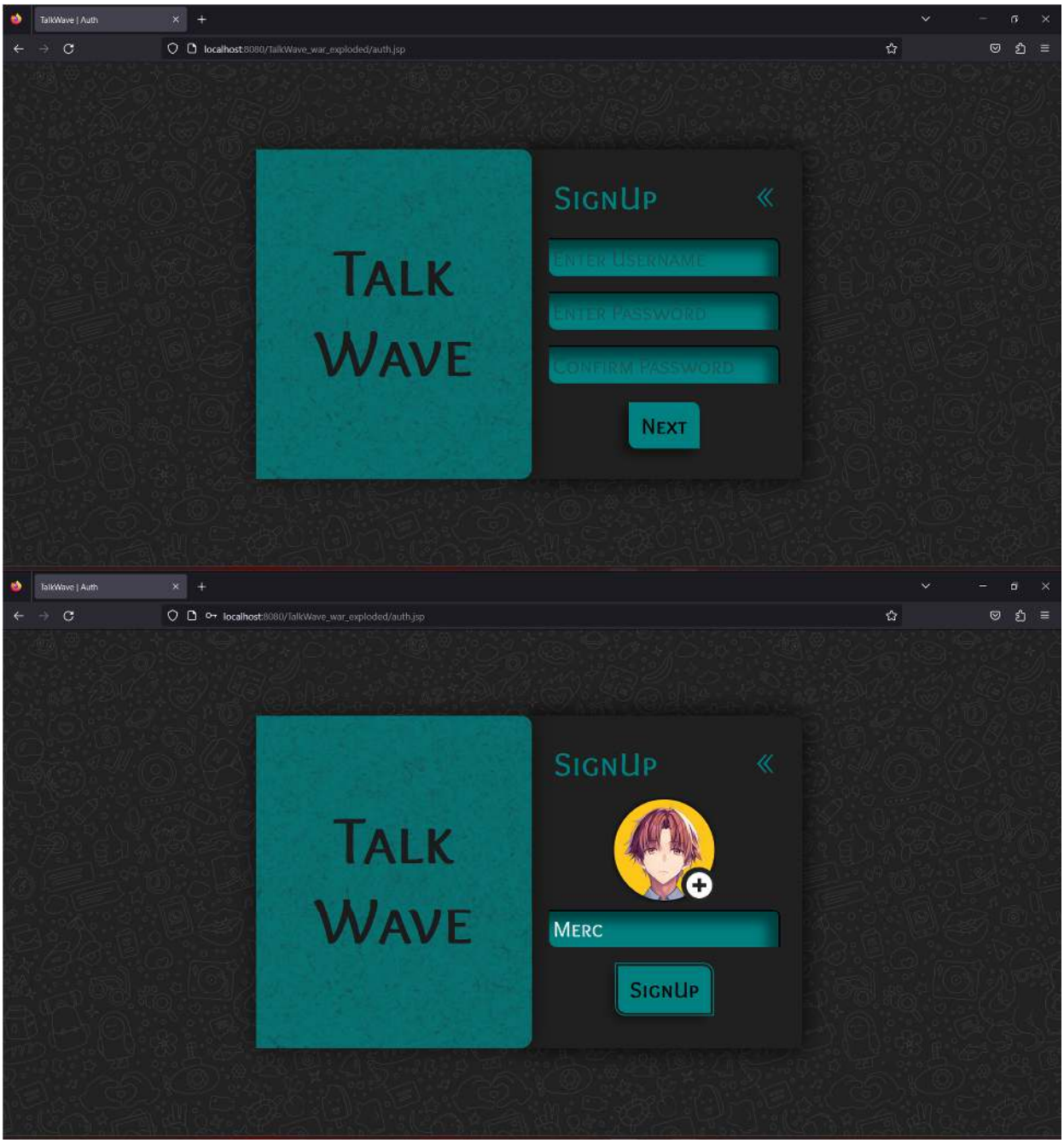
```
        return;
    }

    if (data.includes("$online:")) {
        const id = data.slice(8)
        const friend = friends.find(friend => friend.id === id)
        friend.ref.userStatus.className = "status online"
        friend.status = "online"
        if (receiver && receiver.id === friend.id) {
            receiver.status = "online"
            receiver.ref.chatInfo.innerHTML = "online"
        }
        console.log("$online:", id);
        return;
    } else if (data.includes("$offline:")) {
        const id = data.slice(9)
        const friend = friends.find(friend => friend.id === id)
        friend.ref.userStatus.className = "status offline"
        friend.status = "offline"
        friend.canRead = false

        if (receiver && receiver.id === friend.id) {
            receiver.status = "offline"
            receiver.ref.chatInfo.innerHTML = "offline"
            receiver.canRead = false
            interval = setInterval(() => {
                if (receiver.status === "online")
                    clearInterval(interval)
                websocket.send(`$chat-active:${receiver.id}&${sessionUser.id}`)
            }, 1000)
        }
        console.log("$offline:", id);
        return;
    }

    if (data.includes("$chat-active:")) {
        const ID = data.slice(13).split("&")
        console.log(data)
        const friend = friends.find(f => f.id === ID[1])
        friend.canRead = true;
        receiver && receiver.unreadMessages.forEach(msg => msg.className = "read-receipt
read")
        return;
    }

    if (data.includes("$chat-inactive:")) {
        const ID = data.slice(15).split("&")
        const friend = friends.find(friend => friend.id === ID[1])
        console.log(data, ID)
        friend.canRead = false;
        if (receiver && receiver.id === friend.id) {
            receiver.canRead = false;
        }
        return;
    }
```

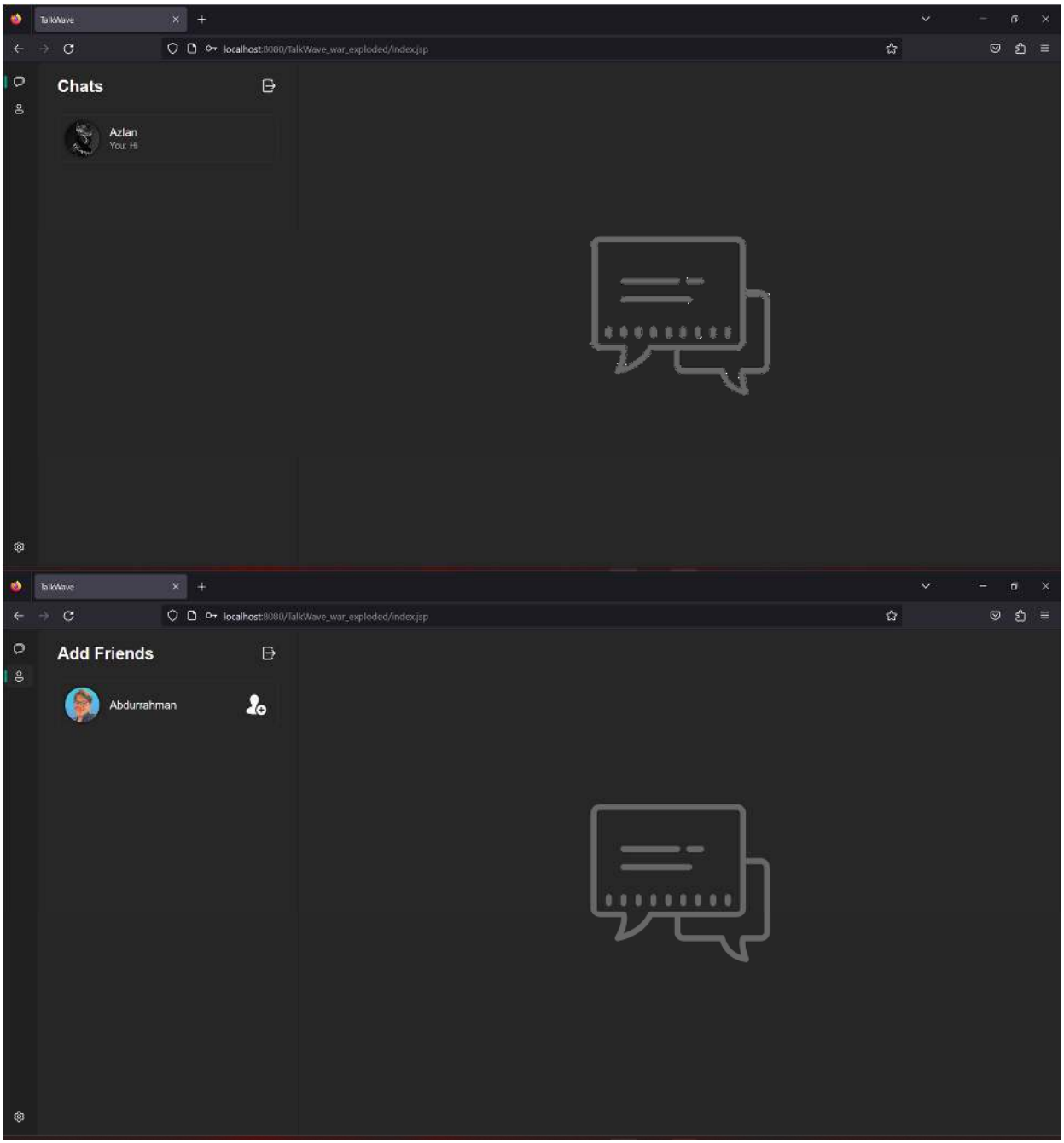
```
    if (data.includes("add-friend:")) {
        fetchFriends().then(()=> console.log("friends fetched successfully"))
        return;
    }

    try {
        const msg = JSON.parse(data)
        if (receiver && receiver.id === msg.senderID) {
            receiver.ref.userLastMsg.title = msg.message
            receiver.ref.userLastMsg.textContent = msg.message
            putReceiverMsg(msg.message, msg.timestamp)
        } else {
            const friend = friends.find(friend => friend.id === msg.senderID)
            friend.ref.userLastMsg.title = msg.message
            friend.ref.userLastMsg.textContent = msg.message
            friend.ref.userLastMsg.style.color = "var(--primary-color)"
        }
    } catch (e) {
        console.log(e)
    }
};

websocket.onclose = function (event) {
    if (event.wasClean)
        console.log('WebSocket connection closed cleanly, code=' + event.code)
    else
        console.log('WebSocket connection abruptly closed')
};

window.addEventListener('beforeunload', () => {
    websocket.send("$offline:" + sessionUser.id)
    websocket.close()
})
```

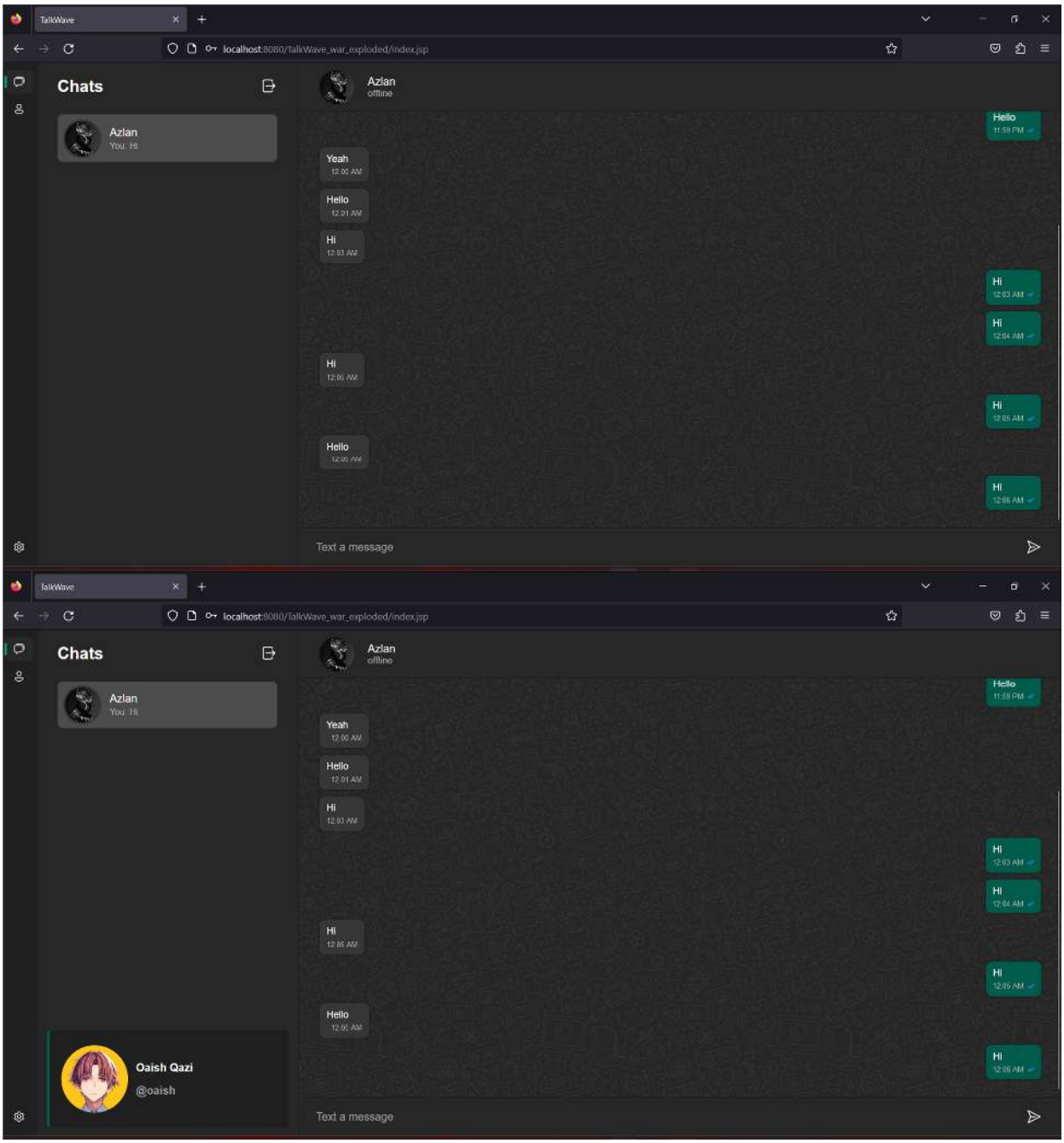
Output:

Annexure - II


TalkWave | Auth
localhost:8080/TalkWave_war_exploded/auth.jsp
TALK
WAVE
LOGIN
ENTER USERNAME
ENTER PASSWORD
LOGIN


TalkWave | Auth
localhost:8080/TalkWave_war_exploded/auth.jsp
TALK WAVE
SIGNUP
ENTER USERNAME
ENTER PASSWORD
CONFIRM PASSWORD
NEXT
TalkWave | Auth
localhost:8080/TalkWave_war_exploded/auth.jsp
TALK WAVE
SIGNUP
MERC
SIGNUP


TalkWave
localhost:8080/TalkWave_war_exploded/index.jsp
Chats
Azlan
You: Hi
TalkWave
localhost:8080/TalkWave_war_exploded/index.jsp
Add Friends
Abdurrahman


TalkWave
localhost:8080/TalkWave_war_exploded/index.jsp
Chats
Azlan
You: Hi
Azlan
offline
Hello
11:59 PM
Yeah
12:00 AM
Hello
12:01 AM
Hi
12:03 AM
Hi
12:03 AM
Hi
12:04 AM
Hi
12:05 AM
Hi
12:05 AM
Hello
12:06 AM
Hi
12:06 AM
Text a message
TalkWave
localhost:8080/TalkWave_war_exploded/index.jsp
Chats
Azlan
You: Hi
Azlan
offline
Hello
11:59 PM
Yeah
12:00 AM
Hello
12:01 AM
Hi
12:03 AM
Hi
12:03 AM
Hi
12:04 AM
Hi
12:05 AM
Hi
12:05 AM
Hello
12:06 AM
Hi
12:06 AM
Oaish Qazi
@oaish
Text a message

Annexure - I I

8.0 Skills Developed / Learning outcome of this Micro-Project

The following skills are developed:

1) **Identifying:** Identifying the problem and cause of problem in the area related and prepare project proposals before starting the project.
2) **Designing:** Designing of micro project with minimum required resources (low cost).
3) **Teamwork:** Learn to work in a team and boost individual confidence.
4) **Time Management:** Timely completion of micro project as scheduled.
5) **Problem-solving:** Develop good problem-solving skills.
6) **Technical Writing:** Preparing a report of the proposed plan and final report.
7) **Confidence:** Confidently, answer the questions asked about the project.
8) **Acknowledgement:** Acknowledge the help rendered by others in the success of the project.

9.0 Applications of this Micro-Project

1. Educational Tool
2. Teaching Resource
3. Personal Portfolio
4. Intranet Communication

(To be evaluated by the concerned teacher)

****

Annexure IV

Micro Project Evaluation Sheet

Name of Student: Abdurrahman Qureshi **Enrollment No.: 2100020112**
Name of Programme: Computer Engineering **Semester: Fifth**
Course Title: Advanced Java Programming (AJP) **Code: 22517**
Title of the Micro-Project: Chat Application
Course Outcomes Achieved: -

d) Develop java programs using networking concept.
e) Develop programs using database.
f) Develop program using Servlet

Sr No.	Characteristics to be assessed	Poor (Marks 1 - 3)	Average (Marks 4 - 5)	Good (Marks 6 - 8)	Excellent (Marks 9- 10)	Sub Total
(A) Process and Product Assessment (Convert above total marks out of 6 Marks)						
1	Relevance to the course					
2	Literature Review/information collection					
3	Completion of the Target as per project proposal					
4	Analysis of Data and representation					
5	Quality of Prototype/Model					
6	Report Preparation					
(B) Individual Presentation/ Viva (Convert above total marks out of 4 Marks)						
7	Presentation					
8	Viva					

(A) Process and Product Assessment (6 marks)	(B) Individual Presentation & viva (4 marks)	Total Marks 10

Comments/Suggestions about teamwork/leadership/inter-personal communication (if any)
.....
.....
.....

Name and designation of the Teacher Prof. Zaibunnisa Malik Ma'am

Dated Signature .....

Micro Project Evaluation Sheet

Name of Student: Qazi Mohd Oaish **Enrollment No.: 2100020108**

Name of Programme: Computer Engineering **Semester: Fifth**

Course Title: Advanced Java Programming (AJP) **Code: 22517**

Title of the Micro-Project: Chat Application

Course Outcomes Achieved: -

d) Develop java programs using networking concept.

e) Develop programs using database.

f) Develop program using Servlet

Sr No.	Characteristics to be assessed	Poor (Marks 1 - 3)	Average (Marks 4 - 5)	Good (Marks 6 - 8)	Excellent (Marks 9- 10)	Sub Total
(A) Process and Product Assessment (Convert above total marks out of 6 Marks)						
1	Relevance to the course					
2	Literature Review/information collection					
3	Completion of the Target as per project proposal					
4	Analysis of Data and representation					
5	Quality of Prototype/Model					
6	Report Preparation					
(B) Individual Presentation/ Viva (Convert above total marks out of 4 Marks)						
7	Presentation					
8	Viva					

(A) Process and Product Assessment (6 marks)	(B) Individual Presentation & viva (4 marks)	Total Marks 10

Comments/Suggestions about teamwork/leadership/inter-personal communication (if any)

.....

.....

.....

Name and designation of the Teacher Prof. Zaibunnisa Malik Ma'am

Dated Signature .....

Micro Project Evaluation Sheet

Name of Student: Shaikh Mohammed Hussain **Enrollment No.: 2200020625**
Name of Programme: Computer Engineering **Semester: Fifth**
Course Title: Advanced Java Programming (AJP) **Code: 22517**
Title of the Micro-Project: Chat Application
Course Outcomes Achieved: -

d) Develop java programs using networking concept.
e) Develop programs using database.
f) Develop program using Servlet

Sr No.	Characteristics to be assessed	Poor (Marks 1 - 3)	Average (Marks 4 - 5)	Good (Marks 6 - 8)	Excellent (Marks 9- 10)	Sub Total
(A) Process and Product Assessment (Convert above total marks out of 6 Marks)						
1	Relevance to the course					
2	Literature Review/information collection					
3	Completion of the Target as per project proposal					
4	Analysis of Data and representation					
5	Quality of Prototype/Model					
6	Report Preparation					
(B) Individual Presentation/ Viva (Convert above total marks out of 4 Marks)						
7	Presentation					
8	Viva					

(A) Process and Product Assessment (6 marks)	(B) Individual Presentation & viva (4 marks)	Total Marks 10

Comments/Suggestions about teamwork/leadership/inter-personal communication (if any)
.....
.....
.....

Name and designation of the Teacher Prof. Zaibunnisa Malik Ma'am

Dated Signature .....