HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.12.1/css/all.min.css"
      integrity="sha256-mmgLkCYLUQbXn0B1SRqzHar6dCnv9oZFPEC1g1cwlkk="
      crossorigin="anonymous"
    />
    <link rel="stylesheet" href="css/style.css" />
    <title>Chat App</title>
  </head>
  <body>
    <div class="join-container">
      <header class="join-header">
        <h1>Chat App</h1>
      </header>
      <main class="join-main">
        <form action="chat.html">
          <div class="form-control">
            <label for="username">Username</label>
            <input
              type="text"
```

```
            name="username"
            id="username"
            placeholder="Please enter username..."
            required
          />
        </div>
        <div class="form-control">
          <label for="room">Rooms</label>
          <select name="room" id="room">
            <option value="Business">Business</option>
            <option value="Courses">Courses</option>
            <option value="Games">Games</option>
          </select>
        </div>
        <button type="submit" class="btn">Join Chat</button>
      </form>
    </main>
  </div>
 </body>
</html>
```

CSS

```
@import url("https://fonts.googleapis.com/css?family=Roboto&display=swap");

:root {
```

```
  --dark-color-a: #006400;
  --dark-color-b: #75c44c;
  --light-color: #e6e9ff;
  --success-color: #4ca64c;
  --error-color: #ff3232;
}

* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: "Roboto", sans-serif;
  font-size: 16px;
  background: var(--light-color);
  margin: 20px;
}

ul {
  list-style: none;
}

a {
  text-decoration: none;
```

```
}

.btn {
  cursor: pointer;
  padding: 5px 15px;
  background: var(--light-color);
  color: var(--dark-color-a);
  border: 0;
  font-size: 17px;
}

/* Chat Page */

.chat-container {
  max-width: 1100px;
  background: #fff;
  margin: 30px auto;
  overflow: hidden;
}

.chat-header {
  background: var(--dark-color-a);
  color: #fff;
  border-top-left-radius: 5px;
  border-top-right-radius: 5px;
  padding: 15px;
```

```
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.chat-main {
  display: grid;
  grid-template-columns: 1fr 3fr;
}

.chat-sidebar {
  background: var(--dark-color-b);
  color: #fff;
  padding: 20px 20px 60px;
  overflow-y: scroll;
}

.chat-sidebar h2 {
  font-size: 20px;
  background: rgba(0, 0, 0, 0.1);
  padding: 10px;
  margin-bottom: 20px;
}

.chat-sidebar h3 {
  margin-bottom: 15px;
```

```
}

.chat-sidebar ul li {
  padding: 10px 0;
}

.chat-messages {
  padding: 30px;
  max-height: 500px;
  overflow-y: scroll;
}

.chat-messages .message {
  padding: 10px;
  margin-bottom: 15px;
  background-color: var(--light-color);
  border-radius: 5px;
}

.chat-messages .message .meta {
  font-size: 15px;
  font-weight: bold;
  color: var(--dark-color-b);
  opacity: 0.7;
  margin-bottom: 7px;
}
```

```css
.chat-messages .message .meta span {
  color: #777;
}

.chat-form-container {
  padding: 20px 30px;
  background-color: var(--dark-color-a);
}

.chat-form-container form {
  display: flex;
}

.chat-form-container input[type="text"] {
  font-size: 16px;
  padding: 5px;
  height: 40px;
  flex: 1;
}

/* Join Page */
.join-container {
  max-width: 500px;
  margin: 80px auto;
  color: #fff;
```

```
}

.join-header {
  text-align: center;
  padding: 20px;
  background: var(--dark-color-a);
  border-top-left-radius: 5px;
  border-top-right-radius: 5px;
}

.join-main {
  padding: 30px 40px;
  background: var(--dark-color-b);
}

.join-main p {
  margin-bottom: 20px;
}

.join-main .form-control {
  margin-bottom: 20px;
}

.join-main label {
  display: block;
  margin-bottom: 5px;
```

```
}

.join-main input[type="text"] {
  font-size: 16px;
  padding: 5px;
  height: 40px;
  width: 100%;
}

.join-main select {
  font-size: 16px;
  padding: 5px;
  height: 40px;
  width: 100%;
}

.join-main .btn {
  margin-top: 20px;
  width: 100%;
}

@media (max-width: 700px) {
  .chat-main {
    display: block;
  }
```

```
  .chat-sidebar {
    display: none;
  }
}
```

JAVASCRIPT

```
const path = require("path");
const http = require("http");
const express = require("express");
const socketio = require("socket.io");
const formatMessage = require("./utils/messages");
const {
  userJoin,
  getCurrentUser,
  userLeave,
  getRoomUsers,
} = require("./utils/users");

const app = express();
const server = http.createServer(app);
const io = socketio(server);

// Set static folder
app.use(express.static(path.join(__dirname, "public")));

const botName = "ChatCord Bot";
```

```
// Run when client connects
io.on("connection", (socket) => {
  socket.on("joinRoom", ({ username, room }) => {
    const user = userJoin(socket.id, username, room);

    socket.join(user.room);

    // Welcome current user
    socket.emit("message", formatMessage(botName, "Welcome to Chat App!"));

    // Broadcast when a user connects
    socket.broadcast
      .to(user.room)
      .emit(
        "message",
        formatMessage(botName, `${user.username} has joined the chat`)
      );

    // Send users and room info
    io.to(user.room).emit("roomUsers", {
      room: user.room,
      users: getRoomUsers(user.room),
    });
  });
```

```
  // Listen for chatMessage
  socket.on("chatMessage", (msg) => {
    const user = getCurrentUser(socket.id);

    io.to(user.room).emit("message", formatMessage(user.username, msg));
  });

  // Runs when client disconnects
  socket.on("disconnect", () => {
    const user = userLeave(socket.id);

    if (user) {
      io.to(user.room).emit(
        "message",
        formatMessage(botName, `${user.username} has left the chat`)
      );

      // Send users and room info
      io.to(user.room).emit("roomUsers", {
        room: user.room,
        users: getRoomUsers(user.room),
      });
    }
  });
});
```

```
const PORT = process.env.PORT || 3000;


server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```