Take-Home Coding Assessment: iOS Developer

Introduction

Congratulations on being selected for the take-home coding assessment for the freelance iOS developer position. This assessment is designed to evaluate your ability to design and implement chat screens for a mobile application, incorporating real-time messaging functionality and integrating with external APIs and services.

Objective

Your task is to design and implement two chat screens based on the provided Figma designs and integrate them with our backend chat server using the provided documentation on APIs and Socket.io.

Instructions

Design:

You will be provided with Figma designs for two chat screens:

1. a chat room list screen and
2. a chat conversation screen.

Review the designs carefully and ensure that your implementation closely matches the provided designs in terms of layout, styling, and functionality.

(Note: Figma Design Link - https://www.figma.com/file/CdlB8pB0KpLCIMKZJenHkO/Md.-Abdul-Gafur?type=design&node-id=0%3A1&mode=design&t=6TkOEuOMTXUM6VLd-1)

Functionality:

Implement the following features in the chat screens:

1. Display a list of chat rooms with the latest messages.
2. Allow users to select a chat room to view the conversation.
3. Display the chat conversation history.
4. Implement real-time messaging using Socket.io to receive new messages instantly.
5. Provide options to block or report a user from the chat screen.
6. Show confirmation dialog UI when blocking/reporting an user.
7. Display the QR code screen for each chat room.

Integration:

1. Use the provided documentation on APIs to pull chat history, chat rooms, and QR codes.
2. Follow the guidelines outlined in the documentation to connect and interact with our remote chat server using Socket.io.

Coding Standards:

1. Write clean, efficient, and maintainable code following iOS development best practices.
2. Ensure proper error handling and user feedback for all interactions.
3. Use appropriate design patterns and architectural principles to structure your code.

Submission:

1. You will have 7 Days to complete the assessment from the time you receive the instructions.
2. Submit your completed project as a zip file containing the Xcode project, source code, and any additional files or resources used.
3. Include a README file with instructions on how to run your project and any notes or considerations for the reviewers.
4. Lastly share your final app build via the following email address: ohodating@gmail.com

Evaluation Criteria:

1. Adherence to provided designs and specifications.
2. Implementation of required features and functionality.
3. Code quality, including readability, organization, and efficiency.
4. Integration with external APIs and Socket.io server.
5. Error handling and edge case scenarios.
6. Overall user experience and interface design.

Confidentiality:

Treat this assessment as confidential and refrain from sharing it with others.

Resources

1. Figma designs: Shared to your email.
2. Documentation on APIs: Check Page 4-9
3. Documentation on Socket.io integration: Check Page 9-10

Conclusion

We appreciate your time and effort in completing this coding assessment. If you have any questions or need clarification on any aspect of the assessment, feel free to reach out to ahmedthossain@outlook.com | (+880) 1796588950

We look forward to reviewing your submission and potentially working with you on our project.

Best regards,
Syed Rumman
CEO
Oho Dating Inc.

https://www.linkedin.com/in/rummansyed/
https://www.ohodating.com/

API Documentation

Get Chat Rooms

(**GET**): https://backend.ohodating.com/api/v1/chat/rooms

*(Note: The URL inside the variable "**`profile_photo`**" is for the thumbnail image of the sender, and it's without an extension. You are required to to append "**`.jpeg`**" to the URL.)*

Header:

KEY: Authorization

VALUE:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MTAzMTM2MjksIm
lhdCI6MTcwOTQ0OTYyOSwic3ViIjo0NjIsInVzZXJfdHlwZSI6Im9obyJ9.-1_K
9LobxzyUkc9qgxmaBommiOzTr7ewKUyZ15er79g
```

Returns a list of chat rooms for the provided user.

Response Body:


```
{
    "status": true,
    "data": {
        "data": [
            {
                "id": 42,
                "channel_name": "6289e5e8-b2c1-4c08-a4e6-35a5b28ce685",
                "status": "active",
                "participants": "187,229",
                "created_at": 1692731650,
                "blocked_by": null,
                "full_name": "Susan D. Fairchild",
                "profile_photo":
"https://oho-assets.s3.amazonaws.com/9031d29a95024562af463afb466397ae",
                "gender": "Female",
                "last_message": "This is a custom message ✨ "
            },
            {
                "id": 41,
                "channel_name": "796b0786-32c5-470a-82d8-7ae12c31dbc6",
                "status": "active",
                "participants": "187,225",
                "created_at": 1692473671,
                "blocked_by": null,
                "full_name": "Bessie Knight",
                "profile_photo":
"https://oho-assets.s3.amazonaws.com/342ae523b9df421ab5eb7e30a53cd405.jpg",
                "gender": "Female",
                "last_message": "Hi Tanzeer, Glad we matched. Your profile
looks amazing! Looking forward to our date at Binge Bar DC - 506 H St NE
LL, Washington, DC 20002 on Friday at 7:00 pm and getting to know you more
:)"
            }
        ]
    }
}
```

Get Chat History

(**GET**): https://backend.ohodating.com/api/v1/chat/history?chat_id=21

*(Note: The **"chat_id"** is the same as **"id"** of each chat room object that we get in the response of **Get Chat Rooms API -** https://backend.ohodating.com/api/v1/chat/rooms*
*See the highlighted section in the below example response**)***

*(Note: The URL inside the variable "**profile_photo**" is for the thumbnail image of the sender, and it's without an extension. You are required to to append "**.jpeg**" to the URL.)*

Example Response:


```
{
  "id":42,
  "channel_name":"6289e5e8-b2c1-4c08-a4e6-35a5b28ce685",
  "status":"active",
  "participants":"187,229",
  "created_at":1692731650,
  "blocked_by":null,
  "full_name":"Susan D. Fairchild",

"profile_photo":"https://oho-assets.s3.amazonaws.com/9031d29a95024562af463afb
466397ae",
  "gender":"Female",
  "last_message":"This is a custom message ✨ "
}
```


Header:

KEY: Authorization

VALUE:


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MTAzMTM2MjksIm
lhdCI6MTcwOTQ0OTYyOSwic3ViIjo0NjIsInVzZXJfdHlwZSI6Im9obyJ9.-1_K
9LobxzyUkc9qgxmaBommiOzTr7ewKUyZ15er79g
```

Returns a list of chat messages between users belonging to the chat room with the provided chat id. You first have to call the **chat rooms api** to get the **chat id**.

Query Parameter:

KEY: chat_id

VALUE: [*value of the chat_id*]

Response Body:


```
{
    "status": true,
    "data": {
        "data": [
            {
                "id": 523,
                "body": "Hi Tanzeer, Glad we matched. Your profile looks
amazing! Looking forward to our date at Binge Bar DC - 506 H St NE LL,
Washington, DC 20002 on Friday at 7:00 pm and getting to know you more :)",
                "sender": 225,
                "chat_id": 41,
                "chat_type": "delegate",
                "created_at": 1692473671
            },
            {
                "id": 522,
                "body": "Hi Bessie, Glad we matched. Your profile looks
amazing! Looking forward to our date at Binge Bar DC - 506 H St NE LL,
Washington, DC 20002 on Friday at 7:00 pm and getting to know you more :)",
                "sender": 187,
                "chat_id": 41,
                "chat_type": "delegate",
                "created_at": 1692473671
            }
        ],
        "has_next": false,
        "has_prev": false
    }
}
```

Get Chat QR Code

(**GET**): https://backend.ohodating.com/api/v1/match/get_qr_code?chat_id=23

*(Note: The **"chat_id"** is the same as **"id"** of each chat room object that we get in the response of **Get Chat Rooms API -** https://backend.ohodating.com/api/v1/chat/rooms*
See the highlighted section in the below example response)

Example Response:


```
{
  "id":42,
  "channel_name":"6289e5e8-b2c1-4c08-a4e6-35a5b28ce685",
  "status":"active",
  "participants":"187,229",
  "created_at":1692731650,
  "blocked_by":null,
  "full_name":"Susan D. Fairchild",

"profile_photo":"https://oho-assets.s3.amazonaws.com/9031d29a95024562af463afb466397ae",
  "gender":"Female",
  "last_message":"This is a custom message ✨ "
}
```


Header:

KEY: Authorization

VALUE:


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MTAzMTM2MjksImlhdCI6MTcwOTQ0OTYyOSwic3ViIjo0NjIsInVzZXJfdHlwZSI6Im9obyJ9.-1_K9LobxzyUkc9qgxmaBommiOzTr7ewKUyZ15er79g
```


Get QR code for a date

Query Parameter:

KEY: chat_id

VALUE: [*value of the chat_id*]

Response Body:

```
{
    "status": true,
    "data": {
        "qr_code":
"https://oho-match-qr-codes.s3.amazonaws.com/464_129.jpg",
        "match_id": 129
    }
}
```

Chat Using Socket.io

you need to initialize the socket instance with the token below:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MDg4ODQ5NjcsImlhdCI6MTcwODc5
ODU2Nywic3ViIjoxODcsInVzZXJfdHlwZSI6Im9obyJ9.T-5CVp4sL74U_6nK6EJmMoJ54RDgUUs7
vD0HYtucrT0
```

Sample JS Code *(You need to convert it to Swift Code)*

```
const URL = "https://chat.backend.ohodating.com";
const socket = io(URL, { auth: { token: tokenFromAbove }, autoConnect: false
});
```

listen on the following events:

- “connect”
- “error”
- chat_room

*(Note: The **‘chat_room’** is the same as **“channel_name”** of each chat room object that we get in the response of **Get Chat Rooms API -** https://backend.ohodating.com/api/v1/chat/rooms See the highlighted section in the below example response)*

Example Response:

```
{
  "id":42,
```

```
  "channel_name":"6289e5e8-b2c1-4c08-a4e6-35a5b28ce685",
  "status":"active",
  "participants":"187,229",
  "created_at":1692731650,
  "blocked_by":null,
  "full_name":"Susan D. Fairchild",

"profile_photo":"https://oho-assets.s3.amazonaws.com/9031d29a95024562af463afb
466397ae",
  "gender":"Female",
  "last_message":"This is a custom message ✨ "
}
```


```
socket.on("connect", () => {
  console.log("connected");
});

socket.on("error", (message) => {
  console.log(message);
});

socket.on(chat_room, (message) => {
  console.log(message);
});
```

emit the following events (check below for sample payload to send for each event):

- “private message”
- chat_room

```
socket.emit("private message", {roomName: chat_room, chat_id: id});

socket.emit(chat_room, plaintextmessage);
```