Documentation for Business Chat Project

Title Page

Project Name: Business Chat

Team Members: Gensik Rubio

Date: 12/04/2024

Instructor: Dr. Habeeb Olufowobi

Customer-UI-URL: https://customer-chat-ui-69178.web.app/

Representative-UI-URL: https://rep-inbox-dashboard-88712.web.app/

Representative email: example@hotmail.com

Representative Password = password2

Table of Contents

Introduction

The Business Chat System is a web-based communication platform that facilitates seamless, real-time interactions between customers and support representatives. The system integrates an automated chatbot for initial user engagement, escalating complex queries to human representatives through an intuitive inbox dashboard. By leveraging Google Cloud services, the project demonstrates the creation of a scalable, efficient, and user-friendly communication solution.

The system architecture comprises multiple cutting-edge components. The Customer Chat Interface and the Representative Inbox Dashboard are hosted on Firebase Hosting, ensuring fast and secure delivery of front-end applications. The backend server, built with Node.js, is deployed on Google App Engine, providing robust scalability and easy maintenance. Socket.IO enables real-time communication between the front-end and backend server. For data persistence, Firestore is used to store escalated chats and representative tokens, ensuring reliable retrieval and notification handling. Notifications are powered by Firebase Cloud Messaging (FCM), which sends alerts to representatives about new escalated chats.

This project not only highlights the potential of Google Cloud technologies in building dynamic web services but also provides a robust foundation for improving customer support systems. By combining automation with human intervention, the Business Chat System aims to enhance user satisfaction while streamlining support workflows, demonstrating its value as an efficient and scalable communication platform.

Project Requirements

The Business Chat System is designed to meet the following functional and technical requirements:

Functional Requirements:

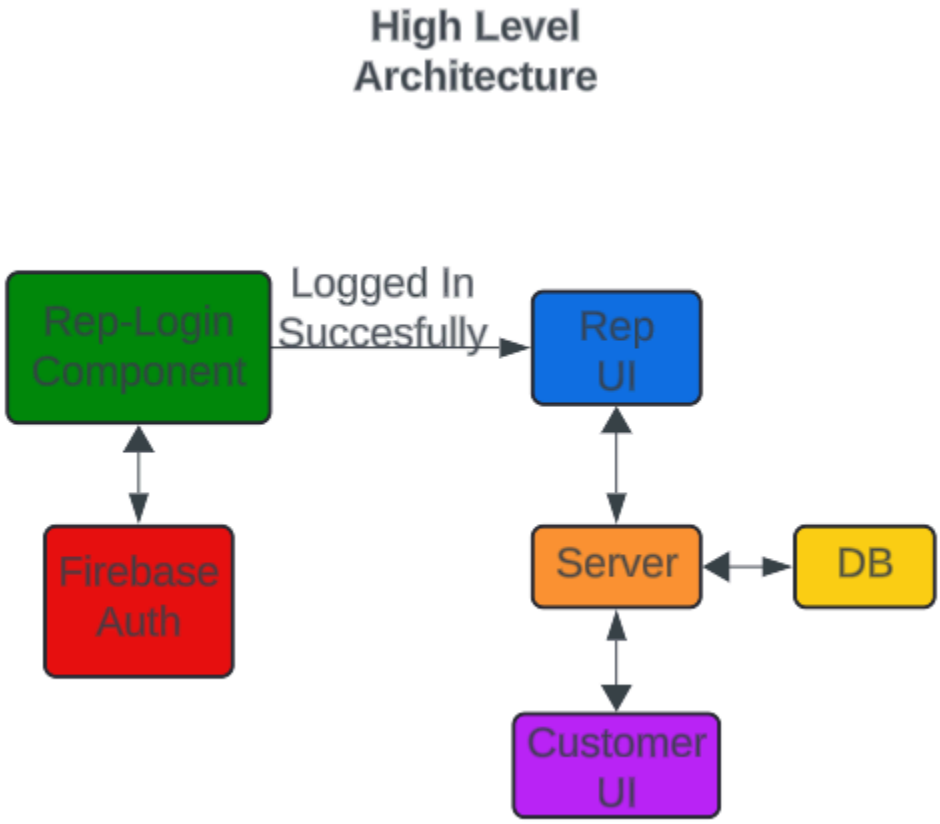
1. **Customer Chat Interface:**
    - Customers must be able to initiate a chat through a web-based interface.
    - An automated chatbot will handle initial interactions by providing predefined responses to common queries.
    - Customers can escalate a chat to a human representative when the chatbot cannot address their concerns.
2. **Representative Inbox Dashboard:**
    - Logged-in support representatives will have access to an inbox displaying escalated chats.
    - Representatives must be able to respond to customer messages in real-time.
    - Each representative will receive notifications for new escalated chats via push notifications.
3. **Real-Time Communication:**
    - Both the customer and representative must experience real-time message delivery during active sessions.
4. **Chat Escalation Process:**
    - When a customer escalates a chat, the system must:
    - Save the chat transcript in Firebase Firestore for future reference.
    - Trigger a push notification via Firebase Cloud Messaging (FCM) to all logged-in representatives.
5. **Data Persistence:**
    - All escalated chats must be stored securely in Firestore.
    - Representative tokens linked to their user IDs will also be stored in Firestore to manage notifications.

System Architecture

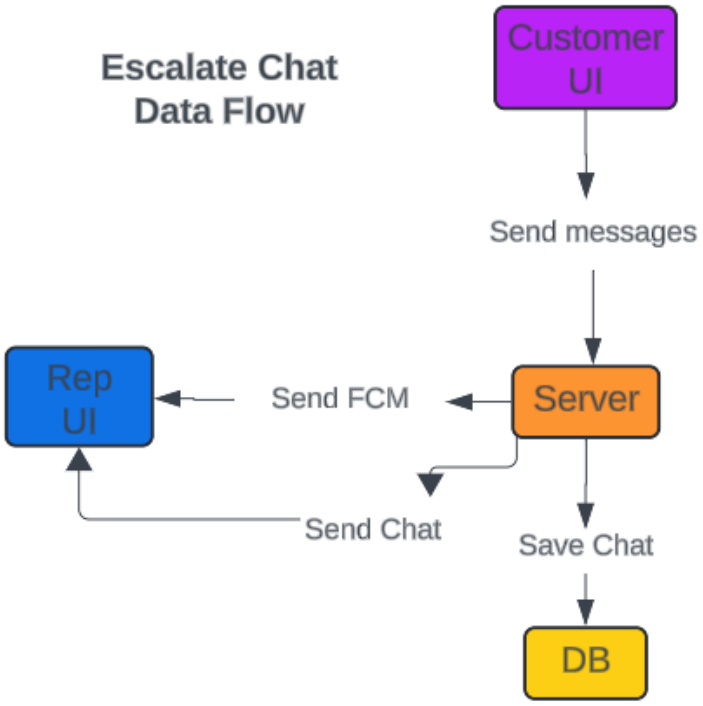
Overview

The system is built on a client-server model:

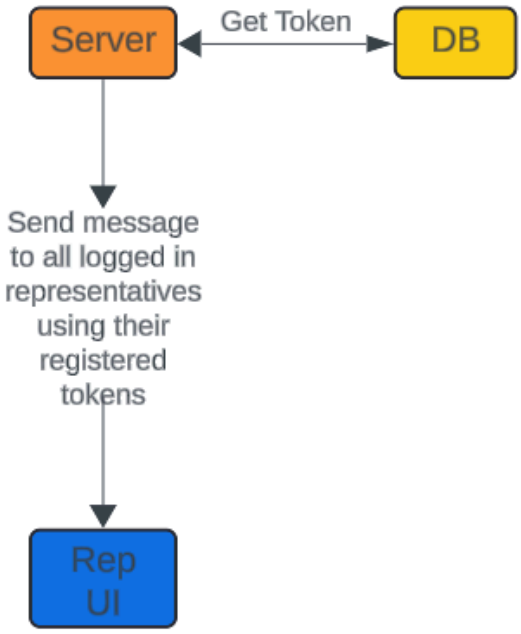
- **Frontend (React):** Hosted on Firebase Hosting, it provides interfaces for customers and representatives.
- **Backend (Node.js):** Deployed on Google App Engine, it manages chat workflows, database interactions, and notification triggers.
- **Database (Firestore):** Stores persistent chat data and supports real-time updates.
- **Notification Service:** Utilizes FCM for representative alerts.

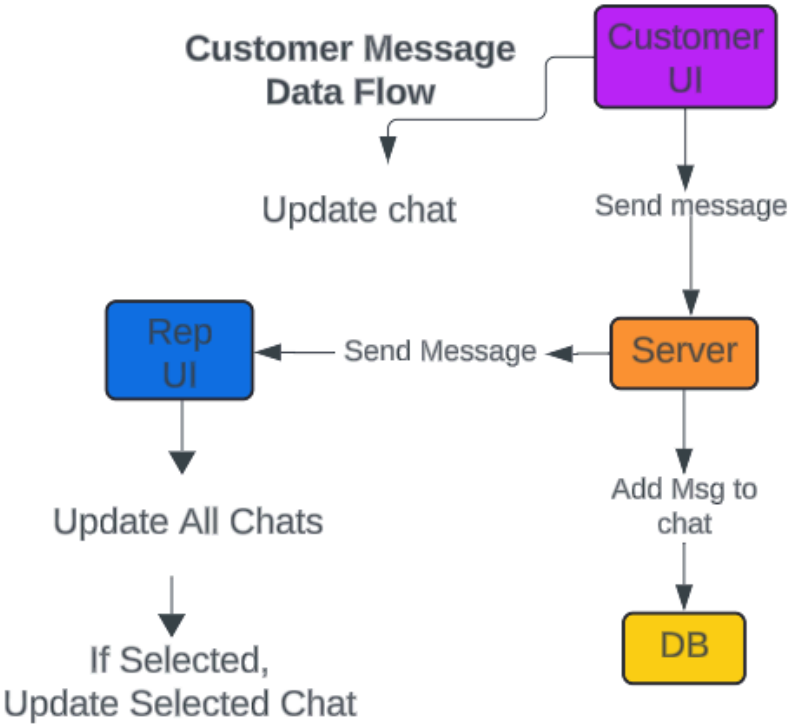


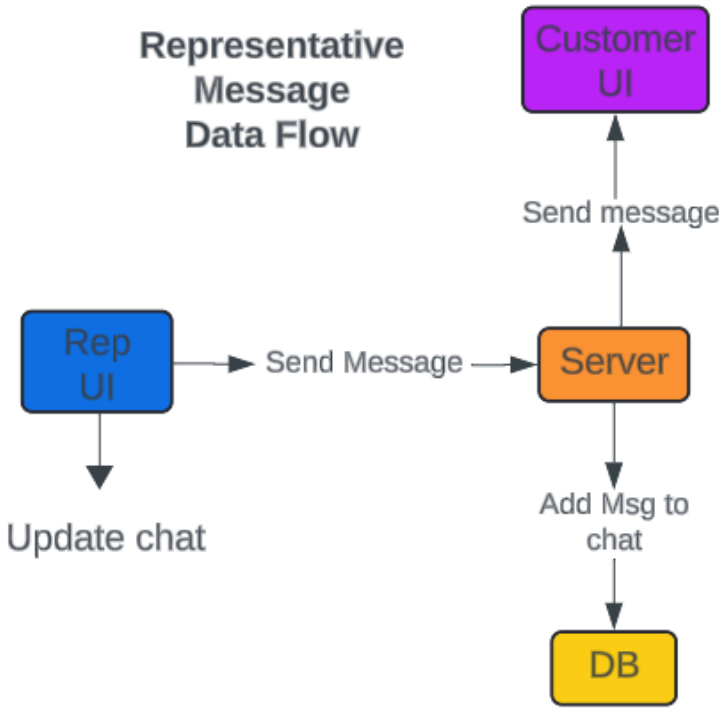
Escalate Chat Data Flow



Firebase Cloud Messaging Notifications Data Flow




Customer Message
Data Flow
Customer UI
Update chat
Send message
Rep UI
Send Message
Server
Update All Chats
Add Msg to chat
If Selected,
Update Selected Chat
DB


Representative
Message
Data Flow
Customer UI
Send message
Rep UI
Send Message
Server
Update chat
Add Msg to chat
DB

Google Cloud Components and Their Interactions

This project leverages several Google Cloud components to create a scalable, efficient, and cohesive Business Chat System. Below is a detailed explanation of the components used and how they interact with each other:

1. Firebase Hosting

- **Purpose:** Firebase Hosting is used to deploy and host the frontend React applications.
- **Applications Hosted:**
    - **Customer UI:** Provides customers with an interface to interact with the chatbot, escalate chats, and communicate with representatives.
    - **Representative UI:** Offers representatives a dashboard to view and respond to escalated chats.
- **Interaction:** Firebase Hosting ensures seamless delivery of the web apps, enabling them to communicate with the backend server hosted on Google App Engine.

2. Firebase Cloud Messaging (FCM)

- **Purpose:** FCM is utilized to send push notifications to all logged-in representatives when a chat is escalated by a customer.
- **Interaction:**
    - The Node.js backend server triggers FCM when a chat is escalated.
    - Tokens for logged-in representatives are retrieved from Firestore to identify the recipients.
    - Notifications alert representatives to take immediate action.

3. Firebase Firestore

- **Purpose:** Firestore serves as the persistent data storage layer for the system.
- **Data Stored:**
    - Escalated chat transcripts for reference and accountability.
    - Tokens associated with representative user IDs for sending targeted notifications.
- **Interaction:** The backend server reads and writes data to Firestore to manage chat histories and FCM tokens. Both frontend apps fetch chat data from Firestore for display and interaction.

4. Google App Engine

- **Purpose:** App Engine hosts the backend Node.js server that facilitates the interaction between the frontend applications, Firestore, and FCM.
- **Key Responsibilities:**
    - Processes and routes chat data between the Customer UI and Representative UI.

    - Saves escalated chats to Firestore.
    - Sends FCM notifications for chat escalations.
- **Interaction:** The App Engine acts as the central hub for backend operations, enabling secure and efficient communication across system components.

5. Firebase Authentication

- **Purpose:** Firebase Auth is implemented to handle representative logins securely.
- **Features:**
    - Authentication ensures that only authorized users access the Representative UI.
    - Each representative is assigned a unique user ID, which is linked to their FCM token in Firestore.
- **Interaction:** Firebase Auth integrates seamlessly with the Representative UI and backend server to enforce user access control.

Component Interactions Summary

1. **Frontend Apps (Customer UI and Representative UI):**
    - Hosted on Firebase Hosting and interact with the backend server via API calls and real-time sockets using Socket.IO.
2. **Backend Server (Node.js):**
    - Hosted on Google App Engine and acts as the intermediary for all communication and data processing.
    - Triggers FCM notifications and manages Firestore operations.
3. **Firestore Database:**
    - Stores chat transcripts and FCM tokens for representatives.
    - Accessed by the backend and Representative UI for data retrieval and updates.
4. **Firebase Authentication:**
    - Secures access to the Representative UI, ensuring only verified representatives can interact with the system.

This combination of Google Cloud components ensures a robust, scalable, and efficient system capable of meeting the demands of real-time customer support interactions.

Deployment Process

Frontend Deployment:

- Built using React.

```
PS C:\Users\gensikr\Desktop\school\fall2024\cse4333\project1\chat_project\rep-inbox-dashboard> npm run build

> rep-inbox@0.1.0 build
> react-scripts build

Creating an optimized production build...
```

- Deployed to Firebase Hosting via the Firebase CLI.

```
i  hosting[customer-chat-ui-69178]: found 14 files in customer-chat-ui/build
PS C:\Users\gensikr\Desktop\school\fall2024\cse4333\project1\chat_project> firebase deploy --only hosting

=== Deploying to 'business-chat-88712'...

i  deploying hosting
i  hosting[customer-chat-ui-69178]: beginning deploy...
i  hosting[customer-chat-ui-69178]: found 14 files in customer-chat-ui/build
+  hosting[customer-chat-ui-69178]: file upload complete
i  hosting[rep-inbox-dashboard-88712]: beginning deploy...
i  hosting[rep-inbox-dashboard-88712]: found 15 files in rep-inbox-dashboard/build
+  hosting[rep-inbox-dashboard-88712]: file upload complete
i  hosting[customer-chat-ui-69178]: finalizing version...
i  hosting[rep-inbox-dashboard-88712]: finalizing version...
+  hosting[rep-inbox-dashboard-88712]: version finalized
i  hosting[rep-inbox-dashboard-88712]: releasing new version...
+  hosting[customer-chat-ui-69178]: version finalized
i  hosting[customer-chat-ui-69178]: releasing new version...
+  hosting[rep-inbox-dashboard-88712]: release complete
+  hosting[customer-chat-ui-69178]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/business-chat-88712/overview
Hosting URL: https://customer-chat-ui-69178.web.app
Hosting URL: https://rep-inbox-dashboard-88712.web.app
```

Backend Deployment:

- Node.js server deployed to Google App Engine.

app.yaml

```yaml
runtime: nodejs20

handlers:
  # Serve the firebase_credentials.json file
  - url: /firebase_key/firebase_credentials.json
    static_files: firebase_key/firebase_credentials.json
    upload: firebase_key/firebase_credentials.json

# Optional configuration
automatic_scaling:
  target_cpu_utilization: 0.65
  min_instances: 1
  max_instances: 10
```

```
PS C:\Users\gensikr\Desktop\school\fall2024\cse4333\project1\chat_project\server> gcloud app deploy
Services to deploy:

descriptor:                  [C:\Users\gensikr\Desktop\school\fall2024\cse4333\project1\chat_project\server\app.yaml]
source:                      [C:\Users\gensikr\Desktop\school\fall2024\cse4333\project1\chat_project\server]
target project:              [chat-server-443722]
target service:              [default]
target version:              [20241204t234954]
target url:                  [http://chat-server-443722.uc.r.appspot.com]
target service account:      [chat-server-443722@appspot.gserviceaccount.com]
```

Database Setup:

- Firestore database initialized with collections for escalated_chats and rep_tokens

Notification Configuration:

- FCM integrated with backend for push notification triggers.

Conclusion

In conclusion, the Business Chat System successfully demonstrates how modern cloud technologies can be leveraged to create an efficient, scalable, and user-friendly communication platform. By integrating an automated chatbot with a human representative interface, the system bridges the gap between automated interactions and personalized customer service. This thoughtful blend of functionality ensures that users receive immediate assistance while offering a seamless escalation process for complex queries.

The use of Google Cloud components has been instrumental in achieving the system's objectives. Firebase Hosting and Cloud Messaging enhance the responsiveness of the front-end applications, while Firestore ensures reliable and persistent data storage. Google App Engine provides a scalable and easily maintainable backend infrastructure, while Firebase Authentication ensures secure access for representatives. Together, these technologies create a cohesive system that emphasizes speed, security, and real-time interactivity.

Looking ahead, this project offers a robust foundation for further enhancements and scalability. Features such as advanced analytics, multi-language support, or integration with third-party CRM tools could be implemented to broaden its capabilities. By demonstrating how automation and human support can work together effectively, the Business Chat System sets a strong example of how cloud-based solutions can transform customer service workflows.