CS306 – Project Phase III

Alper Çamlı 30858

Ege Yardımcı 31024

Emre Berk Hamarat 31188

Korhan Erdoğdu 30838

Report on Real-Time Support Page Implementation with Firebase

Overview

This project aimed to implement a real-time support page for a website using Firebase. It facilitates seamless communication between users or guests and the admin, with privacy and efficiency ensured by Firebase's real-time database features and security rules.

Implementation Details

1. User/Guest Support Page

The support page allows users or guests to:

- Enter their name in an input field.
- Select the subject of their issue from a dropdown menu.
- Enter a chat message and view their conversation history.
- Automatically display the date and time of their messages.

2. Admin Page

The admin interface provides the ability to:

- View all messages from users or guests in one consolidated view.
- Respond to messages in real-time.
- Access messages securely through Firebase security rules.

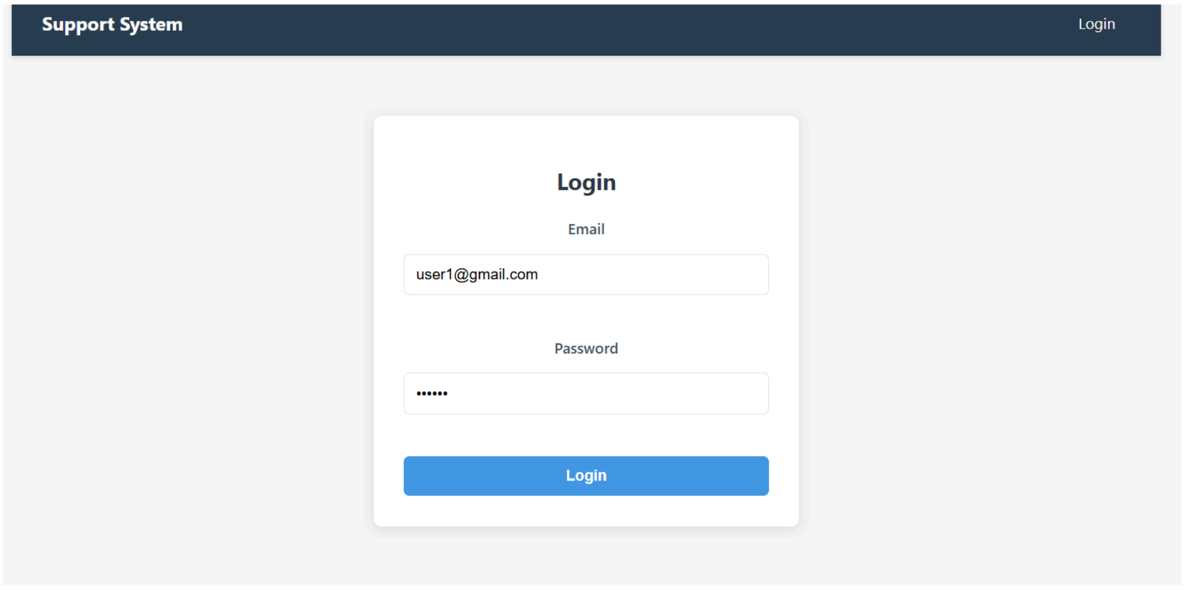
3. Privacy and Security

- Firebase security rules ensure each user or guest can only access their messages.
- The admin has permission to view and respond to all messages.
- Real-time updates ensure instant visibility of new messages or responses.

User1 Login

The workflow begins with **User1** accessing the login screen. This screen is simple yet functional, requiring the user to provide their name to proceed. The process involves:

1. **Name Input:** The name serves as a unique identifier, allowing Firebase to create a distinct session for User1. This ensures their messages are securely stored and easily retrieved without interfering with messages from other users.
2. **Session Initialization:** When User1 submits their name, Firebase generates a unique session ID in the database. This ID is essential for maintaining user-specific privacy and for allowing the admin to categorize and respond to tickets efficiently.
3. **Guided Transition:** Once the login step is completed, User1 is seamlessly redirected to the **ticket screen**, where they can begin interacting with the support system.



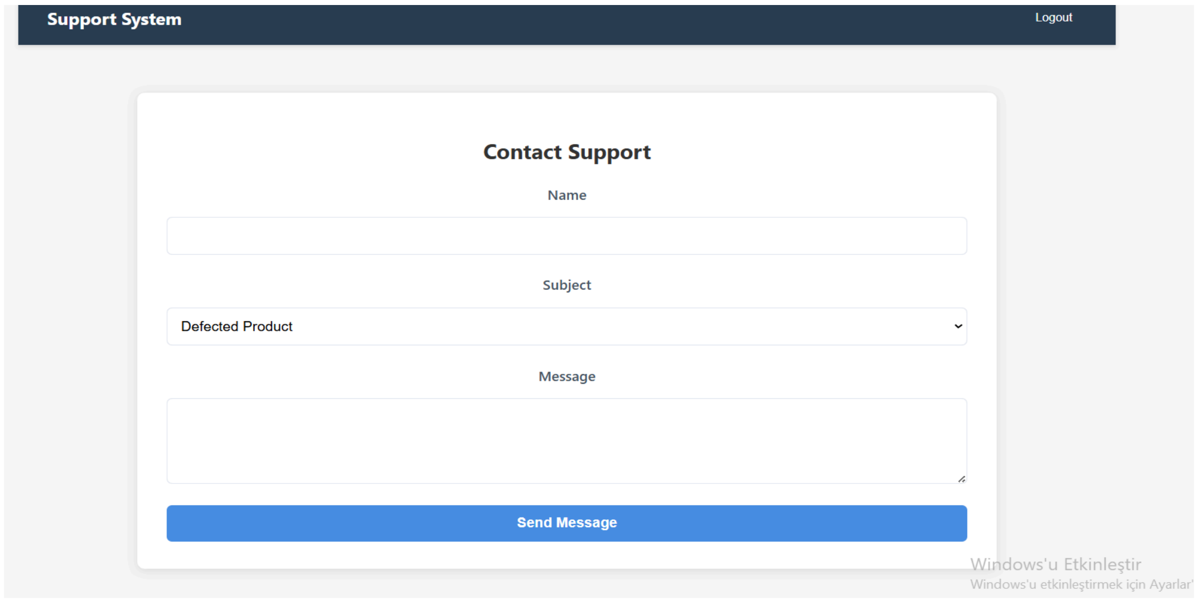
user1@gmail.com

••••••

Ticket Screen

The ticket screen is the primary interface where users interact with the support system. It includes:

1. **Pre-filled Name Input:** The name entered during login is automatically displayed, maintaining continuity and reducing redundant steps.
2. **Subject Dropdown:** Users categorize their issue by selecting a predefined subject, such as “Defective Product” or “Lost Product,” ensuring the admin quickly understands the ticket's context.
3. **Chat Interface:** Users can type their messages in the input box and view all previous messages in the conversation thread, keeping communication transparent.
4. **Real-Time Feedback:** Messages sent by the user appear instantly with timestamps, confirming successful delivery and providing a seamless experience.



Selecting Subject

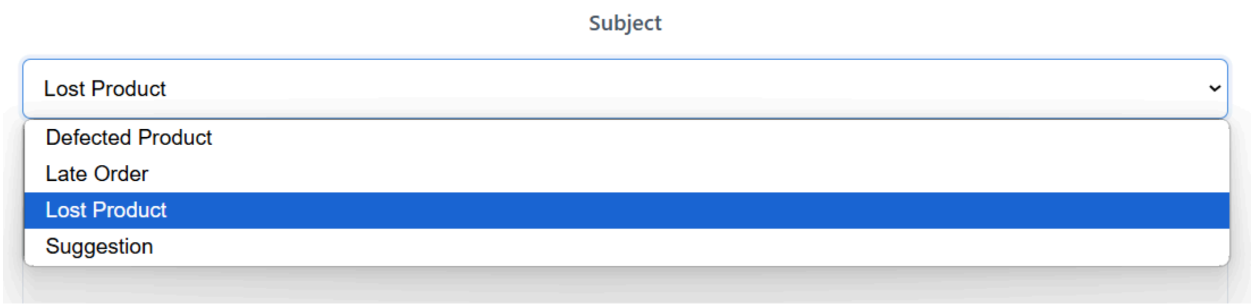
Selecting a subject is a vital step in the user workflow, as it provides context for the admin to better understand the nature of the ticket. The dropdown menu offers predefined categories, such as:

- **Defected Product:** This category is for users reporting issues with products that arrived damaged, malfunctioning, or not meeting expectations.
- **Late Order:** Users select this category to inquire about delays in the delivery of their orders.
- **Lost Product:** This category addresses concerns where users suspect their product is lost during shipment or delivery.
- **Suggestion:** Users select this category to provide feedback or ideas for improving products or services.

By choosing the most relevant subject, User1 ensures that:

- Their issue is categorized correctly in the admin dashboard.
- The admin can prioritize and address the ticket more efficiently, especially when multiple tickets are queued.

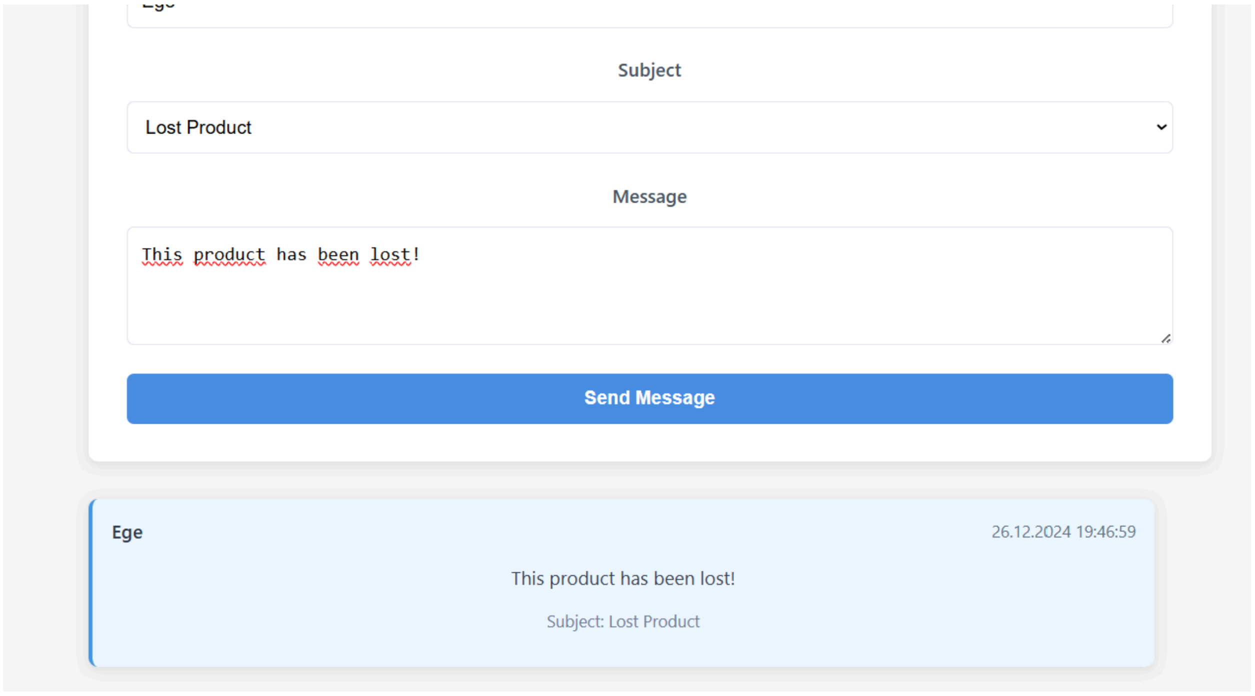
This step demonstrates the system's user-centric design, streamlining the communication process for both users and admins.



User1 Sends Ticket

After entering their name, selecting a subject, and typing a message, User1 clicks the "Send Message" button. This action triggers several processes:

1. **Message Submission:** The message is sent to Firebase's Realtime Database, securely linked to User1's unique session ID.
2. **Real-Time Update:** The message appears instantly in the chat interface with a timestamp, confirming successful delivery.
3. **Database Entry:** Firebase stores the message with metadata (name, subject, timestamp, and content) for organized admin access.
4. **Visual Confirmation:** The interface updates dynamically, showing the message in the chat thread for clear conversation tracking.


Subject

Lost Product

Message

This product has been lost!

Ege 26.12.2024 19:46:59

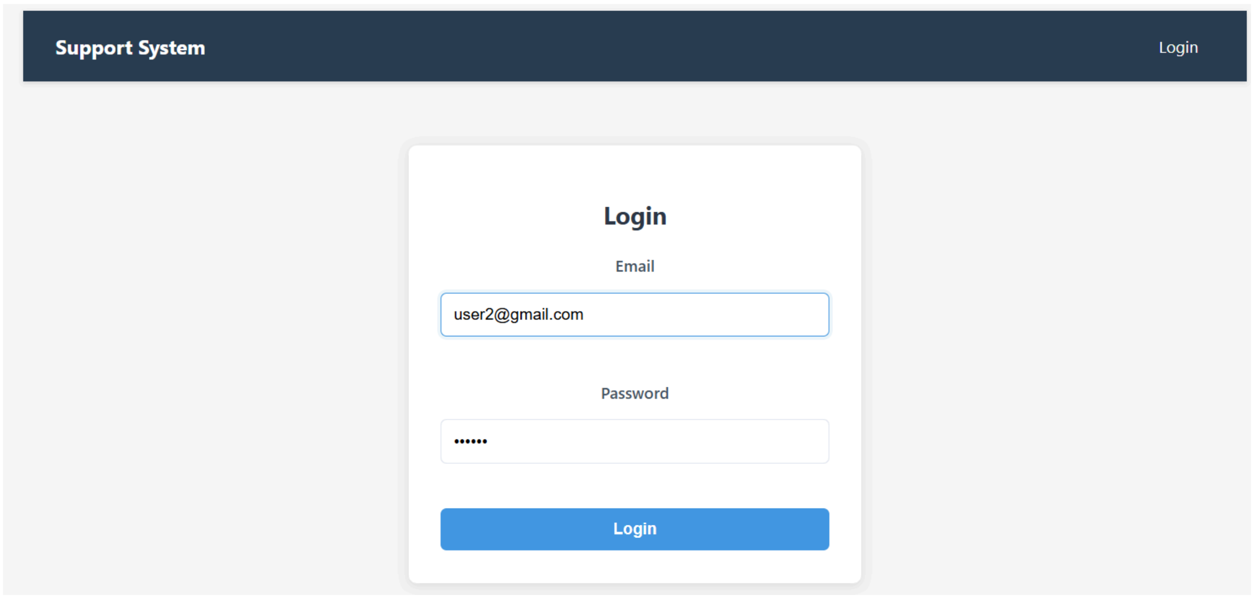
This product has been lost!

Subject: Lost Product

User2 Login

The system's privacy mechanisms are further demonstrated when a second user, **User2**, logs in. The steps for User2 are similar to those for User1 but with critical distinctions to ensure data isolation:

1. **Name Entry and Session Creation:**
    - User2 enters their name on the login screen. Firebase creates a separate session ID for User2, ensuring their messages are stored independently of User1's messages.
2. **Privacy Enforcement:**
    - Once logged in, User2 only has access to their own messages. Firebase's security rules ensure that User2 cannot view or interact with messages from User1 or any other user.
3. **Ticket Creation and Message Submission:**
    - User2 follows the same workflow as User1: selecting a subject, entering a message, and sending the ticket. The system confirms that their message is successfully sent by displaying it in real-time on the chat interface.



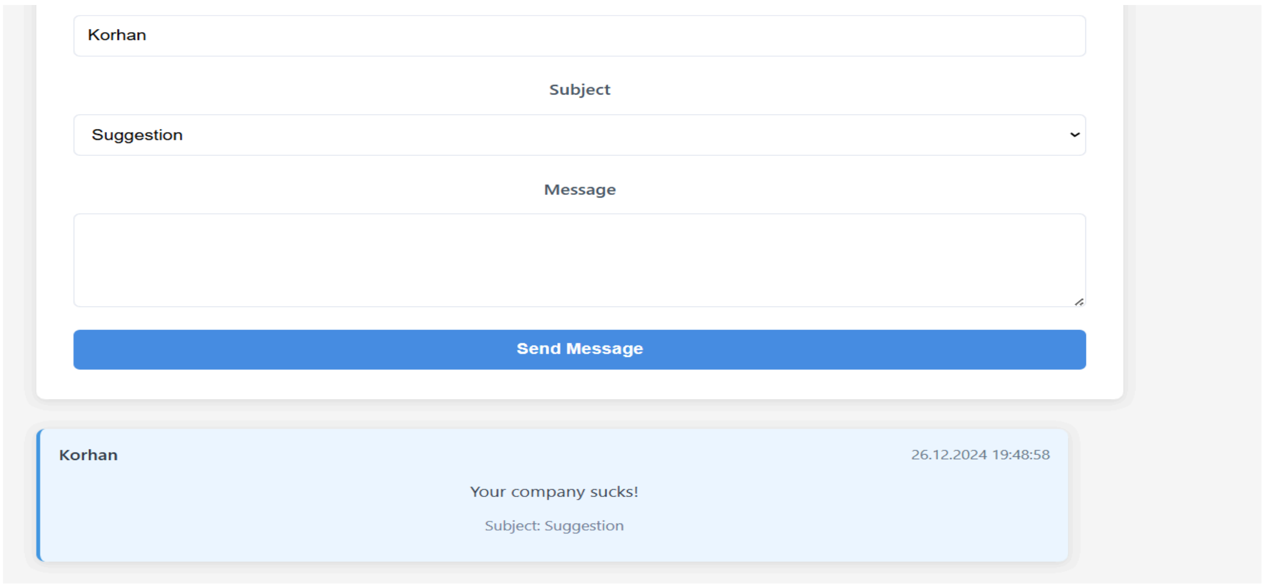
Email: user2@gmail.com

Password: ••••••

User2 Sends a Ticket (not seeing messages of User1)

When User2 sends a ticket, the system handles it independently of User1's data. The workflow includes:

- **Message Categorization:** User2 selects a subject that reflects their issue, helping the admin address it effectively.
- **Database Entry:** Firebase records User2's message under their unique session ID, along with metadata like the subject, timestamp, and content.
- **Interface Update:** The sent message appears in User2's chat interface in real-time, providing immediate feedback and assurance of successful delivery.



Korhan

Suggestion

Korhan 26.12.2024 19:48:58

Your company sucks!

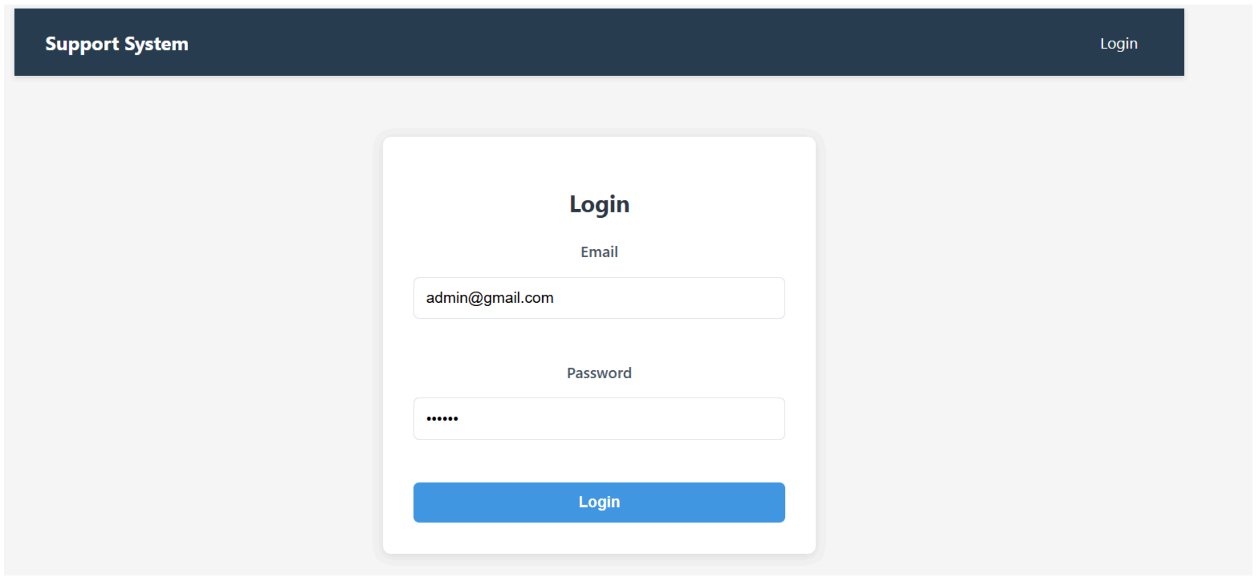
Subject: Suggestion

Admin Login

The admin login process is a crucial step in ensuring secure access to the system's backend. The workflow involves:

1. **Authentication:** The admin enters their credentials (e.g., username and password) into a dedicated login interface. This step verifies their identity and restricts access to authorized personnel only.
2. **Secure Session Initialization:** Once authenticated, the admin's session is initialized in Firebase, granting them access to all user interactions. Firebase's secure authentication protocols ensure that sensitive data is protected and only accessible to the admin.
3. **Access to Dashboard:** Upon successful login, the admin is redirected to the **Admin Dashboard**, a centralized platform for managing all user communications.

This login process ensures that only verified admins can access user messages, maintaining the integrity and privacy of the system.

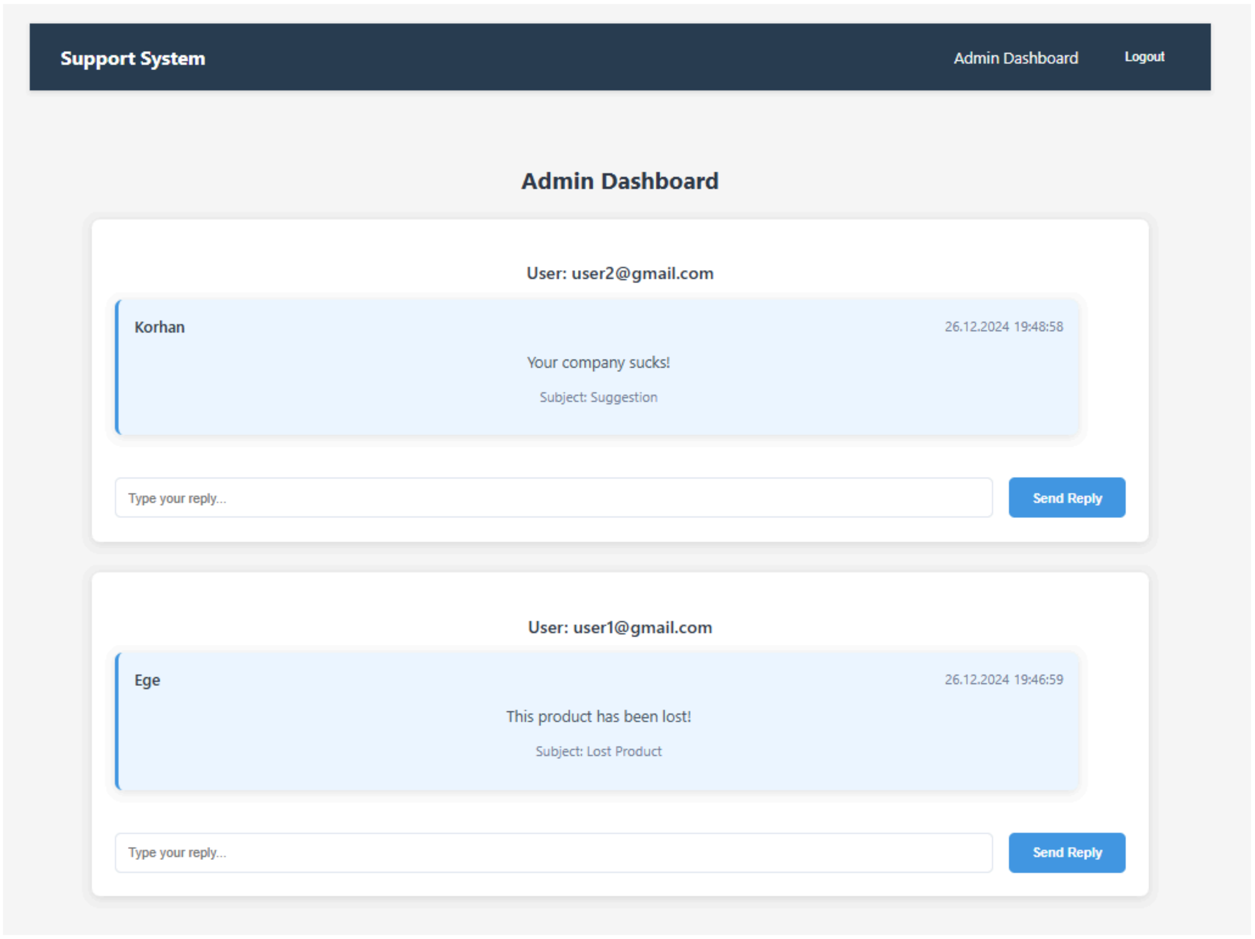


admin@gmail.com
••••••

Admin Dashboard (seeing messages of all users)

The **Admin Dashboard** is a centralized platform where the admin can manage user communications efficiently. Key features include:

1. **Unified View:** The dashboard consolidates all messages, categorized by user name, subject, and timestamp. This organization helps the admin quickly identify and prioritize tickets.
2. **Real-Time Updates:** New messages appear instantly on the dashboard without requiring manual refreshes, ensuring the admin always works with the latest data.
3. **User Threads:** Messages are grouped into threads for each user, allowing the admin to view complete conversation histories for better context.



Admin Dashboard

User: user2@gmail.com

Korhan 26.12.2024 19:48:58

Your company sucks!

Subject: Suggestion

User: user1@gmail.com

Ege 26.12.2024 19:46:59

This product has been lost!

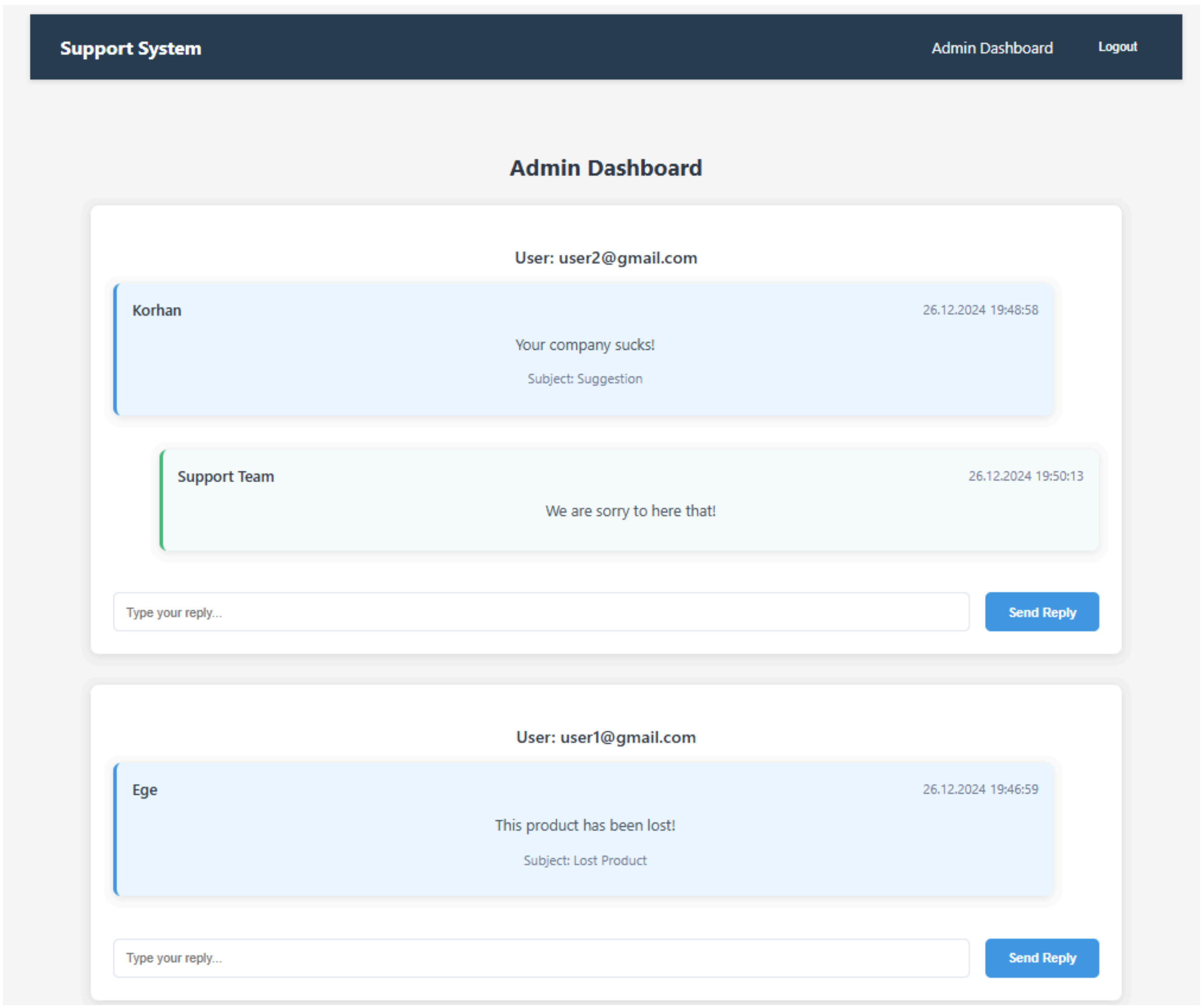
Subject: Lost Product

Admin Replies to User2

The admin begins by selecting User2’s message thread from the dashboard, which is clearly organized by user name and subject. This allows the admin to quickly review the issue and craft a response. For example:

- If User2 reported a "Lost Product," the admin might reply: "We are investigating this issue. Can you confirm your shipping address?"

Once the admin sends the reply, Firebase updates the database in real-time, ensuring that the response immediately appears on User2’s chat interface.



Admin Dashboard

User: user2@gmail.com

Korhan 26.12.2024 19:48:58

Your company sucks!

Subject: Suggestion

Support Team 26.12.2024 19:50:13

We are sorry to here that!

User: user1@gmail.com

Ege 26.12.2024 19:46:59

This product has been lost!

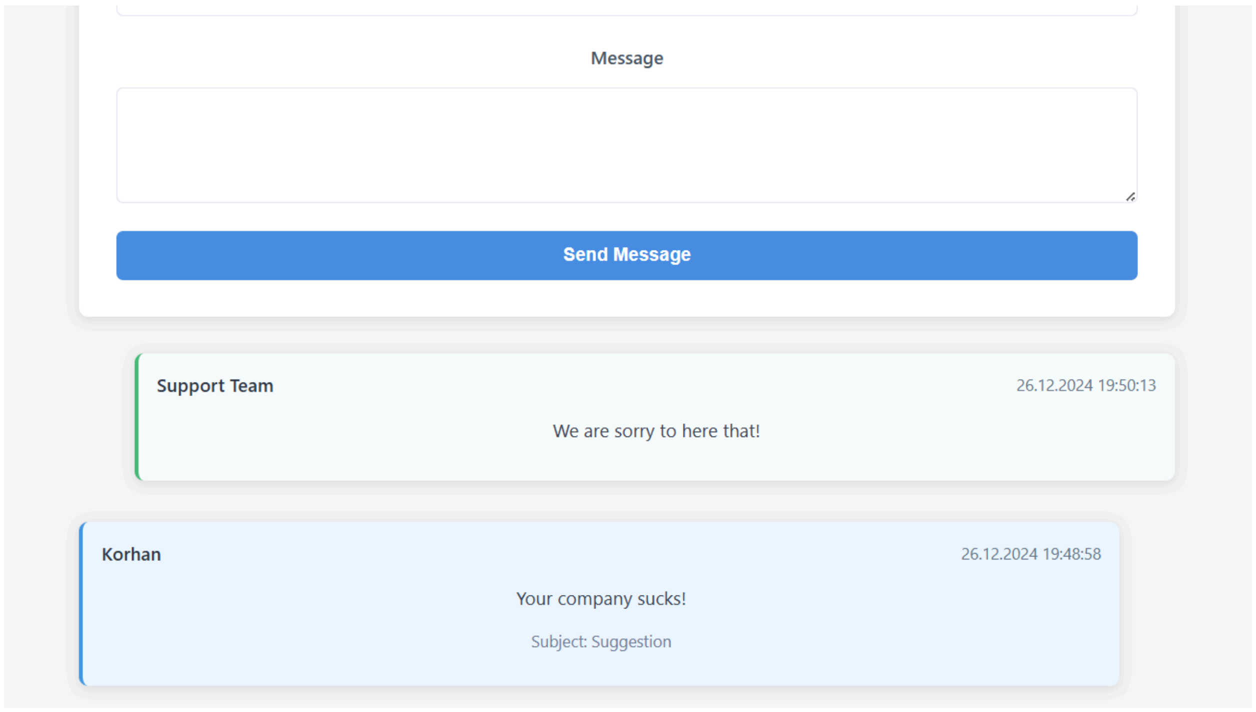
Subject: Lost Product

User2 Sees the Response of the Admin and Send Another Message

The admin’s response appears instantly in User2’s chat interface with a timestamp, reassuring them that their concern is being addressed. Upon reading the reply, User2 can react accordingly:

- Provide additional information, such as a shipping address, if requested by the admin.
- Acknowledge the resolution or ask follow-up questions if needed.

User2 then types and sends another message, which is immediately updated in Firebase and visible to the admin. This seamless exchange continues until the issue is resolved, with both parties able to view the entire conversation thread. This transparency and real-time communication foster trust and improve the support experience.



Support Team 26.12.2024 19:50:13

We are sorry to here that!

Korhan 26.12.2024 19:48:58

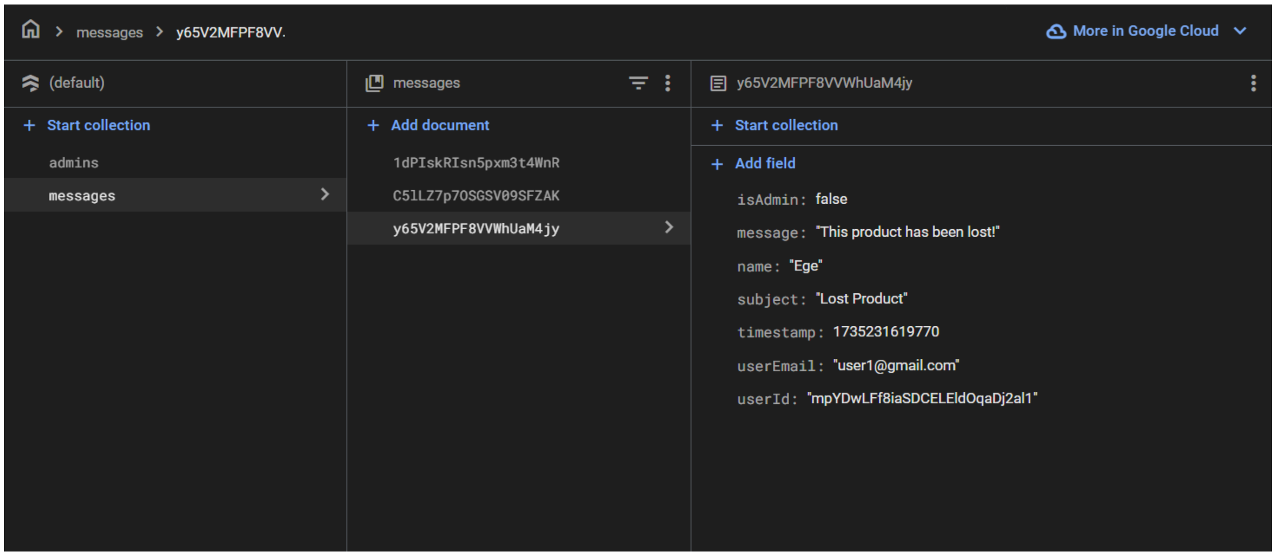
Your company sucks!

Subject: Suggestion

Key Features of the Implementation

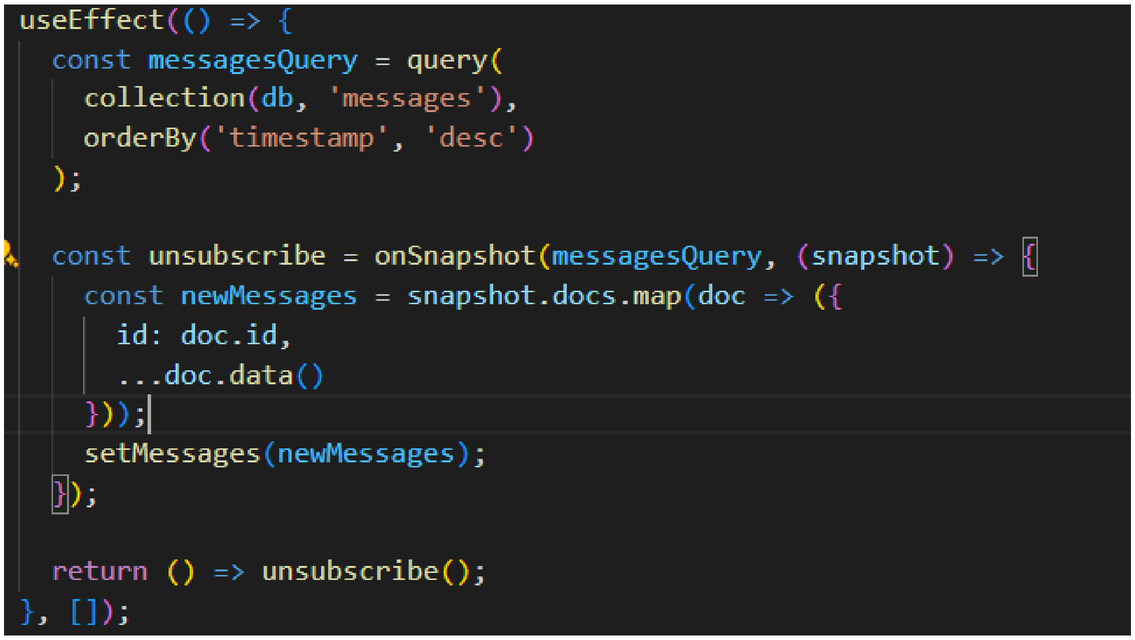
1. **Real-Time Communication:**
   Firebase's Realtime Database is the backbone of this project, enabling instant synchronization of data between users/guests and the admin. Messages sent by users appear immediately in the admin dashboard and vice versa, ensuring that no delays occur in communication. This real-time functionality enhances the user experience by providing immediate feedback and fosters a responsive, dynamic support system.

   This is the firestore document stores our messages:



```
isAdmin: false
message: "This product has been lost!"
name: "Ege"
subject: "Lost Product"
timestamp: 1735231619770
userEmail: "user1@gmail.com"
userId: "mpYDwLFf8iaSDCELEldOqaDj2al1"
```

We have used React for front-end and this is how we achieved real time updates via firestore:

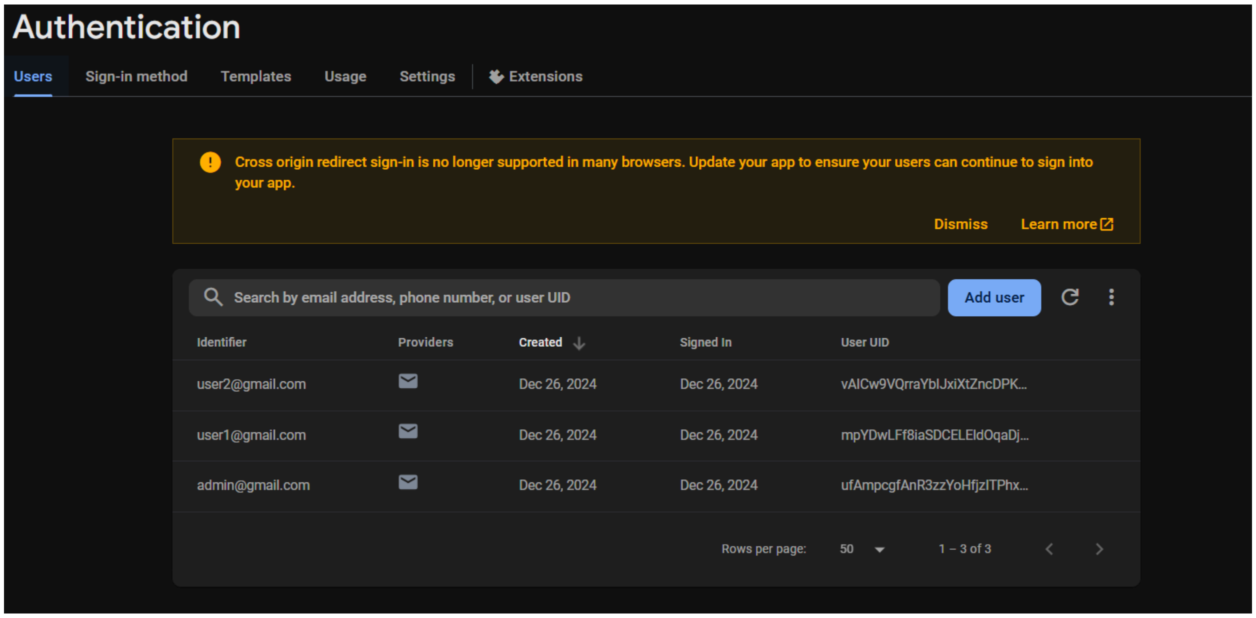


```
useEffect(() => {
  const messagesQuery = query(
    collection(db, 'messages'),
    orderBy('timestamp', 'desc')
  );

  const unsubscribe = onSnapshot(messagesQuery, (snapshot) => {
    const newMessages = snapshot.docs.map(doc => ({
      id: doc.id,
      ...doc.data()
    }));
    setMessages(newMessages);
  });

  return () => unsubscribe();
}, []);
```

This component renders new messages when a new message is added to the database until it is closed.

2. **User Privacy:**
   User privacy is a critical consideration in this implementation. Firebase security rules are configured to ensure that each user can only access their own messages. This isolation prevents unauthorized access to other users' data, creating a secure and trustworthy environment. Meanwhile, admins are granted full visibility of all interactions to manage support tickets effectively, but their access is strictly controlled through secure authentication.

This is our authentication flow:

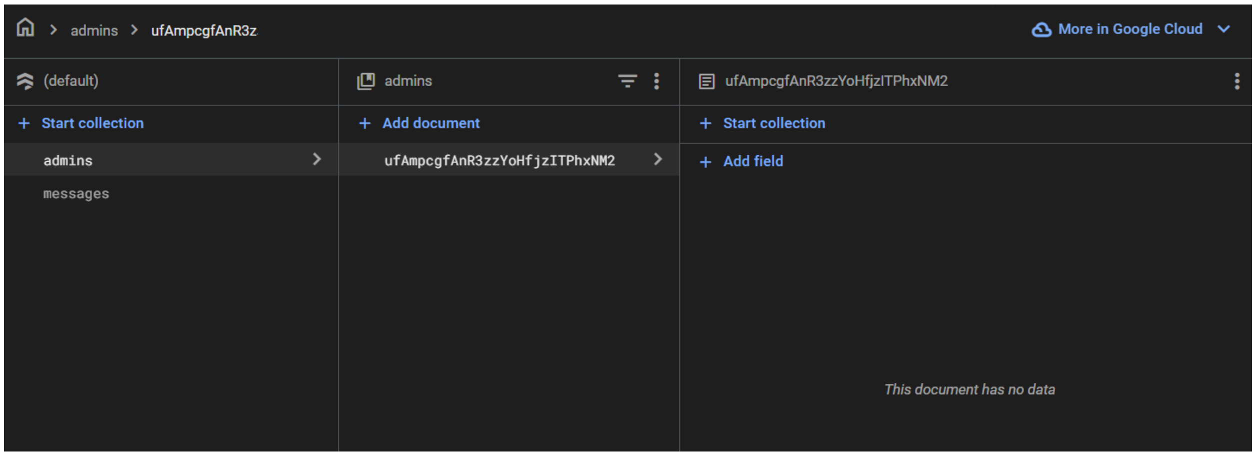

Authentication

Cross origin redirect sign-in is no longer supported in many browsers. Update your app to ensure your users can continue to sign into your app.

Identifier	Providers	Created	Signed In	User UID
user2@gmail.com	✉	Dec 26, 2024	Dec 26, 2024	vAICw9VQrraYbIJxiXtZncDPK...
user1@gmail.com	✉	Dec 26, 2024	Dec 26, 2024	mpYDwLFf8iaSDCELEldOqaDj...
admin@gmail.com	✉	Dec 26, 2024	Dec 26, 2024	ufAmpcgfAnR3zzYoHfjzITPhx...

Rows per page: 50 1 – 3 of 3

This is the document where we store our admins:


admins > ufAmpcgfAnR3z

This document has no data

3. **Categorization:**
   The inclusion of a subject dropdown during ticket creation allows users to categorize their issues efficiently. This feature simplifies ticket organization for the admin, making it easier to prioritize and address issues based on their nature. For example, subjects like "Defective Product" or "Late Order" provide immediate context, helping the admin focus on urgent or recurring problems.

4. **Transparency and Organization:**
   The ability for users and admins to view complete conversation threads ensures no details are missed during the support process. The timestamped messages add clarity to the interactions, allowing both parties to track the timeline of the conversation effortlessly.

Conclusion

This project demonstrates the successful implementation of a robust and user-friendly **real-time support system** using Firebase. By leveraging the strengths of Firebase's Realtime Database, the system achieves:

- **Efficient Data Management:** The structured storage of messages and metadata ensures seamless retrieval and organization.
- **Strong Privacy Measures:** Security rules protect user data while allowing admins to access the necessary information for support management.
- **Enhanced User Experience:** The intuitive interface, real-time updates, and transparency in communication contribute to a positive user experience.

The project not only meets all specified requirements but also provides a scalable foundation for future developments. Potential enhancements could include:

- Integrating sentiment analysis to gauge user satisfaction.
- Adding automation through AI-powered chatbots for preliminary support.
- Expanding functionalities to support multiple languages for broader accessibility.

Overall, the system is a testament to how modern technologies like Firebase can be utilized to create a seamless and secure communication platform, benefiting both users and administrators.