Company Database Management System

CS 306 – PROJECT PHASE 3

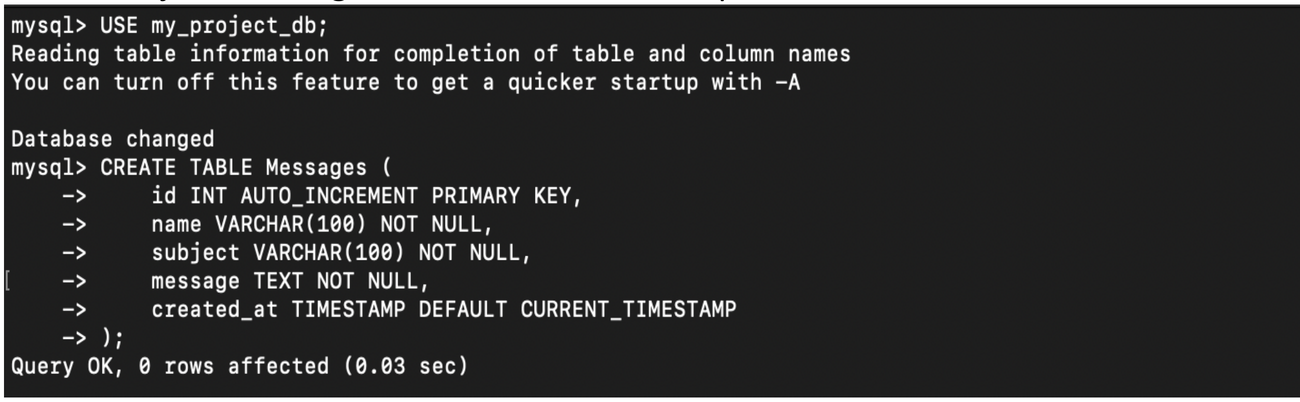
Yasin Barkın Başaran

STUDENT ID: 28890

CS306 Project Phase 3: Implement a Real-Time Support Page

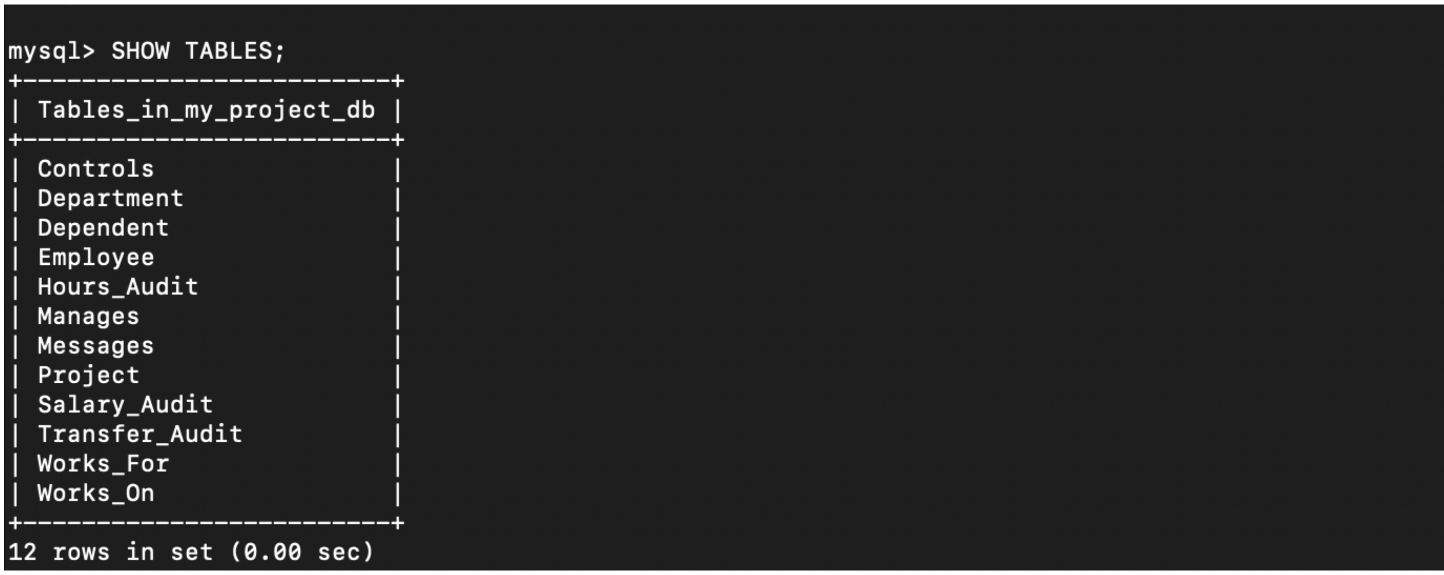
1. Creating the Messages Table

- **Explanation:** A messages table was created in MySQL to store user messages. This table holds details for each message, such as the sender's name, subject, message content, and timestamp.



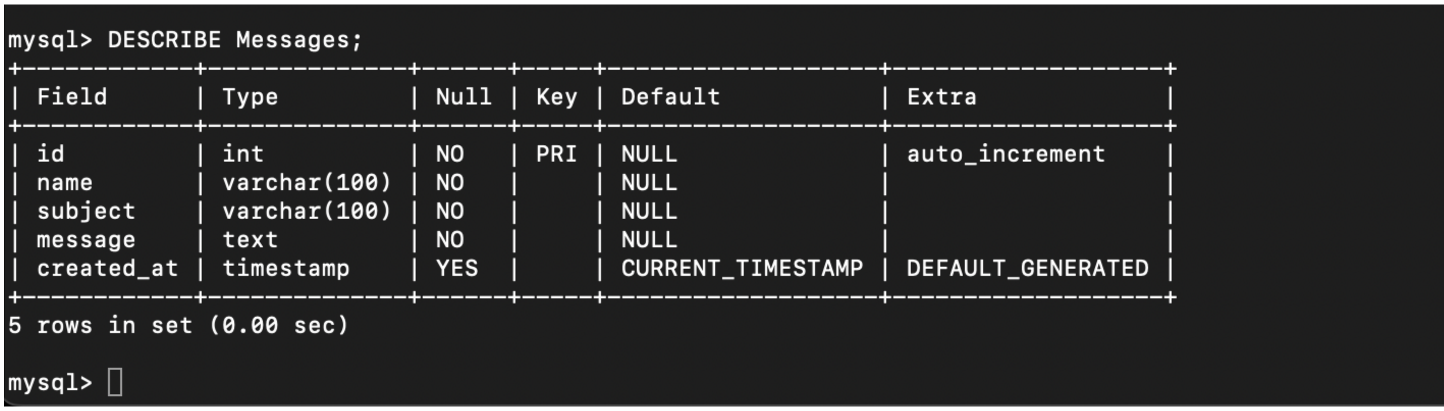
```
mysql> USE my_project_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> CREATE TABLE Messages (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100) NOT NULL,
    ->     subject VARCHAR(100) NOT NULL,
    ->     message TEXT NOT NULL,
    ->     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.03 sec)
```

- Messages Table is in Tables because of the creation of the table.



```
mysql> SHOW TABLES;
+-------------------------+
| Tables_in_my_project_db |
+-------------------------+
| Controls                |
| Department              |
| Dependent               |
| Employee                |
| Hours_Audit             |
| Manages                 |
| Messages                |
| Project                 |
| Salary_Audit            |
| Transfer_Audit          |
| Works_For               |
| Works_On                |
+-------------------------+
12 rows in set (0.00 sec)
```

- You can check the details of Messages Table below,

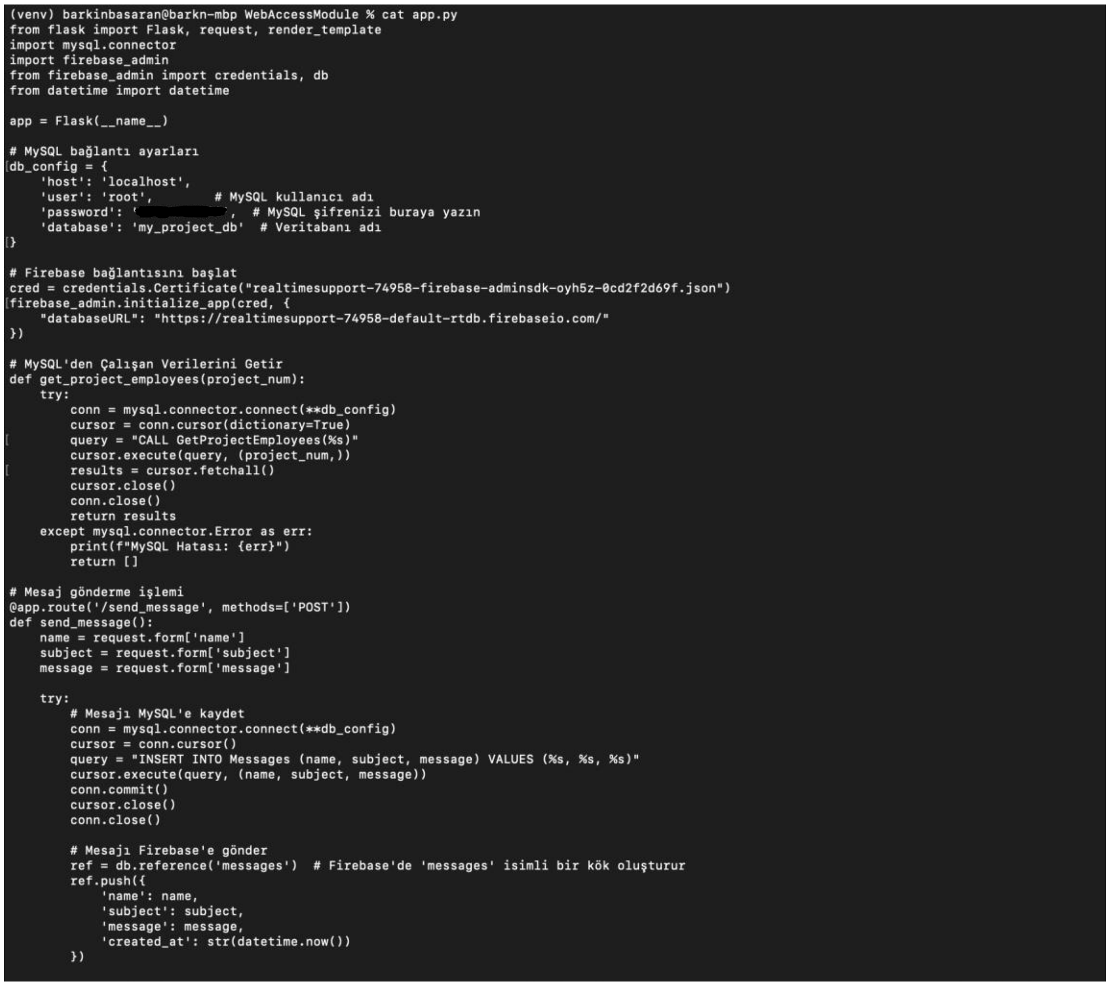


```
mysql> DESCRIBE Messages;
+------------+--------------+------+-----+-------------------+-------------------+
| Field      | Type         | Null | Key | Default           | Extra             |
+------------+--------------+------+-----+-------------------+-------------------+
| id         | int          | NO   | PRI | NULL              | auto_increment    |
| name       | varchar(100) | NO   |     | NULL              |                   |
| subject    | varchar(100) | NO   |     | NULL              |                   |
| message    | text         | NO   |     | NULL              |                   |
| created_at | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+--------------+------+-----+-------------------+-------------------+
5 rows in set (0.00 sec)

mysql> 
```

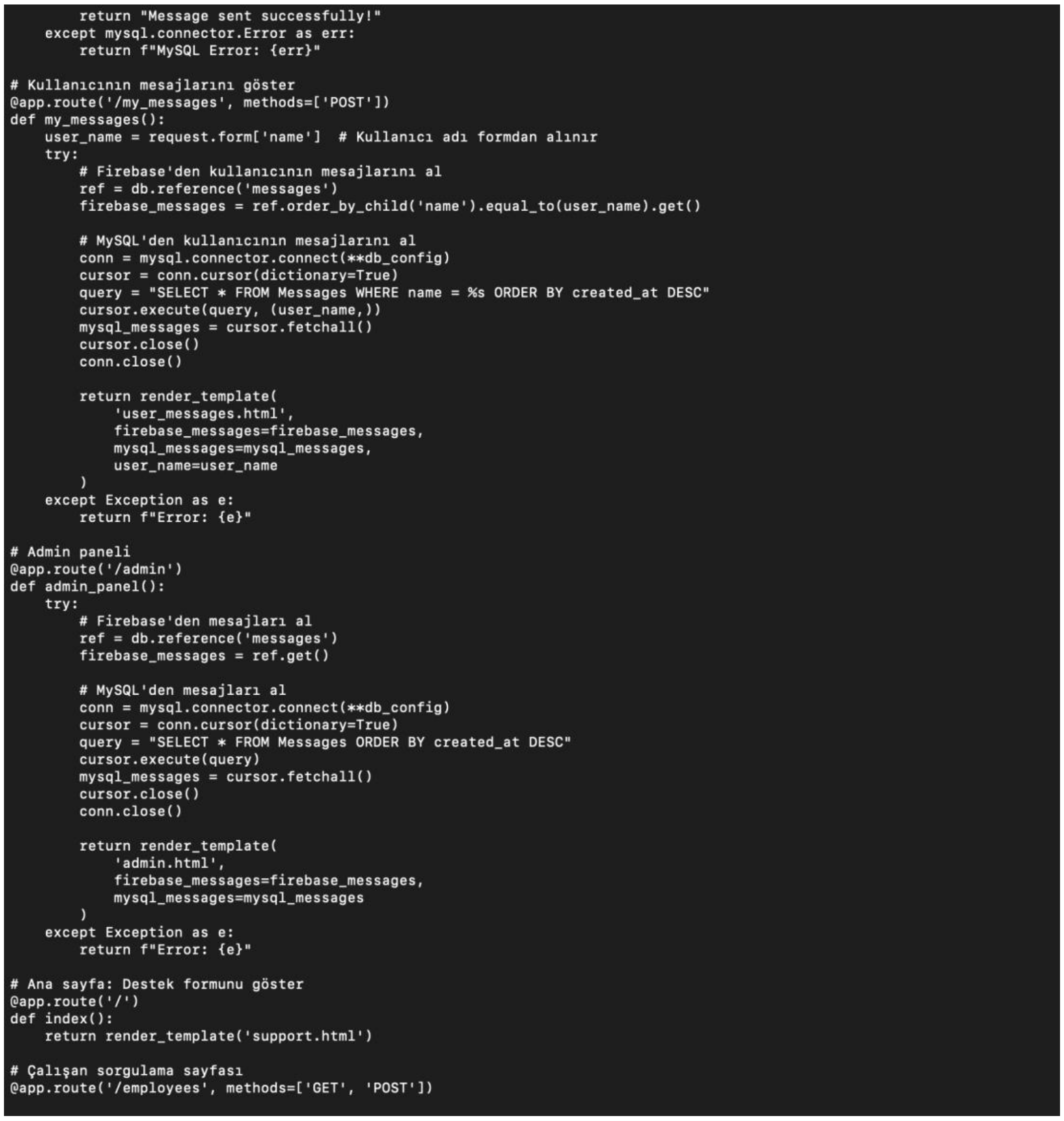
2. Starting the Flask Application

- **Explanation:** A flask application was initialized to manage user message submission and display features. The application connects to both Firebase and MySQL for data storage.
- I revised the app.py file code to implement Firebase and MySQL efficiently. Also, I added code blocks for support.html, and other necessary code blocks for Firebase and MySQL such as admin_panel function. You can check the entire app.py below.

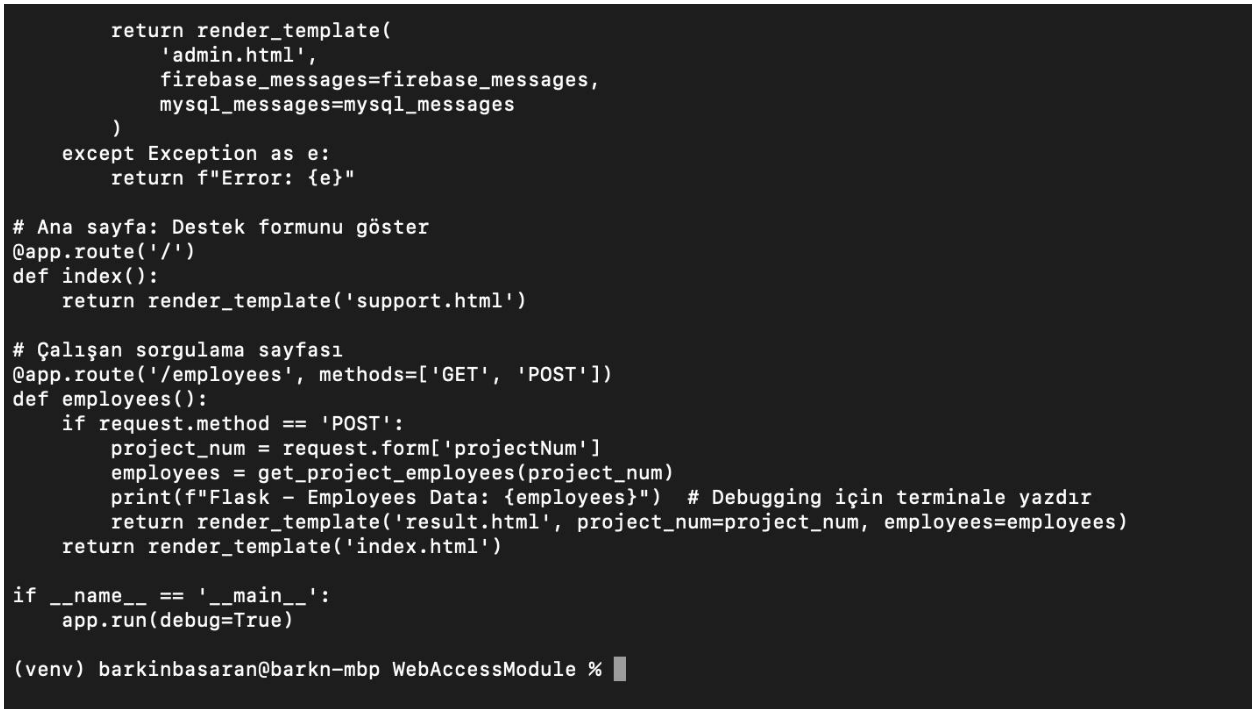
Code of app.py:



```
(venv) barkinbasaran@barkn-mbp WebAccessModule % cat app.py
from flask import Flask, request, render_template
import mysql.connector
import firebase_admin
from firebase_admin import credentials, db
from datetime import datetime

app = Flask(__name__)

# MySQL bağlantı ayarları
db_config = {
    'host': 'localhost',
    'user': 'root',        # MySQL kullanıcı adı
    'password': '[redacted]',  # MySQL şifrenizi buraya yazın
    'database': 'my_project_db'  # Veritabanı adı
}

# Firebase bağlantısını başlat
cred = credentials.Certificate("realtimesupport-74958-firebase-adminsdk-oyh5z-0cd2f2d69f.json")
firebase_admin.initialize_app(cred, {
    "databaseURL": "https://realtimesupport-74958-default-rtdb.firebaseio.com/"
})

# MySQL'den Çalışan Verilerini Getir
def get_project_employees(project_num):
    try:
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor(dictionary=True)
        query = "CALL GetProjectEmployees(%s)"
        cursor.execute(query, (project_num,))
        results = cursor.fetchall()
        cursor.close()
        conn.close()
        return results
    except mysql.connector.Error as err:
        print(f"MySQL Hatası: {err}")
        return []

# Mesaj gönderme işlemi
@app.route('/send_message', methods=['POST'])
def send_message():
    name = request.form['name']
    subject = request.form['subject']
    message = request.form['message']

    try:
        # Mesajı MySQL'e kaydet
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()
        query = "INSERT INTO Messages (name, subject, message) VALUES (%s, %s, %s)"
        cursor.execute(query, (name, subject, message))
        conn.commit()
        cursor.close()
        conn.close()

        # Mesajı Firebase'e gönder
        ref = db.reference('messages')  # Firebase'de 'messages' isimli bir kök oluşturur
        ref.push({
            'name': name,
            'subject': subject,
            'message': message,
            'created_at': str(datetime.now())
        })
```



```
        return "Message sent successfully!"
    except mysql.connector.Error as err:
        return f"MySQL Error: {err}"

# Kullanıcının mesajlarını göster
@app.route('/my_messages', methods=['POST'])
def my_messages():
    user_name = request.form['name']  # Kullanıcı adı formdan alınır
    try:
        # Firebase'den kullanıcının mesajlarını al
        ref = db.reference('messages')
        firebase_messages = ref.order_by_child('name').equal_to(user_name).get()

        # MySQL'den kullanıcının mesajlarını al
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor(dictionary=True)
        query = "SELECT * FROM Messages WHERE name = %s ORDER BY created_at DESC"
        cursor.execute(query, (user_name,))
        mysql_messages = cursor.fetchall()
        cursor.close()
        conn.close()

        return render_template(
            'user_messages.html',
            firebase_messages=firebase_messages,
            mysql_messages=mysql_messages,
            user_name=user_name
        )
    except Exception as e:
        return f"Error: {e}"

# Admin paneli
@app.route('/admin')
def admin_panel():
    try:
        # Firebase'den mesajları al
        ref = db.reference('messages')
        firebase_messages = ref.get()

        # MySQL'den mesajları al
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor(dictionary=True)
        query = "SELECT * FROM Messages ORDER BY created_at DESC"
        cursor.execute(query)
        mysql_messages = cursor.fetchall()
        cursor.close()
        conn.close()

        return render_template(
            'admin.html',
            firebase_messages=firebase_messages,
            mysql_messages=mysql_messages
        )
    except Exception as e:
        return f"Error: {e}"

# Ana sayfa: Destek formunu göster
@app.route('/')
def index():
    return render_template('support.html')

# Çalışan sorgulama sayfası
@app.route('/employees', methods=['GET', 'POST'])
```

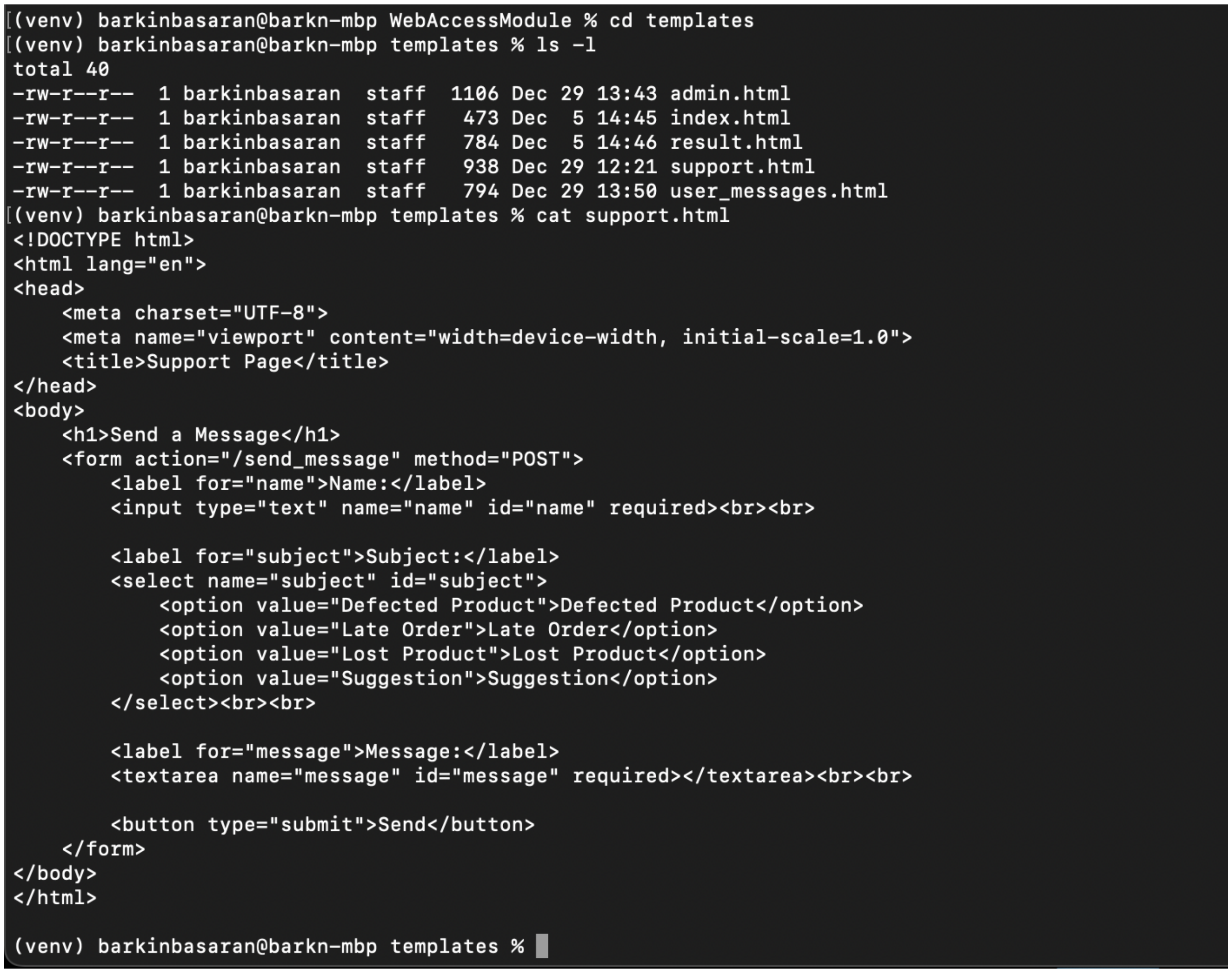


```
        return render_template(
            'admin.html',
            firebase_messages=firebase_messages,
            mysql_messages=mysql_messages
        )
    except Exception as e:
        return f"Error: {e}"

# Ana sayfa: Destek formunu göster
@app.route('/')
def index():
    return render_template('support.html')

# Çalışan sorgulama sayfası
@app.route('/employees', methods=['GET', 'POST'])
def employees():
    if request.method == 'POST':
        project_num = request.form['projectNum']
        employees = get_project_employees(project_num)
        print(f"Flask - Employees Data: {employees}")  # Debugging için terminale yazdır
        return render_template('result.html', project_num=project_num, employees=employees)
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)

(venv) barkinbasaran@barkn-mbp WebAccessModule % 
```

3. User Message Submission Form (Support Page)

- Explanation: A user -facing support page was implemented where users can submit their name, message subject, and message content.
- Support page opened in a browser at **http://127.0.0.1:5000/**

support.html Code:



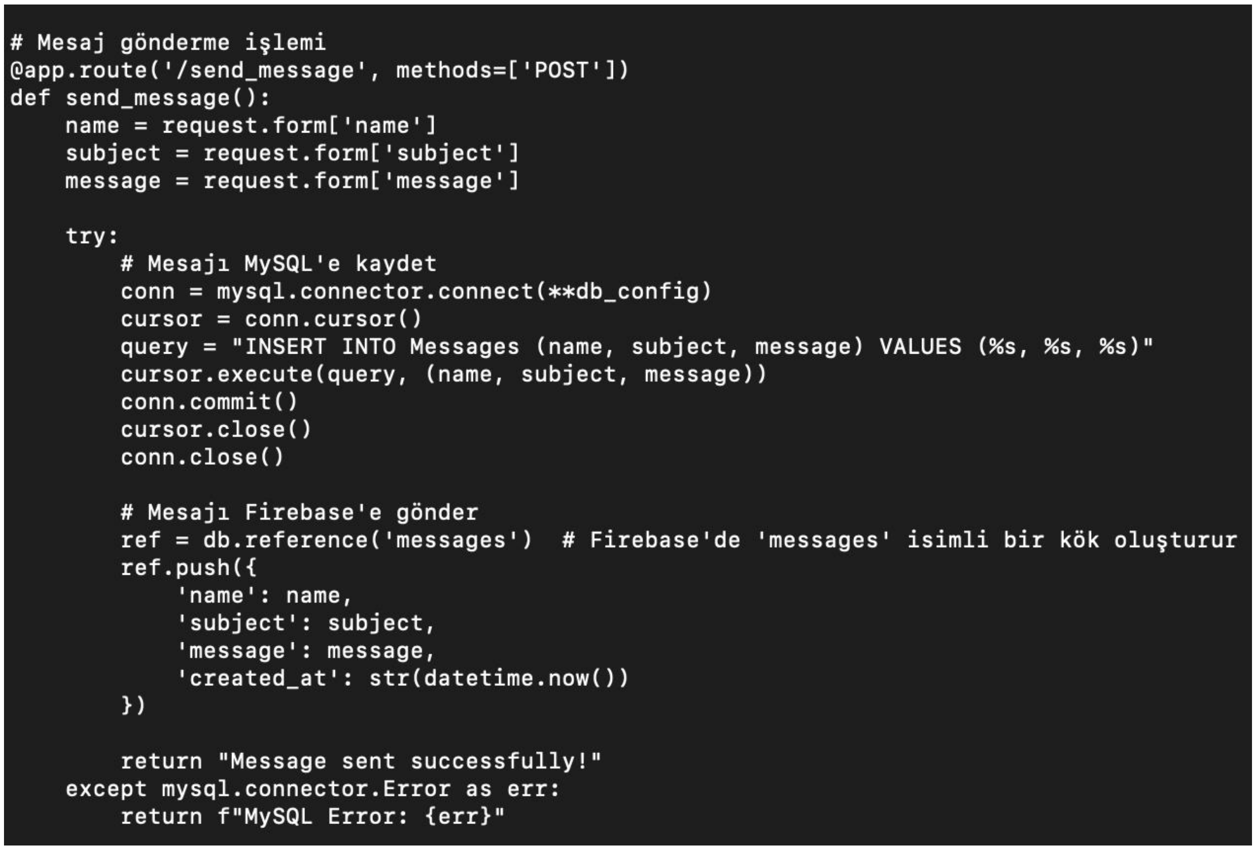
```
(venv) barkinbasaran@barkn-mbp WebAccessModule % cd templates
(venv) barkinbasaran@barkn-mbp templates % ls -l
total 40
-rw-r--r--  1 barkinbasaran  staff  1106 Dec 29 13:43 admin.html
-rw-r--r--  1 barkinbasaran  staff   473 Dec  5 14:45 index.html
-rw-r--r--  1 barkinbasaran  staff   784 Dec  5 14:46 result.html
-rw-r--r--  1 barkinbasaran  staff   938 Dec 29 12:21 support.html
-rw-r--r--  1 barkinbasaran  staff   794 Dec 29 13:50 user_messages.html
(venv) barkinbasaran@barkn-mbp templates % cat support.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Support Page</title>
</head>
<body>
    <h1>Send a Message</h1>
    <form action="/send_message" method="POST">
        <label for="name">Name:</label>
        <input type="text" name="name" id="name" required><br><br>

        <label for="subject">Subject:</label>
        <select name="subject" id="subject">
            <option value="Defected Product">Defected Product</option>
            <option value="Late Order">Late Order</option>
            <option value="Lost Product">Lost Product</option>
            <option value="Suggestion">Suggestion</option>
        </select><br><br>

        <label for="message">Message:</label>
        <textarea name="message" id="message" required></textarea><br><br>

        <button type="submit">Send</button>
    </form>
</body>
</html>

(venv) barkinbasaran@barkn-mbp templates % 
```

4. Saving Messages to Firebase and MySQL

- **Explanation:** Messages submitted by users were saved to both Firebase Realtime Database and MySQL

Code:

- The /sendmessage route in app.py:

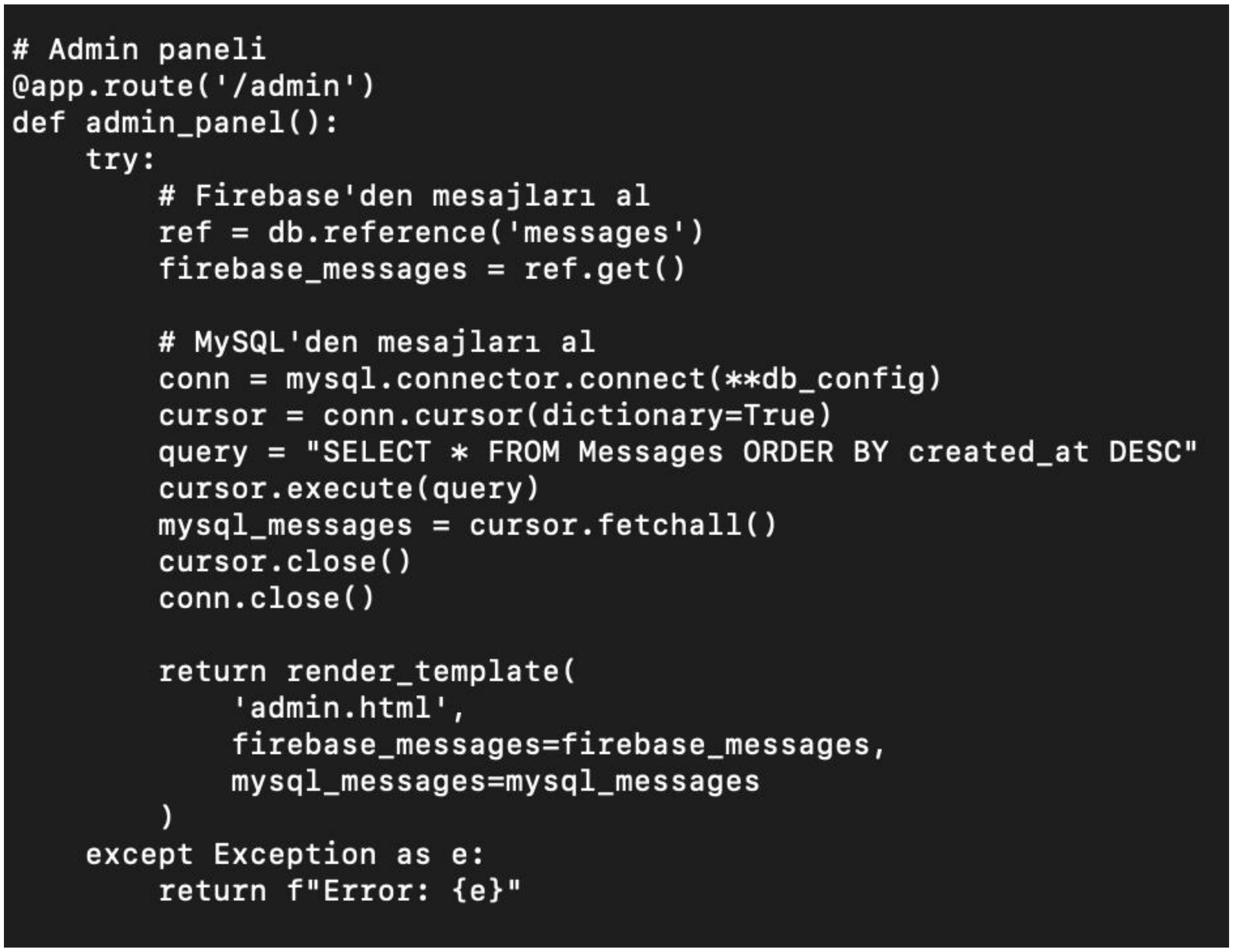


```python
# Mesaj gönderme işlemi
@app.route('/send_message', methods=['POST'])
def send_message():
    name = request.form['name']
    subject = request.form['subject']
    message = request.form['message']

    try:
        # Mesajı MySQL'e kaydet
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()
        query = "INSERT INTO Messages (name, subject, message) VALUES (%s, %s, %s)"
        cursor.execute(query, (name, subject, message))
        conn.commit()
        cursor.close()
        conn.close()

        # Mesajı Firebase'e gönder
        ref = db.reference('messages')  # Firebase'de 'messages' isimli bir kök oluşturur
        ref.push({
            'name': name,
            'subject': subject,
            'message': message,
            'created_at': str(datetime.now())
        })

        return "Message sent successfully!"
    except mysql.connector.Error as err:
        return f"MySQL Error: {err}"
```

5. Admin Panel

- Explanation: An admin panel was implemented to display all messages from both Firebase and MySQL in a unified interface.
- The admin panel opened in a browser at **http://127.0.0.1:5000/admin**

Code:

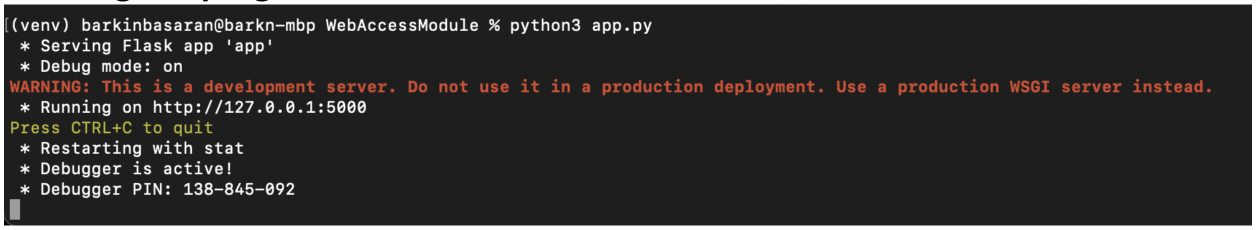


```python
# Admin paneli
@app.route('/admin')
def admin_panel():
    try:
        # Firebase'den mesajları al
        ref = db.reference('messages')
        firebase_messages = ref.get()

        # MySQL'den mesajları al
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor(dictionary=True)
        query = "SELECT * FROM Messages ORDER BY created_at DESC"
        cursor.execute(query)
        mysql_messages = cursor.fetchall()
        cursor.close()
        conn.close()

        return render_template(
            'admin.html',
            firebase_messages=firebase_messages,
            mysql_messages=mysql_messages
        )
    except Exception as e:
        return f"Error: {e}"
```

6. Testing and Showing Hole Process of Phase 3

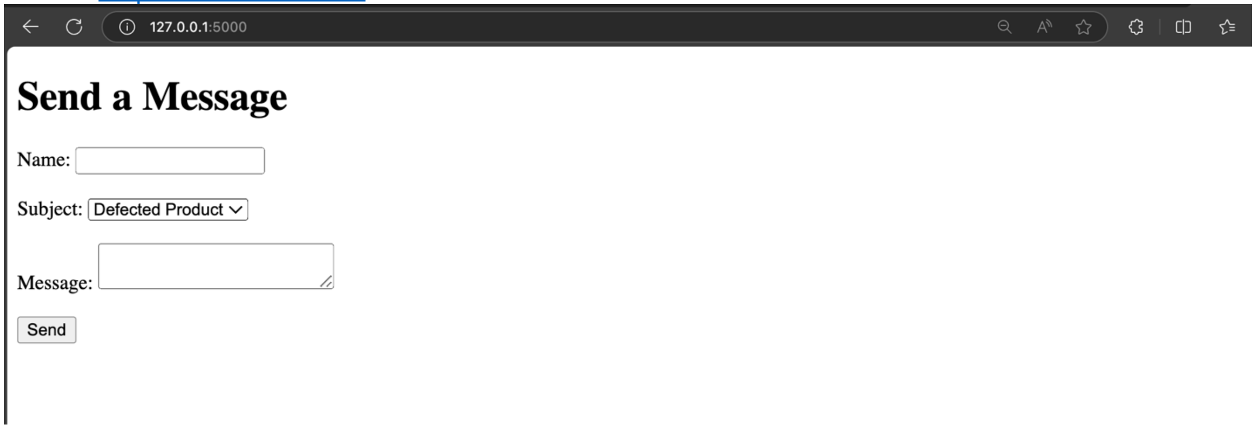
- **Explanation:** This part shows entire process and every screenshot of how the program is running generally to illustrate with an efficient way.

Running the program from terminal:



```
(venv) barkinbasaran@barkn-mbp WebAccessModule % python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 138-845-092
```

Search http://127.0.0.1:5000 on web:



127.0.0.1:5000

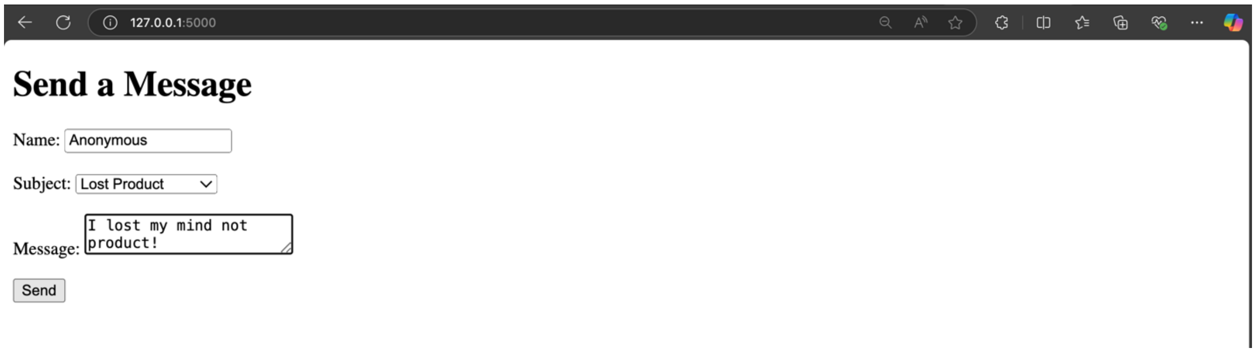
Send a Message

Name:

Subject: Defected Product

Message:

Send



127.0.0.1:5000

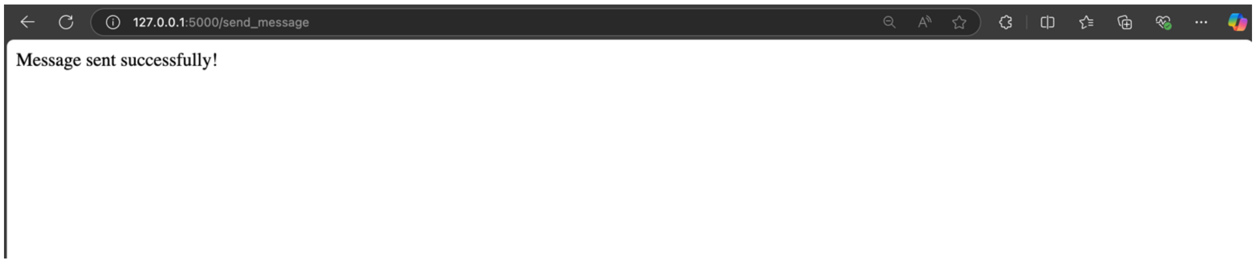
Send a Message

Name: Anonymous

Subject: Lost Product

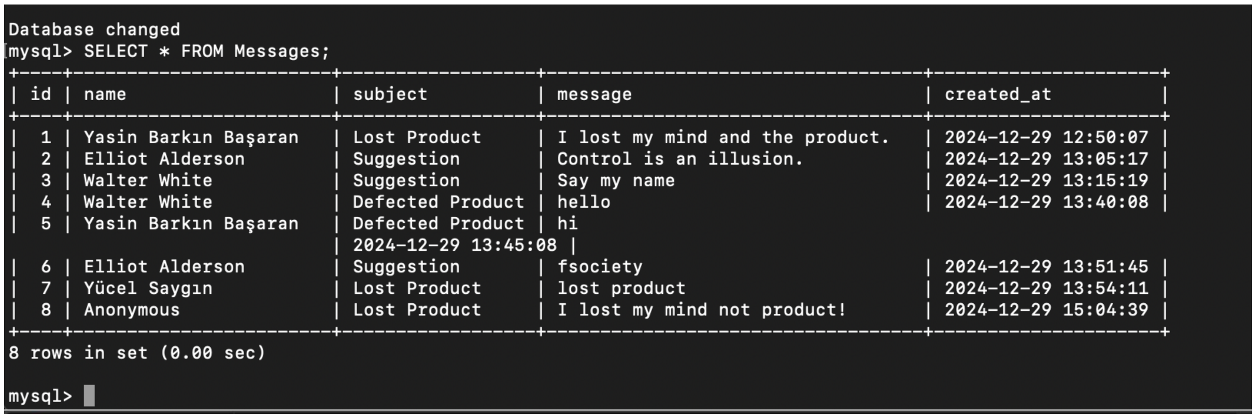
Message: I lost my mind not product!

Send

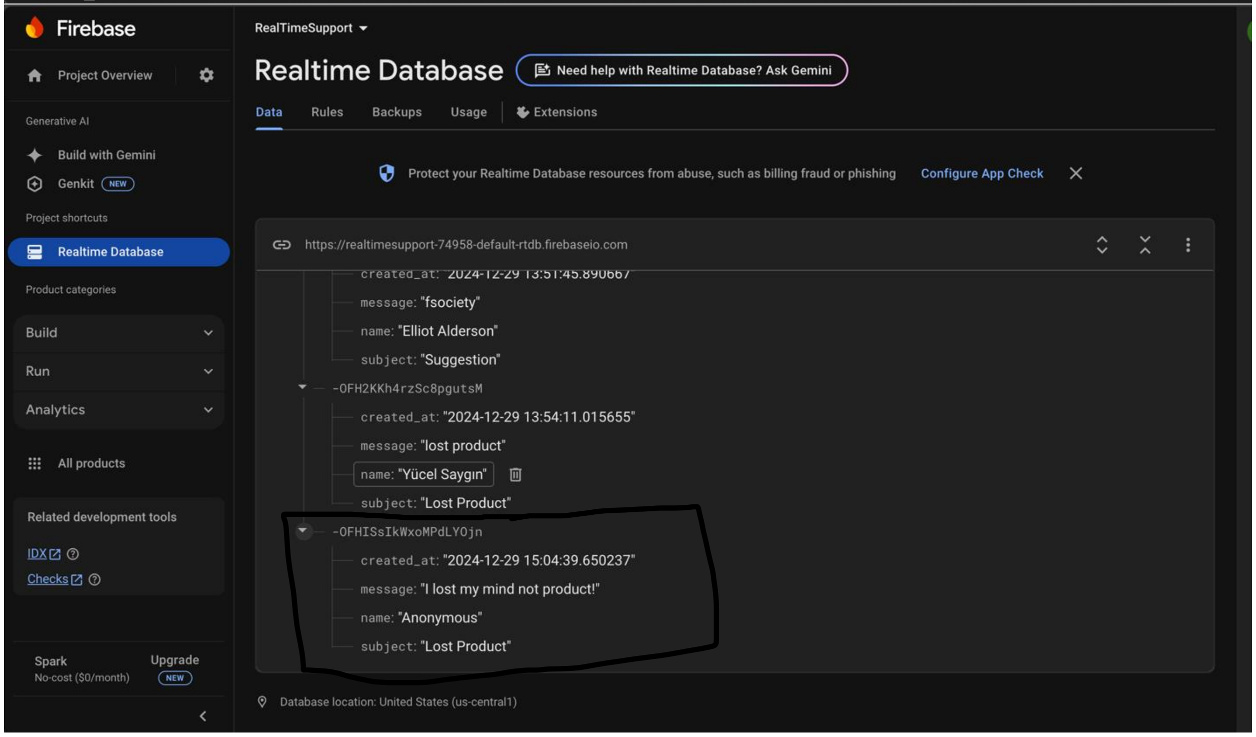


127.0.0.1:5000/send_message

Message sent successfully!



```
Database changed
mysql> SELECT * FROM Messages;
+----+-----------------------+-----------------+----------------------------------+---------------------+
| id | name                  | subject         | message                          | created_at          |
+----+-----------------------+-----------------+----------------------------------+---------------------+
|  1 | Yasin Barkın Başaran  | Lost Product    | I lost my mind and the product.  | 2024-12-29 12:50:07 |
|  2 | Elliot Alderson       | Suggestion      | Control is an illusion.          | 2024-12-29 13:05:17 |
|  3 | Walter White          | Suggestion      | Say my name                      | 2024-12-29 13:15:19 |
|  4 | Walter White          | Defected Product | hello                           | 2024-12-29 13:40:08 |
|  5 | Yasin Barkın Başaran  | Defected Product | hi
                          | 2024-12-29 13:45:08 |
|  6 | Elliot Alderson       | Suggestion      | fsociety                         | 2024-12-29 13:51:45 |
|  7 | Yücel Saygın          | Lost Product    | lost product                     | 2024-12-29 13:54:11 |
|  8 | Anonymous             | Lost Product    | I lost my mind not product!      | 2024-12-29 15:04:39 |
+----+-----------------------+-----------------+----------------------------------+---------------------+
8 rows in set (0.00 sec)

mysql>
```



Realtime Database

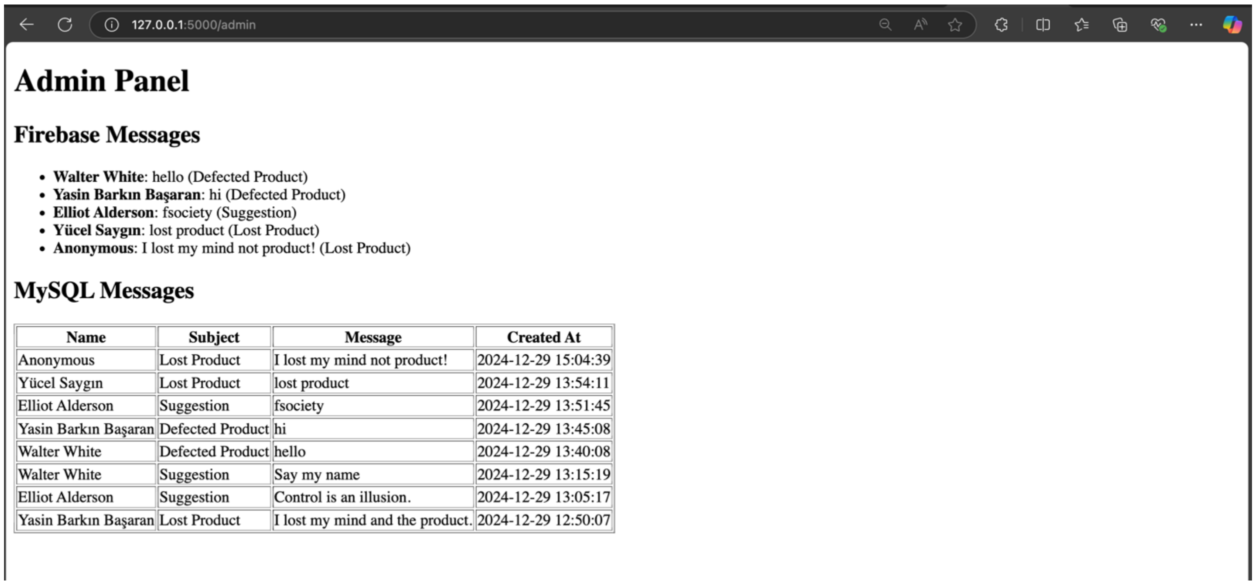
Protect your Realtime Database resources from abuse, such as billing fraud or phishing

https://realtimesupport-74958-default-rtdb.firebaseio.com

```
created_at: "2024-12-29 13:51:45.890667"
message: "fsociety"
name: "Elliot Alderson"
subject: "Suggestion"
-OFH2KKh4rzSc8pgutsM
  created_at: "2024-12-29 13:54:11.015655"
  message: "lost product"
  name: "Yücel Saygın"
  subject: "Lost Product"
-OFHISsIkWxoMPdLYOjn
  created_at: "2024-12-29 15:04:39.650237"
  message: "I lost my mind not product!"
  name: "Anonymous"
  subject: "Lost Product"
```

Database location: United States (us-central1)

You can check the admin panel:



127.0.0.1:5000/admin

Admin Panel

Firebase Messages

- **Walter White**: hello (Defected Product)
- **Yasin Barkın Başaran**: hi (Defected Product)
- **Elliot Alderson**: fsociety (Suggestion)
- **Yücel Saygın**: lost product (Lost Product)
- **Anonymous**: I lost my mind not product! (Lost Product)

MySQL Messages

Name	Subject	Message	Created At
Anonymous	Lost Product	I lost my mind not product!	2024-12-29 15:04:39
Yücel Saygın	Lost Product	lost product	2024-12-29 13:54:11
Elliot Alderson	Suggestion	fsociety	2024-12-29 13:51:45
Yasin Barkın Başaran	Defected Product	hi	2024-12-29 13:45:08
Walter White	Defected Product	hello	2024-12-29 13:40:08
Walter White	Suggestion	Say my name	2024-12-29 13:15:19
Elliot Alderson	Suggestion	Control is an illusion.	2024-12-29 13:05:17
Yasin Barkın Başaran	Lost Product	I lost my mind and the product.	2024-12-29 12:50:07

Conclusion: Requirements of CS306 Project Phase studied and showed each step. Also, the process of how the program is running.