PYTHON BOOTCAMP

www.jomhack.com

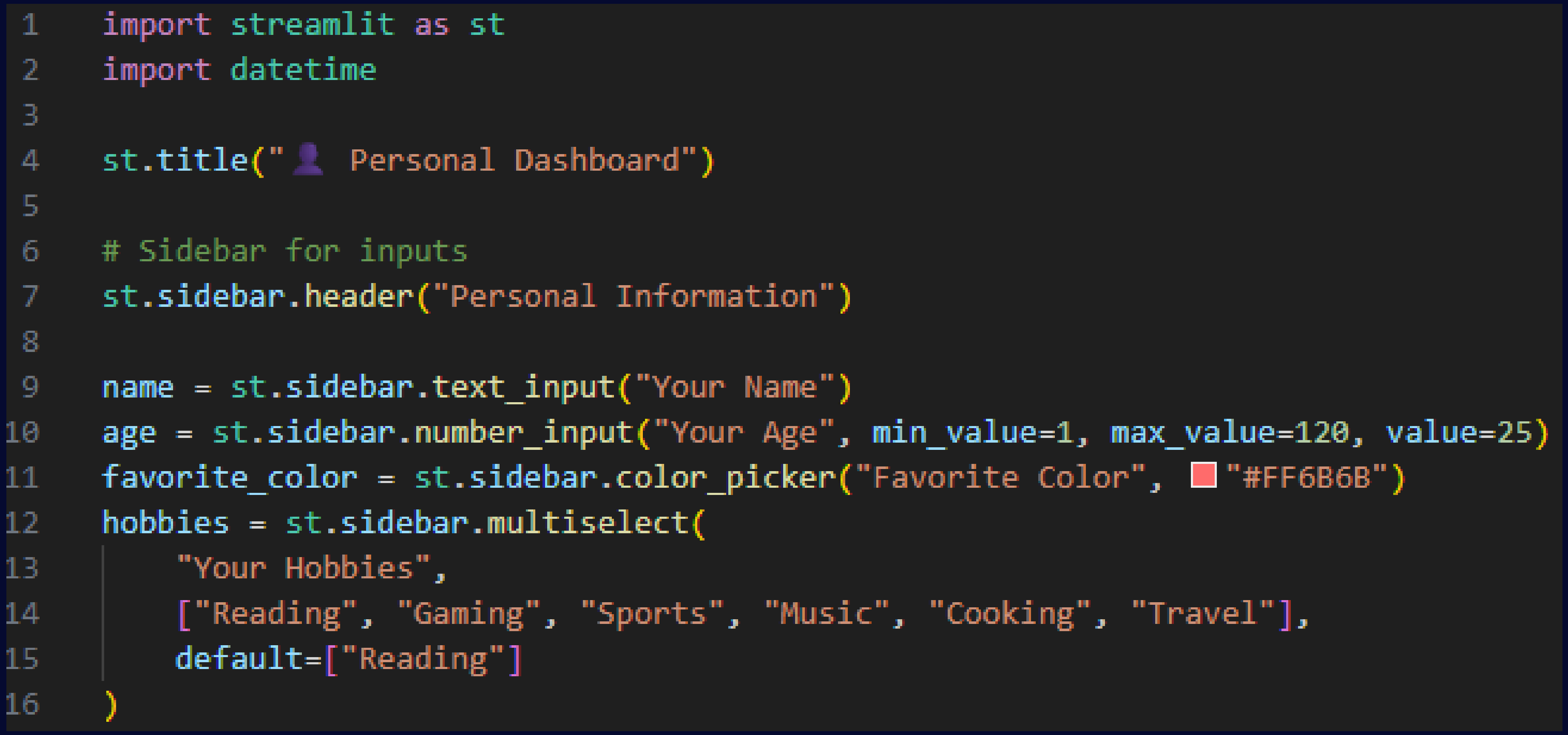
STREAMLIT

- Open-source Python framework that allows developers to create user interface(UI)
- Rapid Prototyping
- pip install streamlit

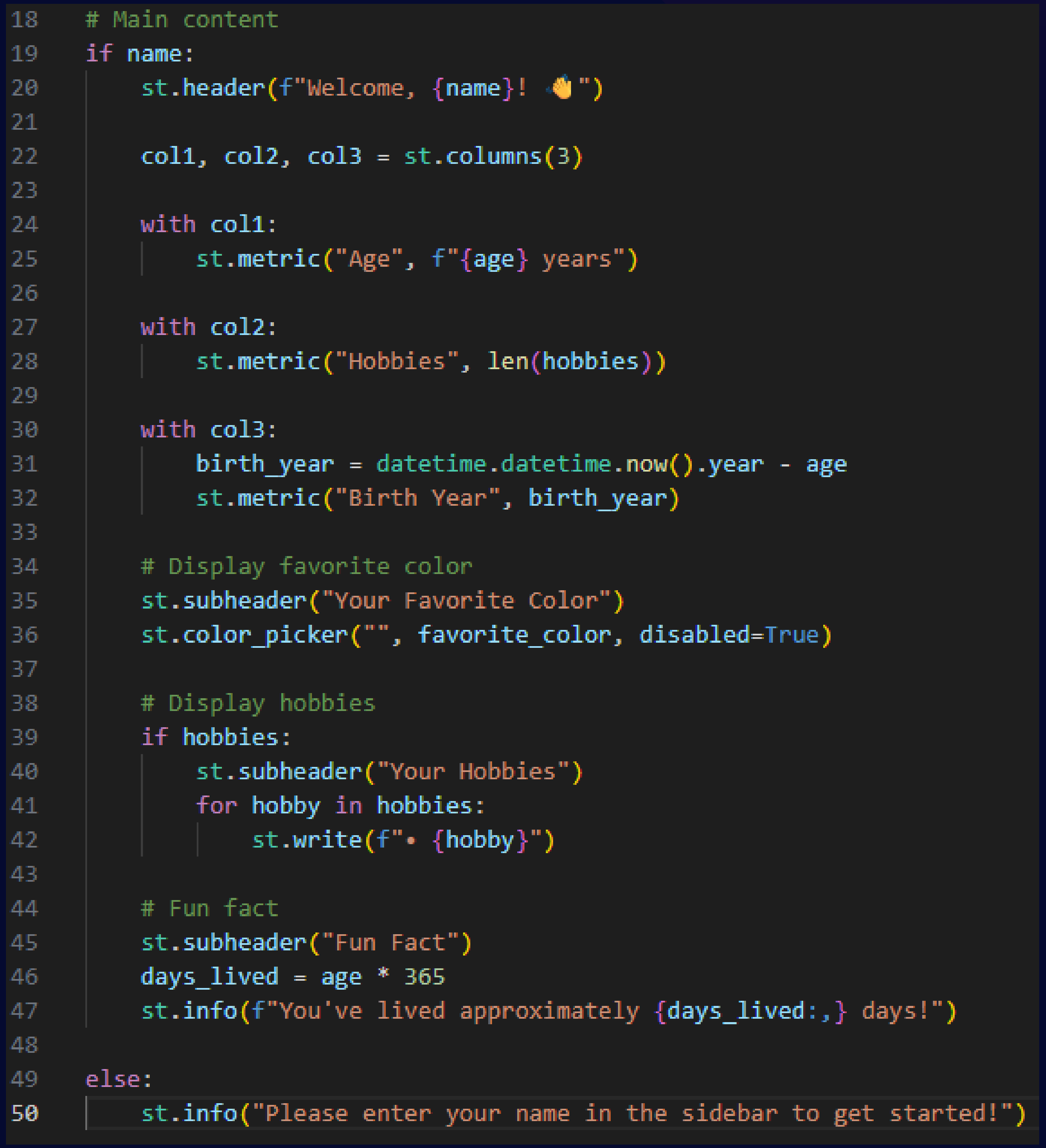
STREAMLIT

JOMHACK
INSPIRE INNOVATION

Example:



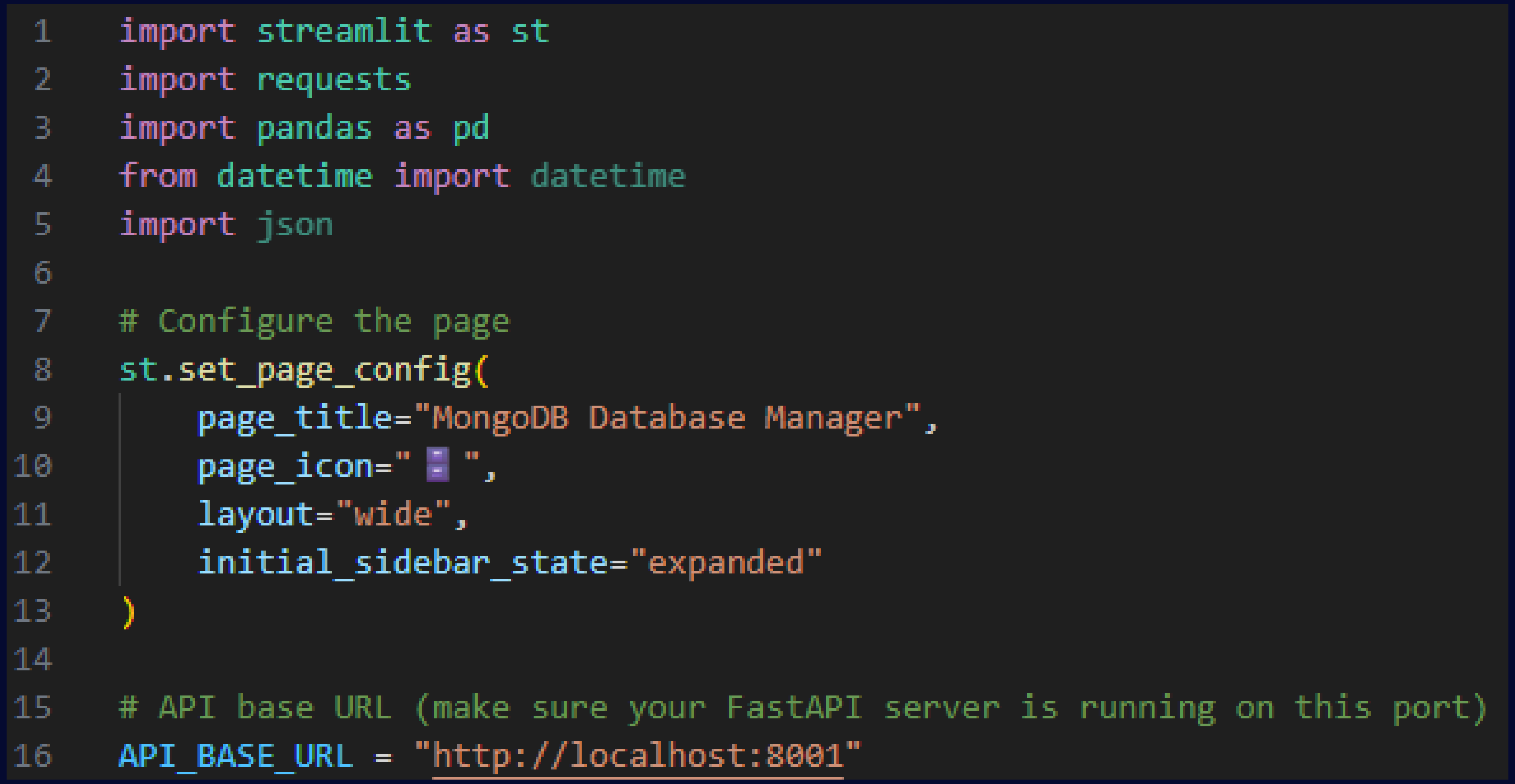
```python
import streamlit as st
import datetime

st.title("👤 Personal Dashboard")

# Sidebar for inputs
st.sidebar.header("Personal Information")

name = st.sidebar.text_input("Your Name")
age = st.sidebar.number_input("Your Age", min_value=1, max_value=120, value=25)
favorite_color = st.sidebar.color_picker("Favorite Color", "#FF6B6B")
hobbies = st.sidebar.multiselect(
    "Your Hobbies",
    ["Reading", "Gaming", "Sports", "Music", "Cooking", "Travel"],
    default=["Reading"]
)
```



```python
# Main content
if name:
    st.header(f"Welcome, {name}! 👋")

    col1, col2, col3 = st.columns(3)

    with col1:
        st.metric("Age", f"{age} years")

    with col2:
        st.metric("Hobbies", len(hobbies))

    with col3:
        birth_year = datetime.datetime.now().year - age
        st.metric("Birth Year", birth_year)

    # Display favorite color
    st.subheader("Your Favorite Color")
    st.color_picker("", favorite_color, disabled=True)

    # Display hobbies
    if hobbies:
        st.subheader("Your Hobbies")
        for hobby in hobbies:
            st.write(f"• {hobby}")

    # Fun fact
    st.subheader("Fun Fact")
    days_lived = age * 365
    st.info(f"You've lived approximately {days_lived:,} days!")

else:
    st.info("Please enter your name in the sidebar to get started!")
```

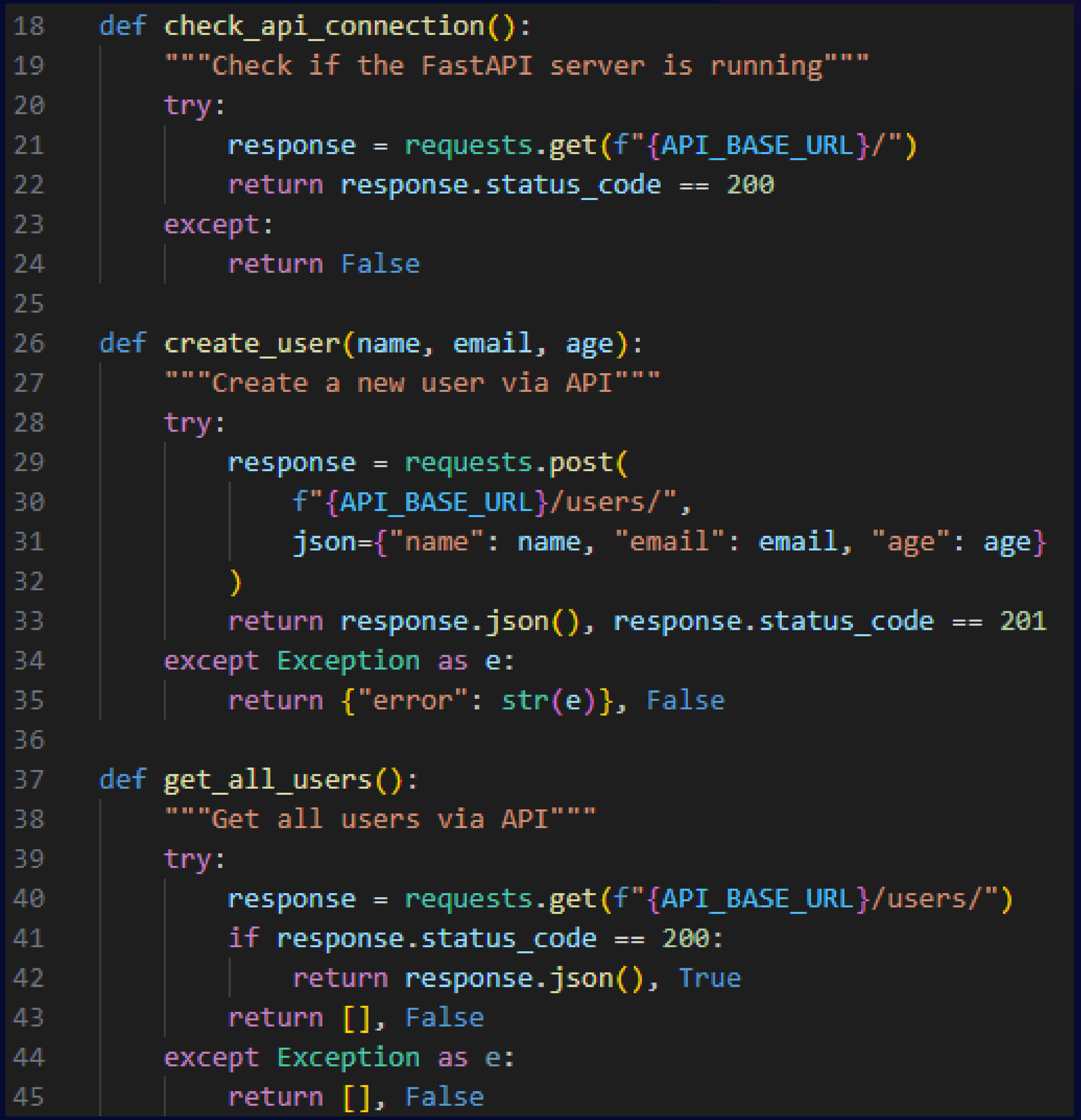
STREAMLIT (MONGO)

Imports & Configuration:



```python
import streamlit as st
import requests
import pandas as pd
from datetime import datetime
import json

# Configure the page
st.set_page_config(
    page_title="MongoDB Database Manager",
    page_icon="🗄️",
    layout="wide",
    initial_sidebar_state="expanded"
)

# API base URL (make sure your FastAPI server is running on this port)
API_BASE_URL = "http://localhost:8001"
```

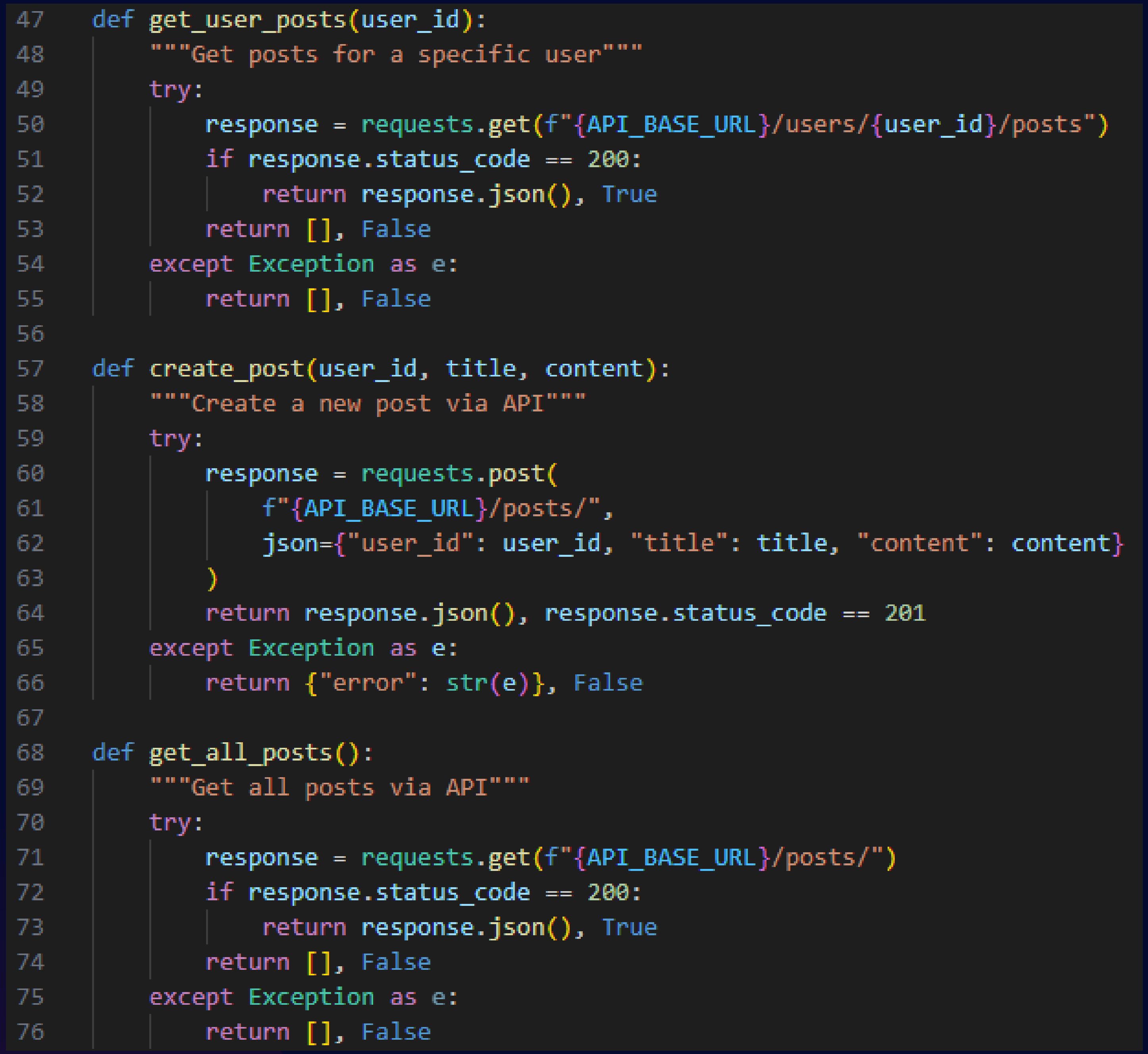
Functions:



```python
def check_api_connection():
    """Check if the FastAPI server is running"""
    try:
        response = requests.get(f"{API_BASE_URL}/")
        return response.status_code == 200
    except:
        return False

def create_user(name, email, age):
    """Create a new user via API"""
    try:
        response = requests.post(
            f"{API_BASE_URL}/users/",
            json={"name": name, "email": email, "age": age}
        )
        return response.json(), response.status_code == 201
    except Exception as e:
        return {"error": str(e)}, False

def get_all_users():
    """Get all users via API"""
    try:
        response = requests.get(f"{API_BASE_URL}/users/")
        if response.status_code == 200:
            return response.json(), True
        return [], False
    except Exception as e:
        return [], False
```

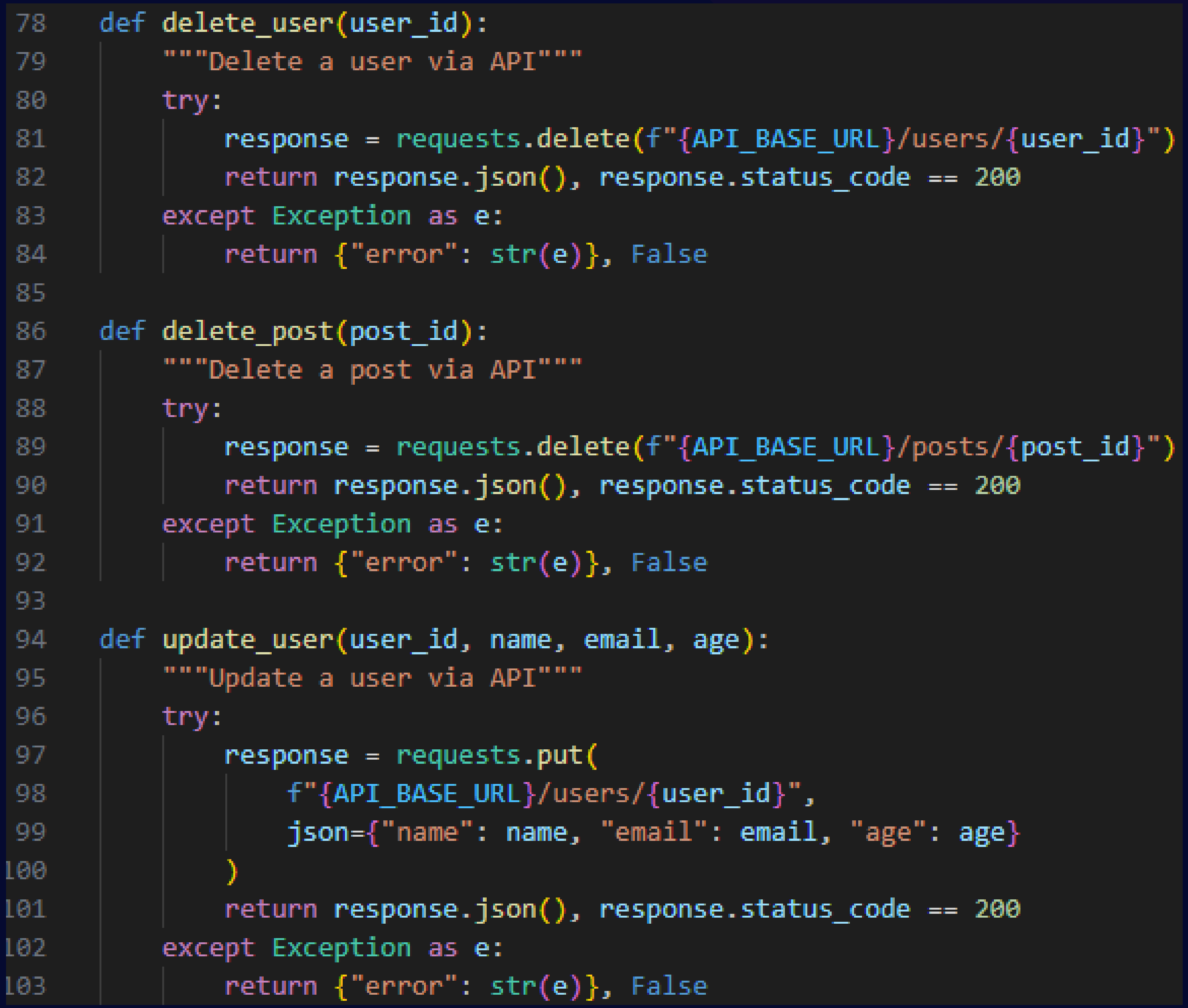
STREAMLIT (MONGO)

Functions:



```python
def get_user_posts(user_id):
    """Get posts for a specific user"""
    try:
        response = requests.get(f"{API_BASE_URL}/users/{user_id}/posts")
        if response.status_code == 200:
            return response.json(), True
        return [], False
    except Exception as e:
        return [], False

def create_post(user_id, title, content):
    """Create a new post via API"""
    try:
        response = requests.post(
            f"{API_BASE_URL}/posts/",
            json={"user_id": user_id, "title": title, "content": content}
        )
        return response.json(), response.status_code == 201
    except Exception as e:
        return {"error": str(e)}, False

def get_all_posts():
    """Get all posts via API"""
    try:
        response = requests.get(f"{API_BASE_URL}/posts/")
        if response.status_code == 200:
            return response.json(), True
        return [], False
    except Exception as e:
        return [], False
```

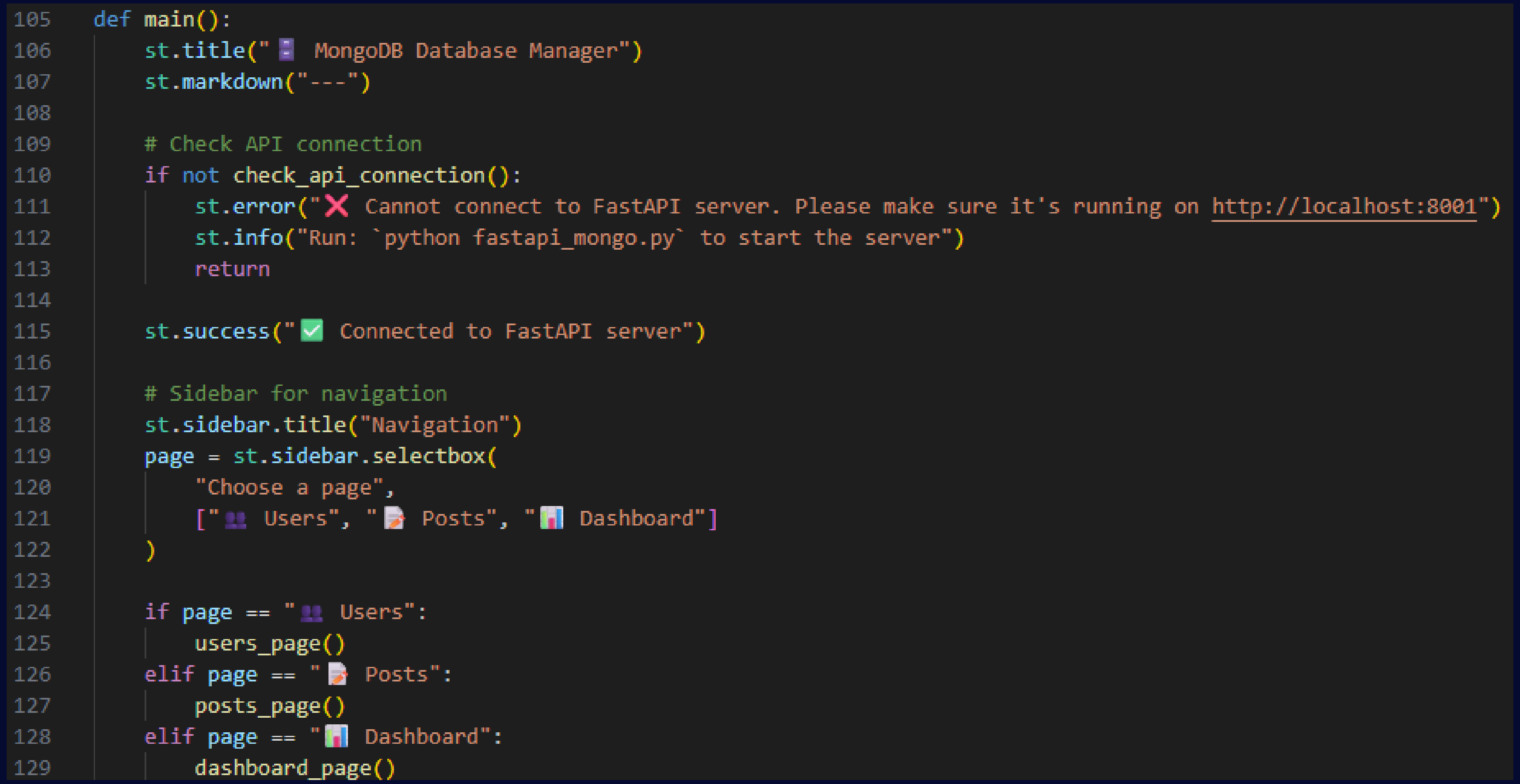
Functions:



```python
def delete_user(user_id):
    """Delete a user via API"""
    try:
        response = requests.delete(f"{API_BASE_URL}/users/{user_id}")
        return response.json(), response.status_code == 200
    except Exception as e:
        return {"error": str(e)}, False

def delete_post(post_id):
    """Delete a post via API"""
    try:
        response = requests.delete(f"{API_BASE_URL}/posts/{post_id}")
        return response.json(), response.status_code == 200
    except Exception as e:
        return {"error": str(e)}, False

def update_user(user_id, name, email, age):
    """Update a user via API"""
    try:
        response = requests.put(
            f"{API_BASE_URL}/users/{user_id}",
            json={"name": name, "email": email, "age": age}
        )
        return response.json(), response.status_code == 200
    except Exception as e:
        return {"error": str(e)}, False
```

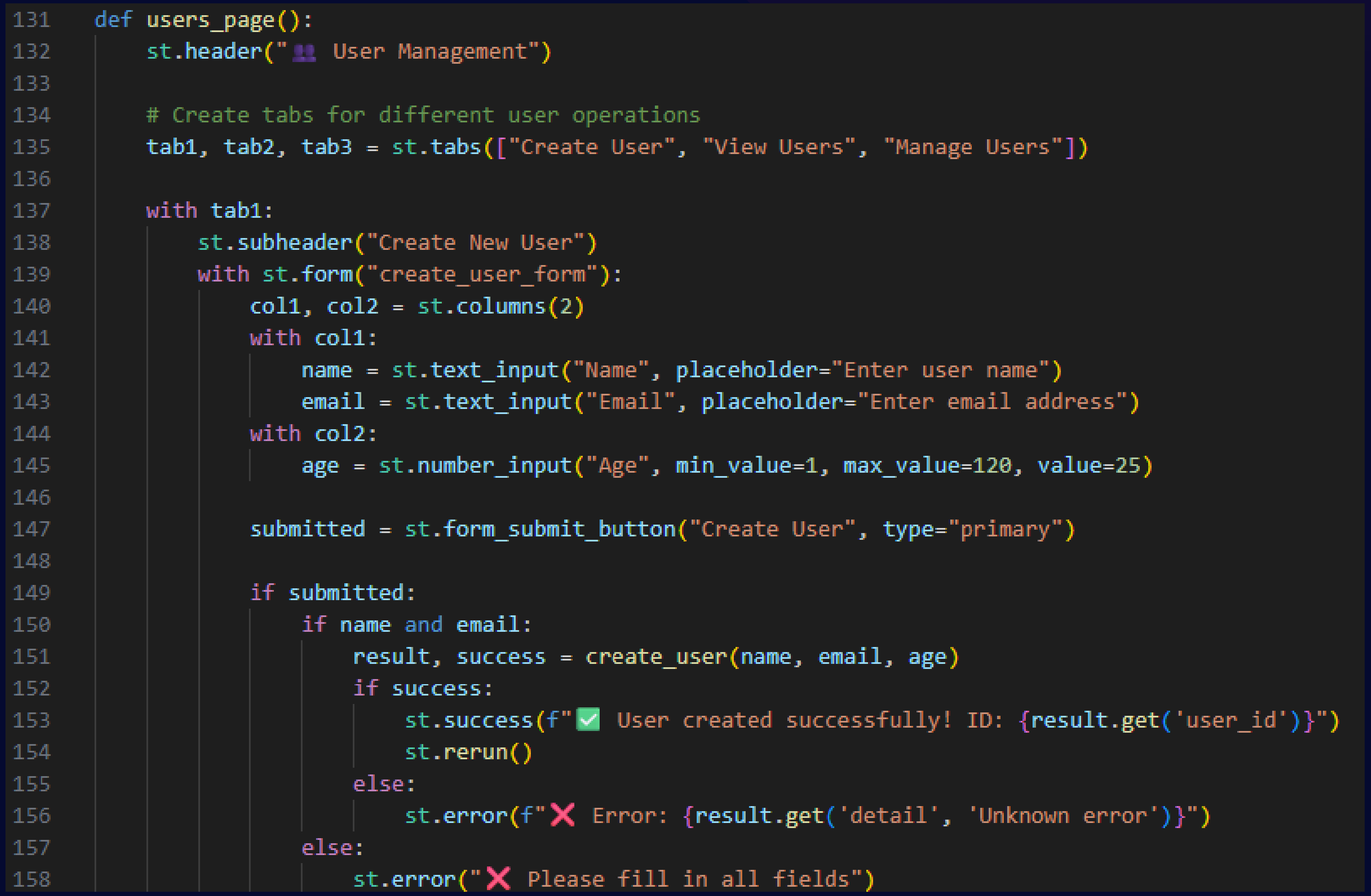
STREAMLIT (MONGO)

Functions:



```python
def main():
    st.title("🗄️ MongoDB Database Manager")
    st.markdown("---")

    # Check API connection
    if not check_api_connection():
        st.error("❌ Cannot connect to FastAPI server. Please make sure it's running on http://localhost:8001")
        st.info("Run: `python fastapi_mongo.py` to start the server")
        return

    st.success("✅ Connected to FastAPI server")

    # Sidebar for navigation
    st.sidebar.title("Navigation")
    page = st.sidebar.selectbox(
        "Choose a page",
        ["👥 Users", "📝 Posts", "📊 Dashboard"]
    )

    if page == "👥 Users":
        users_page()
    elif page == "📝 Posts":
        posts_page()
    elif page == "📊 Dashboard":
        dashboard_page()
```

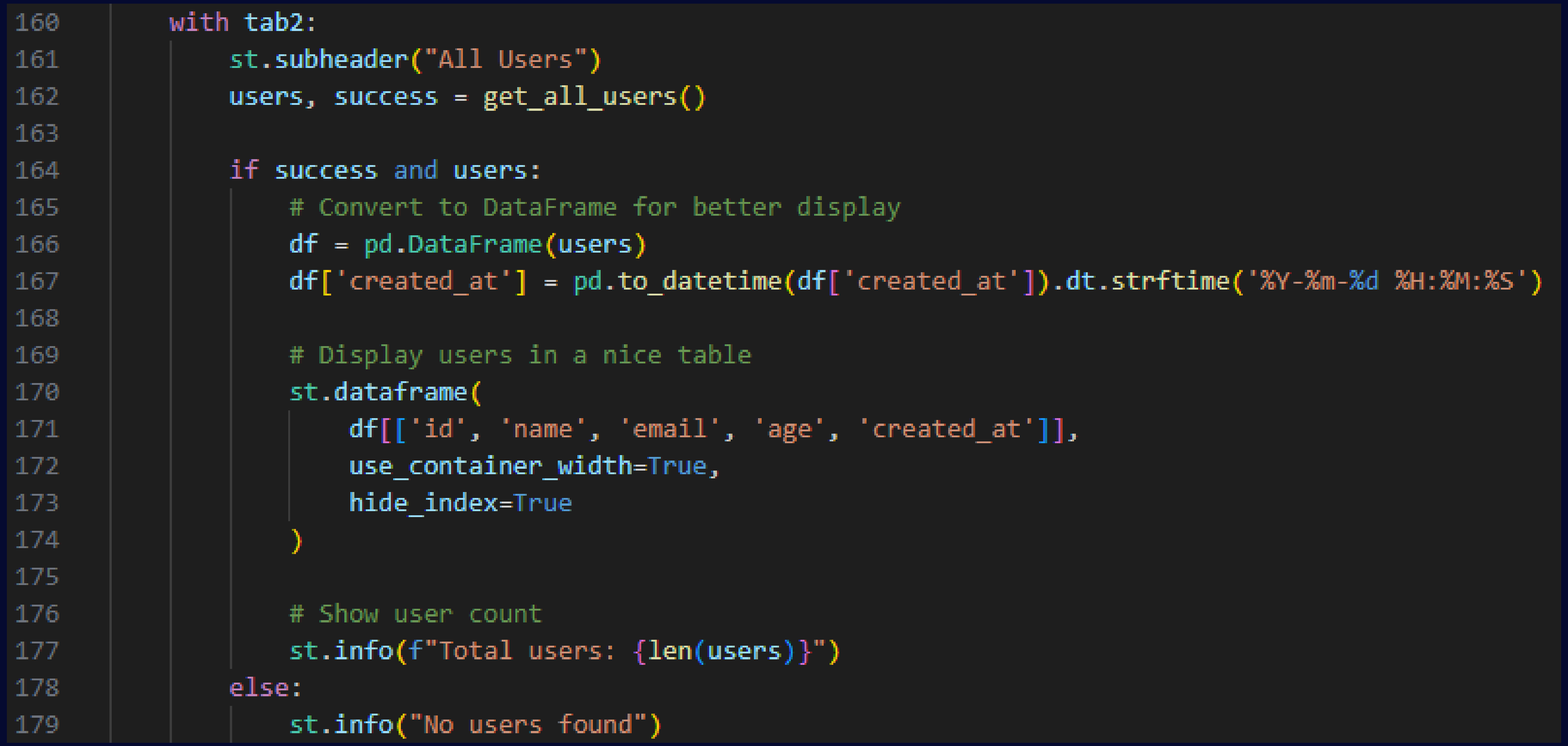
Functions:



```python
def users_page():
    st.header("👥 User Management")

    # Create tabs for different user operations
    tab1, tab2, tab3 = st.tabs(["Create User", "View Users", "Manage Users"])

    with tab1:
        st.subheader("Create New User")
        with st.form("create_user_form"):
            col1, col2 = st.columns(2)
            with col1:
                name = st.text_input("Name", placeholder="Enter user name")
                email = st.text_input("Email", placeholder="Enter email address")
            with col2:
                age = st.number_input("Age", min_value=1, max_value=120, value=25)

            submitted = st.form_submit_button("Create User", type="primary")

            if submitted:
                if name and email:
                    result, success = create_user(name, email, age)
                    if success:
                        st.success(f"✅ User created successfully! ID: {result.get('user_id')}")
                        st.rerun()
                    else:
                        st.error(f"❌ Error: {result.get('detail', 'Unknown error')}")
                else:
                    st.error("❌ Please fill in all fields")
```

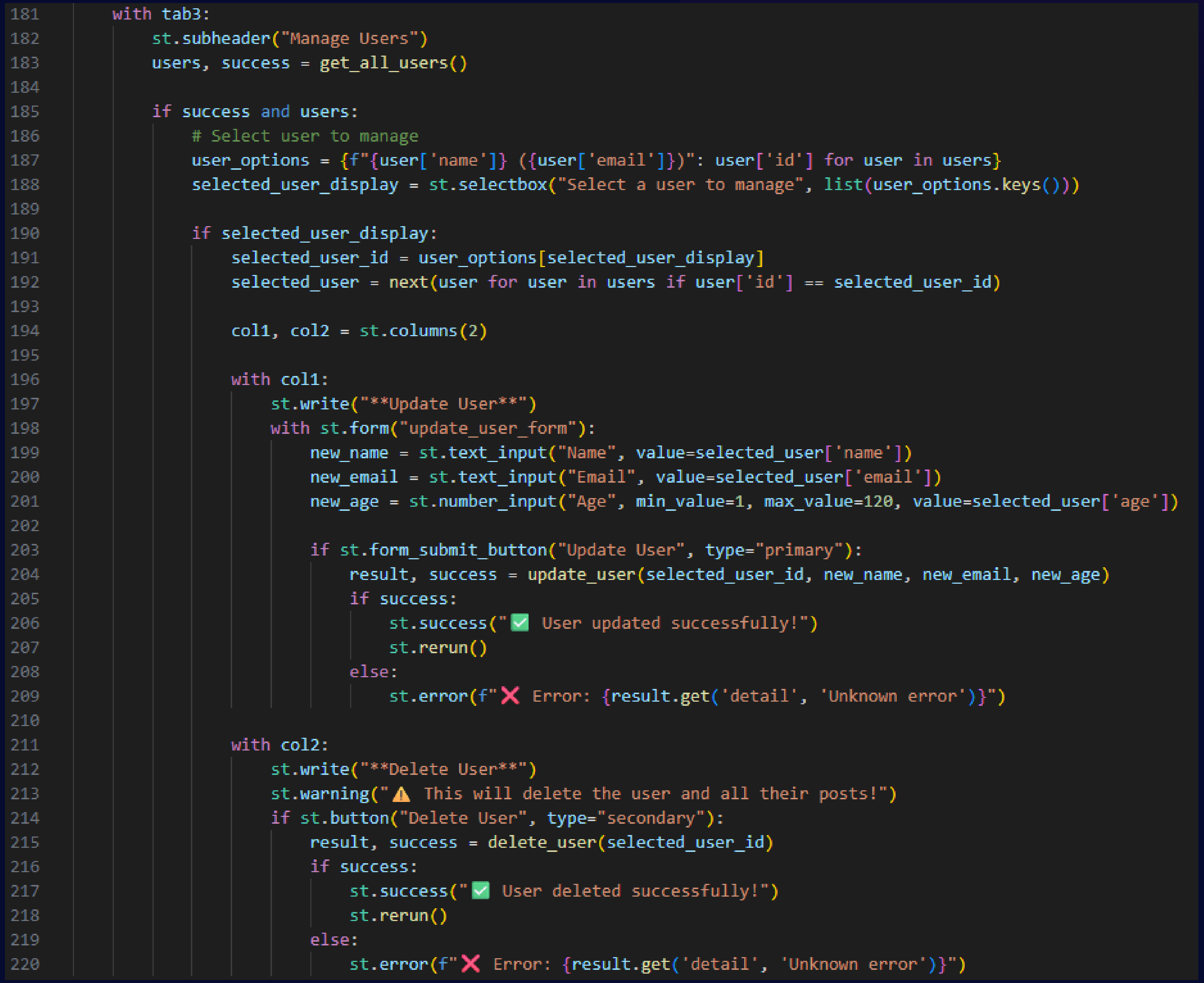
STREAMLIT (MONGO)

Functions:



```python
    with tab2:
        st.subheader("All Users")
        users, success = get_all_users()

        if success and users:
            # Convert to DataFrame for better display
            df = pd.DataFrame(users)
            df['created_at'] = pd.to_datetime(df['created_at']).dt.strftime('%Y-%m-%d %H:%M:%S')

            # Display users in a nice table
            st.dataframe(
                df[['id', 'name', 'email', 'age', 'created_at']],
                use_container_width=True,
                hide_index=True
            )

            # Show user count
            st.info(f"Total users: {len(users)}")
        else:
            st.info("No users found")
```

Functions:

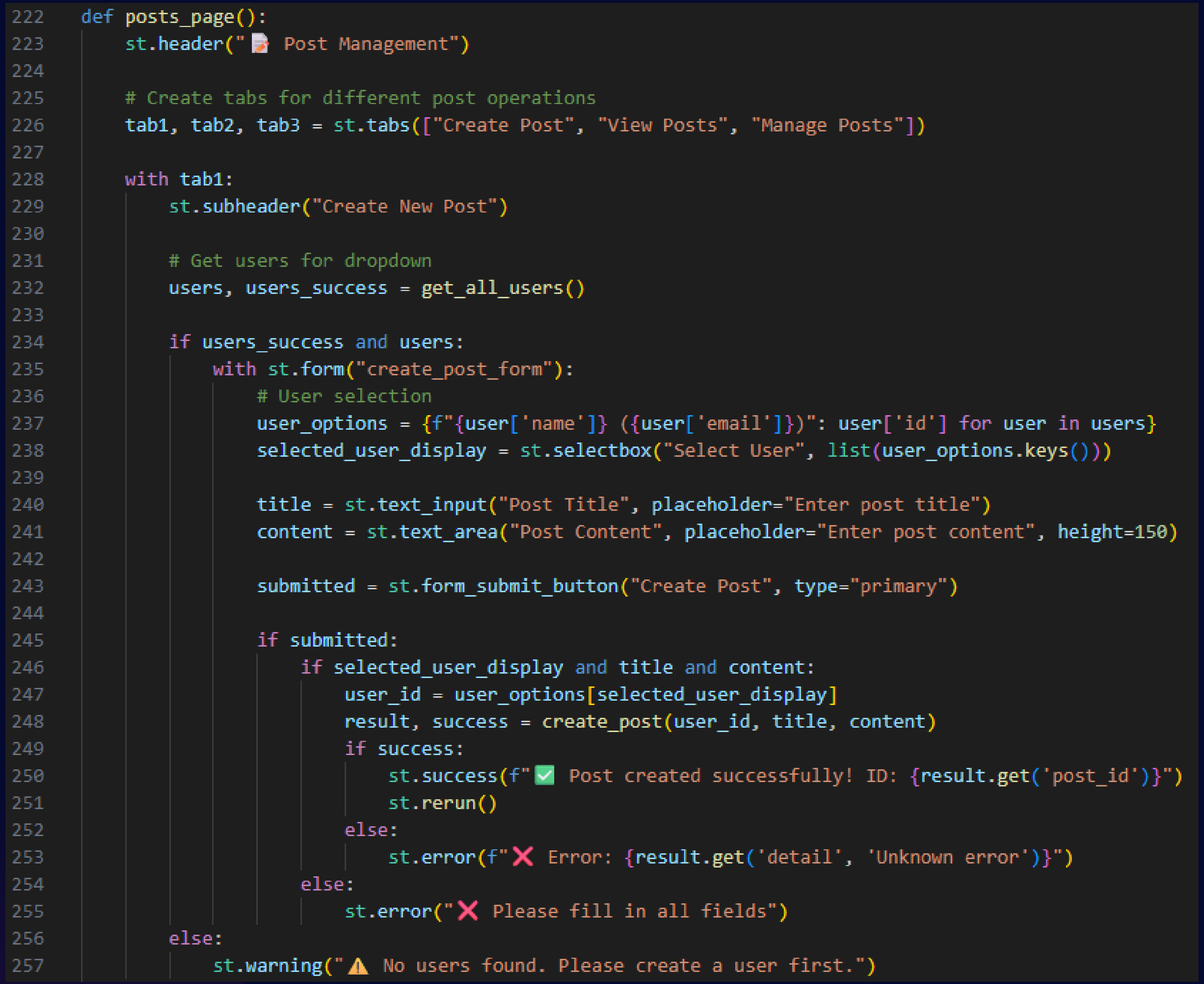


```python
    with tab3:
        st.subheader("Manage Users")
        users, success = get_all_users()

        if success and users:
            # Select user to manage
            user_options = {f"{user['name']} ({user['email']})": user['id'] for user in users}
            selected_user_display = st.selectbox("Select a user to manage", list(user_options.keys()))

            if selected_user_display:
                selected_user_id = user_options[selected_user_display]
                selected_user = next(user for user in users if user['id'] == selected_user_id)

                col1, col2 = st.columns(2)

                with col1:
                    st.write("**Update User**")
                    with st.form("update_user_form"):
                        new_name = st.text_input("Name", value=selected_user['name'])
                        new_email = st.text_input("Email", value=selected_user['email'])
                        new_age = st.number_input("Age", min_value=1, max_value=120, value=selected_user['age'])

                        if st.form_submit_button("Update User", type="primary"):
                            result, success = update_user(selected_user_id, new_name, new_email, new_age)
                            if success:
                                st.success("✅ User updated successfully!")
                                st.rerun()
                            else:
                                st.error(f"❌ Error: {result.get('detail', 'Unknown error')}")

                with col2:
                    st.write("**Delete User**")
                    st.warning("⚠️ This will delete the user and all their posts!")
                    if st.button("Delete User", type="secondary"):
                        result, success = delete_user(selected_user_id)
                        if success:
                            st.success("✅ User deleted successfully!")
                            st.rerun()
                        else:
                            st.error(f"❌ Error: {result.get('detail', 'Unknown error')}")
```

STREAMLIT (MONGO)

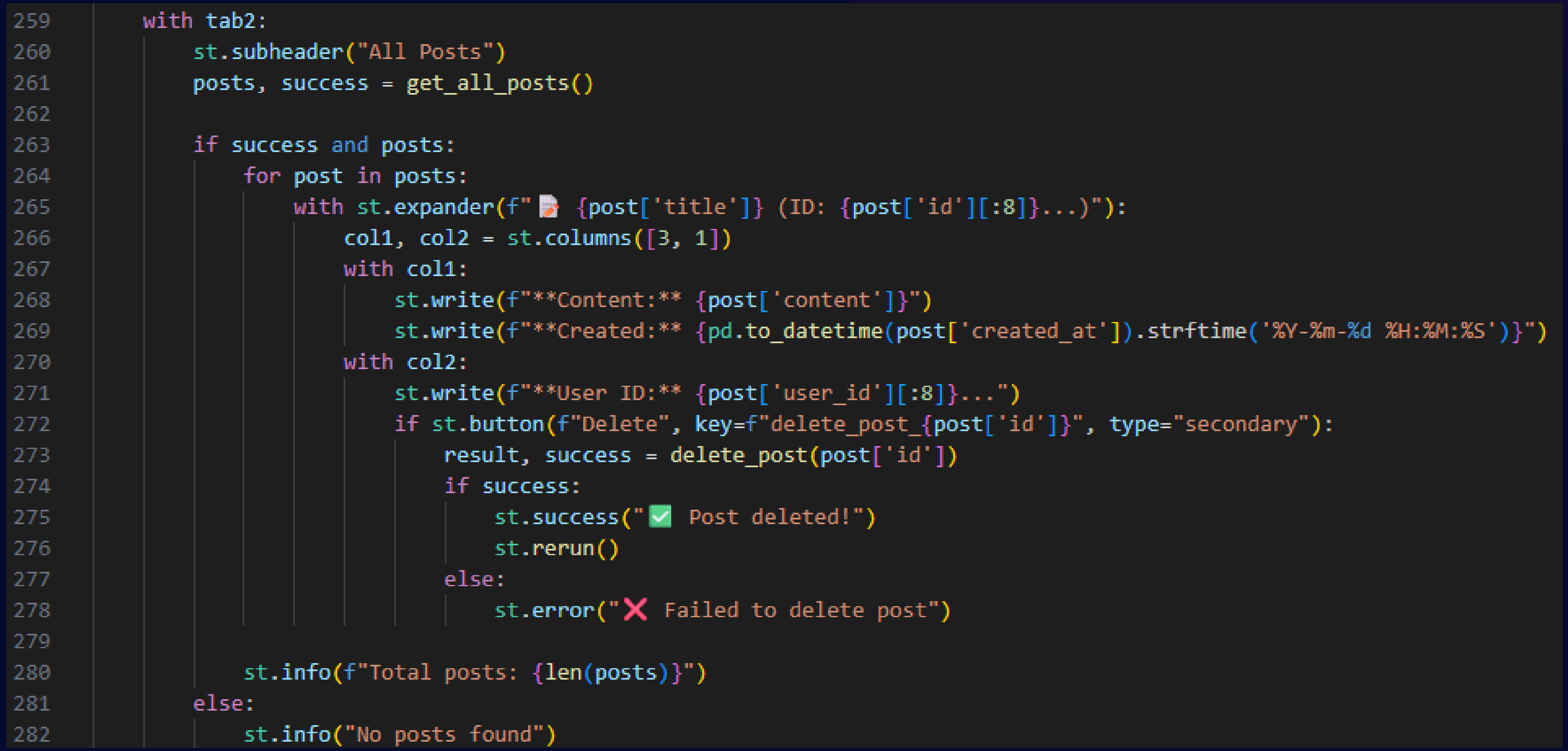
JOMHACK INSPIRE INNOVATION

Functions:



```python
def posts_page():
    st.header("📝 Post Management")

    # Create tabs for different post operations
    tab1, tab2, tab3 = st.tabs(["Create Post", "View Posts", "Manage Posts"])

    with tab1:
        st.subheader("Create New Post")

        # Get users for dropdown
        users, users_success = get_all_users()

        if users_success and users:
            with st.form("create_post_form"):
                # User selection
                user_options = {f"{user['name']} ({user['email']})": user['id'] for user in users}
                selected_user_display = st.selectbox("Select User", list(user_options.keys()))

                title = st.text_input("Post Title", placeholder="Enter post title")
                content = st.text_area("Post Content", placeholder="Enter post content", height=150)

                submitted = st.form_submit_button("Create Post", type="primary")

                if submitted:
                    if selected_user_display and title and content:
                        user_id = user_options[selected_user_display]
                        result, success = create_post(user_id, title, content)
                        if success:
                            st.success(f"✅ Post created successfully! ID: {result.get('post_id')}")
                            st.rerun()
                        else:
                            st.error(f"❌ Error: {result.get('detail', 'Unknown error')}")
                    else:
                        st.error("❌ Please fill in all fields")
        else:
            st.warning("⚠️ No users found. Please create a user first.")
```

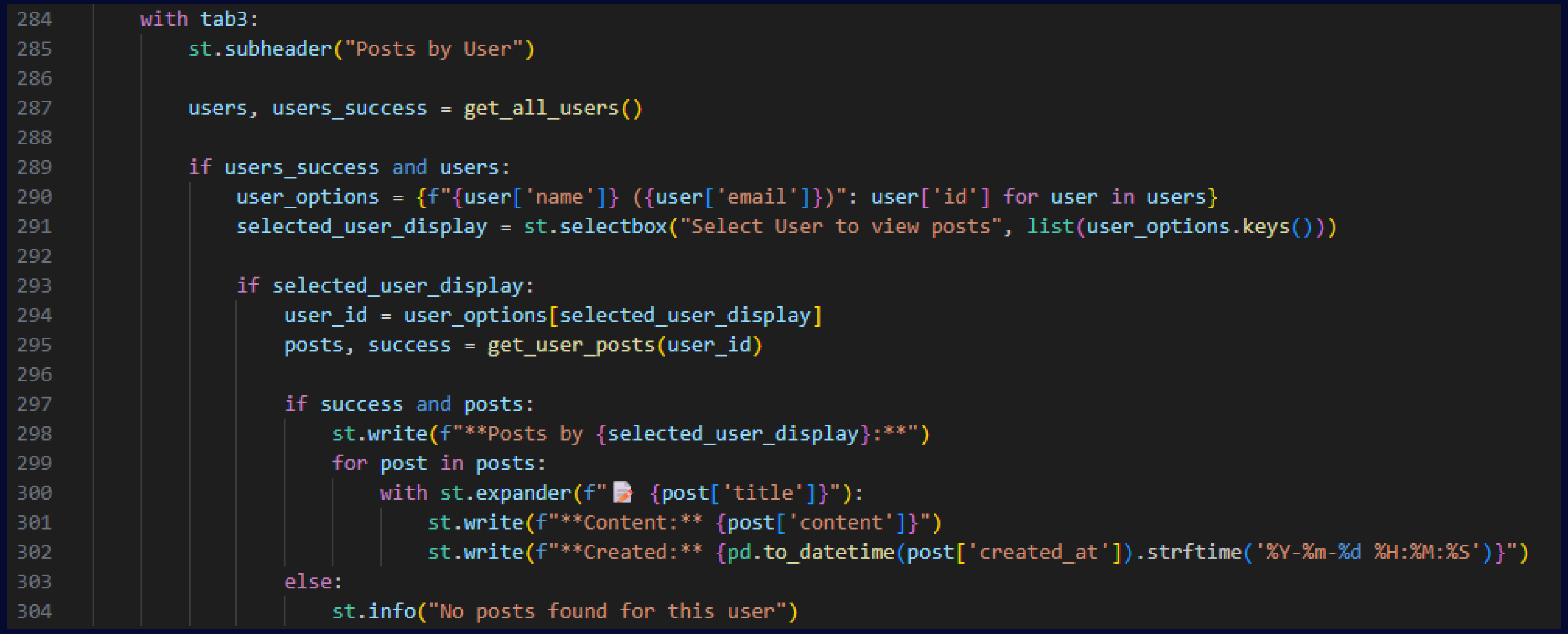
Functions:



```python
    with tab2:
        st.subheader("All Posts")
        posts, success = get_all_posts()

        if success and posts:
            for post in posts:
                with st.expander(f"📝 {post['title']} (ID: {post['id'][:8]}...)"):
                    col1, col2 = st.columns([3, 1])
                    with col1:
                        st.write(f"**Content:** {post['content']}")
                        st.write(f"**Created:** {pd.to_datetime(post['created_at']).strftime('%Y-%m-%d %H:%M:%S')}")
                    with col2:
                        st.write(f"**User ID:** {post['user_id'][:8]}...")
                        if st.button(f"Delete", key=f"delete_post_{post['id']}", type="secondary"):
                            result, success = delete_post(post['id'])
                            if success:
                                st.success("✅ Post deleted!")
                                st.rerun()
                            else:
                                st.error("❌ Failed to delete post")

            st.info(f"Total posts: {len(posts)}")
        else:
            st.info("No posts found")
```

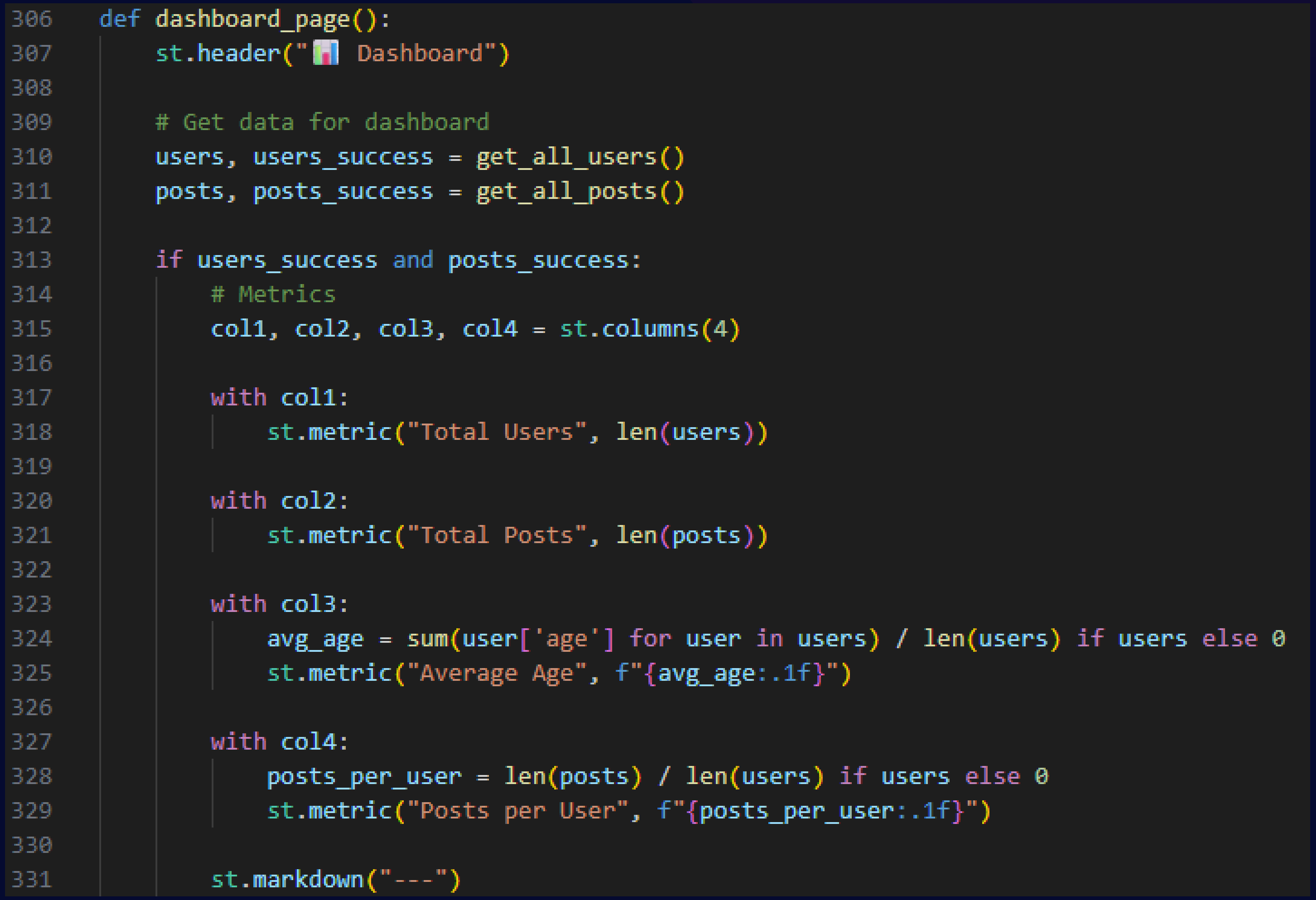
STREAMLIT (MONGO)

Functions:



```python
    with tab3:
        st.subheader("Posts by User")

        users, users_success = get_all_users()

        if users_success and users:
            user_options = {f"{user['name']} ({user['email']})": user['id'] for user in users}
            selected_user_display = st.selectbox("Select User to view posts", list(user_options.keys()))

            if selected_user_display:
                user_id = user_options[selected_user_display]
                posts, success = get_user_posts(user_id)

                if success and posts:
                    st.write(f"**Posts by {selected_user_display}:**")
                    for post in posts:
                        with st.expander(f"📝 {post['title']}"):
                            st.write(f"**Content:** {post['content']}")
                            st.write(f"**Created:** {pd.to_datetime(post['created_at']).strftime('%Y-%m-%d %H:%M:%S')}")
                else:
                    st.info("No posts found for this user")
```

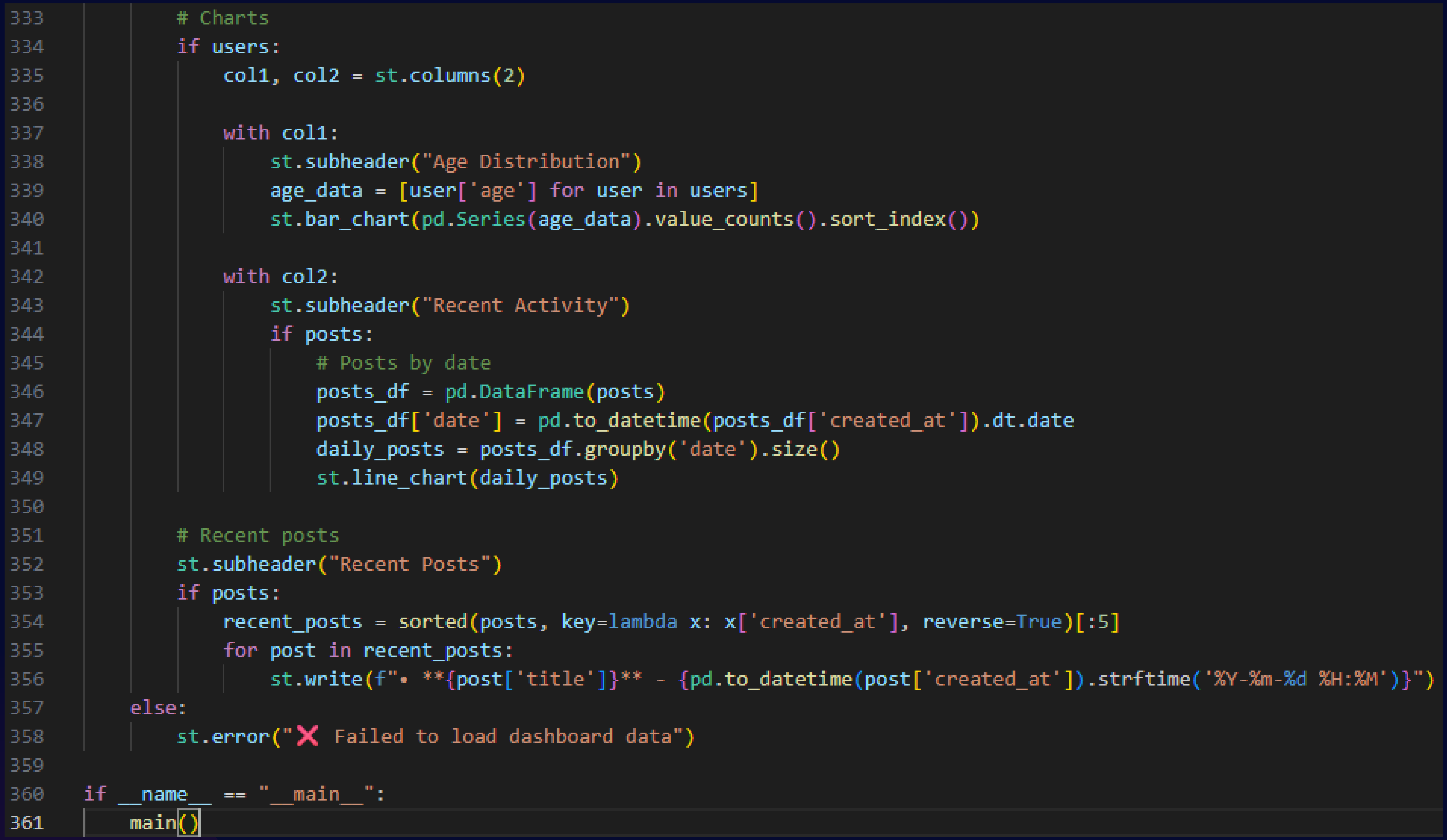
Functions:



```python
def dashboard_page():
    st.header("📊 Dashboard")

    # Get data for dashboard
    users, users_success = get_all_users()
    posts, posts_success = get_all_posts()

    if users_success and posts_success:
        # Metrics
        col1, col2, col3, col4 = st.columns(4)

        with col1:
            st.metric("Total Users", len(users))

        with col2:
            st.metric("Total Posts", len(posts))

        with col3:
            avg_age = sum(user['age'] for user in users) / len(users) if users else 0
            st.metric("Average Age", f"{avg_age:.1f}")

        with col4:
            posts_per_user = len(posts) / len(users) if users else 0
            st.metric("Posts per User", f"{posts_per_user:.1f}")

        st.markdown("---")
```

STREAMLIT (MONGO)

Functions:



```
        # Charts
        if users:
            col1, col2 = st.columns(2)

            with col1:
                st.subheader("Age Distribution")
                age_data = [user['age'] for user in users]
                st.bar_chart(pd.Series(age_data).value_counts().sort_index())

            with col2:
                st.subheader("Recent Activity")
                if posts:
                    # Posts by date
                    posts_df = pd.DataFrame(posts)
                    posts_df['date'] = pd.to_datetime(posts_df['created_at']).dt.date
                    daily_posts = posts_df.groupby('date').size()
                    st.line_chart(daily_posts)

        # Recent posts
        st.subheader("Recent Posts")
        if posts:
            recent_posts = sorted(posts, key=lambda x: x['created_at'], reverse=True)[:5]
            for post in recent_posts:
                st.write(f"• **{post['title']}** - {pd.to_datetime(post['created_at']).strftime('%Y-%m-%d %H:%M')}")
    else:
        st.error("❌ Failed to load dashboard data")

if __name__ == "__main__":
    main()
```