



# PYTHON BOOTCAMP

[www.jomhack.com](http://www.jomhack.com)

# STREAMLIT

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

# STREAMLIT



## Example:

```
1  import streamlit as st
2  import datetime
3
4  st.title("👤 Personal Dashboard")
5
6  # Sidebar for inputs
7  st.sidebar.header("Personal Information")
8
9  name = st.sidebar.text_input("Your Name")
10 age = st.sidebar.number_input("Your Age", min_value=1, max_value=120, value=25)
11 favorite_color = st.sidebar.color_picker("Favorite Color", "#FF6B6B")
12 hobbies = st.sidebar.multiselect(
13     "Your Hobbies",
14     ["Reading", "Gaming", "Sports", "Music", "Cooking", "Travel"],
15     default=["Reading"]
16 )
```

```
18 # Main content
19 if name:
20     st.header(f"Welcome, {name}! 🎉")
21
22     col1, col2, col3 = st.columns(3)
23
24     with col1:
25         st.metric("Age", f"{age} years")
26
27     with col2:
28         st.metric("Hobbies", len(hobbies))
29
30     with col3:
31         birth_year = datetime.datetime.now().year - age
32         st.metric("Birth Year", birth_year)
33
34     # Display favorite color
35     st.subheader("Your Favorite Color")
36     st.color_picker("", favorite_color, disabled=True)
37
38     # Display hobbies
39     if hobbies:
40         st.subheader("Your Hobbies")
41         for hobby in hobbies:
42             st.write(f"• {hobby}")
43
44     # Fun fact
45     st.subheader("Fun Fact")
46     days_lived = age * 365
47     st.info(f"You've lived approximately {days_lived:,} days!")
48
49 else:
50     st.info("Please enter your name in the sidebar to get started!")
```

# STREAMLIT (MONGO)



## Imports & Configuration:

```
1  import streamlit as st
2  import requests
3  import pandas as pd
4  from datetime import datetime
5  import json
6
7  # Configure the page
8  st.set_page_config(
9      page_title="MongoDB Database Manager",
10     page_icon="🗄️",
11     layout="wide",
12     initial_sidebar_state="expanded"
13 )
14
15 # API base URL (make sure your FastAPI server is running on this port)
16 API_BASE_URL = "http://localhost:8001"
```

## Functions:

```
18 def check_api_connection():
19     """Check if the FastAPI server is running"""
20     try:
21         response = requests.get(f"{API_BASE_URL}/")
22         return response.status_code == 200
23     except:
24         return False
25
26 def create_user(name, email, age):
27     """Create a new user via API"""
28     try:
29         response = requests.post(
30             f"{API_BASE_URL}/users/",
31             json={"name": name, "email": email, "age": age}
32         )
33         return response.json(), response.status_code == 201
34     except Exception as e:
35         return {"error": str(e)}, False
36
37 def get_all_users():
38     """Get all users via API"""
39     try:
40         response = requests.get(f"{API_BASE_URL}/users/")
41         if response.status_code == 200:
42             return response.json(), True
43         return [], False
44     except Exception as e:
45         return [], False
```

# STREAMLIT (MONGO)



## Functions:

```
47 def get_user_posts(user_id):
48     """Get posts for a specific user"""
49     try:
50         response = requests.get(f"{API_BASE_URL}/users/{user_id}/posts")
51         if response.status_code == 200:
52             return response.json(), True
53         return [], False
54     except Exception as e:
55         return [], False
56
57 def create_post(user_id, title, content):
58     """Create a new post via API"""
59     try:
60         response = requests.post(
61             f"{API_BASE_URL}/posts/",
62             json={"user_id": user_id, "title": title, "content": content}
63         )
64         return response.json(), response.status_code == 201
65     except Exception as e:
66         return {"error": str(e)}, False
67
68 def get_all_posts():
69     """Get all posts via API"""
70     try:
71         response = requests.get(f"{API_BASE_URL}/posts/")
72         if response.status_code == 200:
73             return response.json(), True
74         return [], False
75     except Exception as e:
76         return [], False
```

## Functions:

```
78 def delete_user(user_id):
79     """Delete a user via API"""
80     try:
81         response = requests.delete(f"{API_BASE_URL}/users/{user_id}")
82         return response.json(), response.status_code == 200
83     except Exception as e:
84         return {"error": str(e)}, False
85
86 def delete_post(post_id):
87     """Delete a post via API"""
88     try:
89         response = requests.delete(f"{API_BASE_URL}/posts/{post_id}")
90         return response.json(), response.status_code == 200
91     except Exception as e:
92         return {"error": str(e)}, False
93
94 def update_user(user_id, name, email, age):
95     """Update a user via API"""
96     try:
97         response = requests.put(
98             f"{API_BASE_URL}/users/{user_id}",
99             json={"name": name, "email": email, "age": age}
100         )
101         return response.json(), response.status_code == 200
102     except Exception as e:
103         return {"error": str(e)}, False
```

# STREAMLIT (MONGO)



## Functions:

```
105 def main():
106     st.title("🗄️ MongoDB Database Manager")
107     st.markdown("---")
108
109     # Check API connection
110     if not check_api_connection():
111         st.error("❌ Cannot connect to FastAPI server. Please make sure it's running on http://localhost:8001")
112         st.info("Run: `python fastapi_mongo.py` to start the server")
113         return
114
115     st.success("✅ Connected to FastAPI server")
116
117     # Sidebar for navigation
118     st.sidebar.title("Navigation")
119     page = st.sidebar.selectbox(
120         "Choose a page",
121         ["👤 Users", "📄 Posts", "📊 Dashboard"]
122     )
123
124     if page == "👤 Users":
125         users_page()
126     elif page == "📄 Posts":
127         posts_page()
128     elif page == "📊 Dashboard":
129         dashboard_page()
```

## Functions:

```
131 def users_page():
132     st.header("👤 User Management")
133
134     # Create tabs for different user operations
135     tab1, tab2, tab3 = st.tabs(["Create User", "View Users", "Manage Users"])
136
137     with tab1:
138         st.subheader("Create New User")
139         with st.form("create_user_form"):
140             col1, col2 = st.columns(2)
141             with col1:
142                 name = st.text_input("Name", placeholder="Enter user name")
143                 email = st.text_input("Email", placeholder="Enter email address")
144             with col2:
145                 age = st.number_input("Age", min_value=1, max_value=120, value=25)
146
147             submitted = st.form_submit_button("Create User", type="primary")
148
149             if submitted:
150                 if name and email:
151                     result, success = create_user(name, email, age)
152                     if success:
153                         st.success(f"✅ User created successfully! ID: {result.get('user_id')}")
154                         st.rerun()
155                     else:
156                         st.error(f"❌ Error: {result.get('detail', 'Unknown error')}")
157                 else:
158                     st.error("❌ Please fill in all fields")
```

# STREAMLIT (MONGO)



## Functions:

```
160 with tab2:
161     st.subheader("All Users")
162     users, success = get_all_users()
163
164     if success and users:
165         # Convert to DataFrame for better display
166         df = pd.DataFrame(users)
167         df['created_at'] = pd.to_datetime(df['created_at']).dt.strftime('%Y-%m-%d %H:%M:%S')
168
169         # Display users in a nice table
170         st.dataframe(
171             df[['id', 'name', 'email', 'age', 'created_at']],
172             use_container_width=True,
173             hide_index=True
174         )
175
176         # Show user count
177         st.info(f"Total users: {len(users)}")
178     else:
179         st.info("No users found")
```

## Functions:

```
181 with tab3:
182     st.subheader("Manage Users")
183     users, success = get_all_users()
184
185     if success and users:
186         # Select user to manage
187         user_options = {f"{user['name']} ({user['email']})": user['id'] for user in users}
188         selected_user_display = st.selectbox("Select a user to manage", list(user_options.keys()))
189
190         if selected_user_display:
191             selected_user_id = user_options[selected_user_display]
192             selected_user = next(user for user in users if user['id'] == selected_user_id)
193
194             col1, col2 = st.columns(2)
195
196             with col1:
197                 st.write("**Update User**")
198                 with st.form("update_user_form"):
199                     new_name = st.text_input("Name", value=selected_user['name'])
200                     new_email = st.text_input("Email", value=selected_user['email'])
201                     new_age = st.number_input("Age", min_value=1, max_value=120, value=selected_user['age'])
202
203                     if st.form_submit_button("Update User", type="primary"):
204                         result, success = update_user(selected_user_id, new_name, new_email, new_age)
205                         if success:
206                             st.success("✅ User updated successfully!")
207                             st.rerun()
208                         else:
209                             st.error(f"❌ Error: {result.get('detail', 'Unknown error')}")
210
211             with col2:
212                 st.write("**Delete User**")
213                 st.warning("⚠️ This will delete the user and all their posts!")
214                 if st.button("Delete User", type="secondary"):
215                     result, success = delete_user(selected_user_id)
216                     if success:
217                         st.success("✅ User deleted successfully!")
218                         st.rerun()
219                     else:
220                         st.error(f"❌ Error: {result.get('detail', 'Unknown error')}")
```



# STREAMLIT (MONGO)



## Functions:

```
222 def posts_page():
223     st.header("📄 Post Management")
224
225     # Create tabs for different post operations
226     tab1, tab2, tab3 = st.tabs(["Create Post", "View Posts", "Manage Posts"])
227
228     with tab1:
229         st.subheader("Create New Post")
230
231         # Get users for dropdown
232         users, users_success = get_all_users()
233
234         if users_success and users:
235             with st.form("create_post_form"):
236                 # User selection
237                 user_options = {f"{user['name']} ({user['email']})": user['id'] for user in users}
238                 selected_user_display = st.selectbox("Select User", list(user_options.keys()))
239
240                 title = st.text_input("Post Title", placeholder="Enter post title")
241                 content = st.text_area("Post Content", placeholder="Enter post content", height=150)
242
243                 submitted = st.form_submit_button("Create Post", type="primary")
244
245                 if submitted:
246                     if selected_user_display and title and content:
247                         user_id = user_options[selected_user_display]
248                         result, success = create_post(user_id, title, content)
249                         if success:
250                             st.success(f"✅ Post created successfully! ID: {result.get('post_id')}")
251                             st.rerun()
252                         else:
253                             st.error(f"❌ Error: {result.get('detail', 'Unknown error')}")
254                     else:
255                         st.error("❌ Please fill in all fields")
256                 else:
257                     st.warning("⚠️ No users found. Please create a user first.")
```

## Functions:

```
259     with tab2:
260         st.subheader("All Posts")
261         posts, success = get_all_posts()
262
263         if success and posts:
264             for post in posts:
265                 with st.expander(f"📄 {post['title']} (ID: {post['id'][:8]}...)"):
266                     col1, col2 = st.columns([3, 1])
267                     with col1:
268                         st.write(f"***Content:** {post['content']}")
269                         st.write(f"***Created:** {pd.to_datetime(post['created_at']).strftime('%Y-%m-%d %H:%M:%S')}")
270                     with col2:
271                         st.write(f"***User ID:** {post['user_id'][:8]}...")
272                         if st.button(f"Delete", key=f"delete_post_{post['id']}", type="secondary"):
273                             result, success = delete_post(post['id'])
274                             if success:
275                                 st.success("✅ Post deleted!")
276                                 st.rerun()
277                             else:
278                                 st.error("❌ Failed to delete post")
279
280                 st.info(f"Total posts: {len(posts)}")
281         else:
282             st.info("No posts found")
```



# STREAMLIT (MONGO)



## Functions:

```
284 with tab3:
285     st.subheader("Posts by User")
286
287     users, users_success = get_all_users()
288
289     if users_success and users:
290         user_options = {f"{user['name']} ({user['email']})": user['id'] for user in users}
291         selected_user_display = st.selectbox("Select User to view posts", list(user_options.keys()))
292
293         if selected_user_display:
294             user_id = user_options[selected_user_display]
295             posts, success = get_user_posts(user_id)
296
297             if success and posts:
298                 st.write(f"***Posts by {selected_user_display}:***")
299                 for post in posts:
300                     with st.expander(f"📄 {post['title']}"):
301                         st.write(f"***Content:** {post['content']}")
302                         st.write(f"***Created:** {pd.to_datetime(post['created_at']).strftime('%Y-%m-%d %H:%M:%S')}")
303             else:
304                 st.info("No posts found for this user")
```

## Functions:

```
306 def dashboard_page():
307     st.header("📊 Dashboard")
308
309     # Get data for dashboard
310     users, users_success = get_all_users()
311     posts, posts_success = get_all_posts()
312
313     if users_success and posts_success:
314         # Metrics
315         col1, col2, col3, col4 = st.columns(4)
316
317         with col1:
318             st.metric("Total Users", len(users))
319
320         with col2:
321             st.metric("Total Posts", len(posts))
322
323         with col3:
324             avg_age = sum(user['age'] for user in users) / len(users) if users else 0
325             st.metric("Average Age", f"{avg_age:.1f}")
326
327         with col4:
328             posts_per_user = len(posts) / len(users) if users else 0
329             st.metric("Posts per User", f"{posts_per_user:.1f}")
330
331     st.markdown("---")
```

# STREAMLIT (MONGO)



## Functions:

```
333     # Charts
334     if users:
335         col1, col2 = st.columns(2)
336
337         with col1:
338             st.subheader("Age Distribution")
339             age_data = [user['age'] for user in users]
340             st.bar_chart(pd.Series(age_data).value_counts().sort_index())
341
342         with col2:
343             st.subheader("Recent Activity")
344             if posts:
345                 # Posts by date
346                 posts_df = pd.DataFrame(posts)
347                 posts_df['date'] = pd.to_datetime(posts_df['created_at']).dt.date
348                 daily_posts = posts_df.groupby('date').size()
349                 st.line_chart(daily_posts)
350
351     # Recent posts
352     st.subheader("Recent Posts")
353     if posts:
354         recent_posts = sorted(posts, key=lambda x: x['created_at'], reverse=True)[:5]
355         for post in recent_posts:
356             st.write(f"• **{post['title']}** - {pd.to_datetime(post['created_at']).strftime('%Y-%m-%d %H:%M')}")
357     else:
358         st.error("❌ Failed to load dashboard data")
359
360 if __name__ == "__main__":
361     main()
```