Redis Meeting App

A **Teams/Zoom-inspired platform** for **physical meetings**: participants can join sessions based on their physical proximity to a meeting location. The app combines **SQLite** for database storage and **Redis** for real-time meeting management, including chatrooms, participant tracking, and location-based discovery.

What this app does

- Create users and meetings
- Manage meeting participation (join, leave, end)
- Detect nearby active meetings (within 100 meters)
- Provide real-time chatrooms inside active meetings
- Manage meetings automatically via a background scheduler
- GUI interface for all major operations
- Automatic handling of Redis server startup and shutdown for tests

Limitations:

- Physical proximity is simulated via manual coordinates (no GPS).
- The app resets all stored data (database and Redis) on every restart (designed for development and testing).
- No user authentication system (simple by design).

🗁 Project Structure

```
/    (root)
|-- docker-compose.yml
|-- Dockerfile
|-- launch.bat (one-click launcher)
|-- launch_app.py (Python launcher script)
|-- Proj1_Redis.pdf (Project specification)
|-- README.md (this file)
|-- src/
|   ├── app.py (Flask API server)
|   ├── db.py (SQLAlchemy ORM models)
|   ├── logic.py (Business logic functions)
|   ├── scheduler.py (Meeting activation)
|   ├── redis_client.py (Auto-detect Redis host)
|   ├── backend_utils.py (Helper functions)
|   ├── launch_utils.py (Docker orchestration helpers)
|   └── gui/
|       ├── gui.py (CustomTkinter GUI)
|       ├── gui_utils.py (GUI helper functions)
|-- tests/
|   ├── test_script_easy.py (Basic functionality test)
|   ├── test_script_difficult.py (Advanced multi-user test)
```

How to Run

1 Quick Run (Recommended)

Docker Desktop must be running ! ! !

Just **double-click** `launch.bat` or run it from a terminal:

```
launch.bat
```

This will:

- Build and start Docker containers (`docker-compose up`)
- Launch the Flask backend server
- Open the GUI automatically
- Stop all Docker containers properly when the GUI closes

☑ **One click to launch everything and clean shutdown!**

---

2 Manual Steps (only if needed)

```
# Step 1: Build and start backend and Redis
docker-compose up --build

# Step 2: In a new terminal, launch GUI
python src/gui/gui.py
```

Backend API available at: `http://localhost:5000`

Running Test Scripts

Test scripts **automatically**:

- Start a Redis container if needed
- Run the test scenario
- Stop and remove the Redis container when finished

To run:

```
python tests/test_script_easy.py
python tests/test_script_difficult.py
```

☑ No manual Redis or Docker setup required for tests!

Technologies Used

- **Backend**: Python (Flask)
- **Database**: SQLite (SQLAlchemy ORM)
- **Real-Time Store**: Redis 7 (Dockerized)
- **Containerization**: Docker + Docker Compose
- **Frontend GUI**: CustomTkinter (CTk)
- **Testing**: Python test scripts with auto Redis handling

API Endpoints

Endpoint	Method	Description
`/health`	GET	Health check for backend
`/create_user`	POST	Create a new user
`/delete_user`	POST	Delete a user
`/create_meeting`	POST	Create a new meeting
`/delete_meeting`	POST	Delete a meeting
`/join`	POST	Join an active meeting
`/leave`	POST	Leave a meeting
`/end_meeting`	POST	End a meeting and timeout participants
`/post_message`	POST	Post a chat message
`/get_chat`	GET	Fetch all messages of a meeting
`/user_messages`	GET	Fetch all messages posted by a user
`/user_chat_in_meeting`	GET	Fetch user's messages in joined meeting
`/get_nearby`	GET	Get nearby active meetings
`/active_meetings`	GET	List all active meetings
`/force_scheduler`	POST	Manually run the scheduler (for dev/testing)

Important Notes

- Times are internally stored as **UTC**; GUI displays them in **Greek timezone (Europe/Athens)**.
- **Data resets** at every startup (database and Redis) for clean testing.
- Full real-time meeting management based on physical location simulation.
- Fully meets the requirements of `Proj1_Redis.pdf`.

Developers

Maria Schoinaki, BSc Student
Department of Informatics, Athens University of Economics and Business
p3210191@aueb.gr

Nikos Mitsakis, BSc Student
Department of Informatics, Athens University of Economics and Business
p3210122@aueb.gr

Big Data Management Systems Course @AUEB 2024 - 2025
