Laboration 2 - Modelling and API-implementation

Key dates:

1. 2025-12-06 23:59 - Deadline to choose a partner or work alone
2. 2025-12-17 23:59 - Deadline to submit video & miro board
3. 2025-12-19 23:59 - Deadline to give feedback

Assignment Structure

This assignment is structured such that you will model a database based on a real world case, create an API based on it, record a demo video, and finally give feedback on another group's video and code via a Discord thread.

Your code will be public, so you are expected to share it with others.

Timeline

1. Choose whether to work alone or with a partner before the deadline 2025-12-06 23:59
2. Choose a case to work on from the cases section below and accept the github classroom invitation (your repository will be public, so you can easily share a link to it)
3. Model the database based on your research and experimentation
4. Create an API with FastAPI based on your model (it can be somewhat limited)
5. Bonus: Use AI to generate a simple React frontend that consumes your API to make your demo more engaging
6. Create a demo video, around 30 minutes long, where you present your case, database, and API. Post the video in our Discord channel in a thread with your group name.
7. Give feedback to another group's video in their Discord thread, focusing on the modelling step and not the API-code (Should be a group with a different case).

The purpose of this assignment is to practice modeling something realistic, implementing what you've modeled in code, and building the confidence to share your code publicly and receive feedback — just like in real code reviews at companies.

Feedback will come not only from the teacher but also from other students, meaning you'll need to review someone else's code too. You'll decide for yourself which feedback you agree with and think carefully about what feedback to give others. You aren't required to reply to the feedback/hold a discussion, but why not? Disagreement is common in tech, even though some things obviously are incorrect or correct 😃

Your grades, however, will not be made public, and students should not attempt to grade each other.

Late Submissions

Late submission means you cannot receive a VG because it means you technically could look at other groups videos and just replicate it.

Grading Information

You cannot receive a VG (higher grade) if you submit after deadline 1. You are graded individually, but I still recommend grouping with someone you think might aim for a similar grade.

Grade G

- You must complete all steps except the bonus step.
- Demonstrate a solid understanding of relational databases (e.g., when discussing your ERD diagram)
- Show basic understanding of REST APIs and how to implement CRUD operations
- Implement at least 21 endpoints in your API using different HTTP methods and CRUD operations
- Demonstrate basic proficiency with Postman or a similar tool
- Provide feedback to your assigned group through their Discord thread, focusing on their ERD-diagram / the modelling step.

Grade VG

- Meets all criteria for grade G
- Demonstrates deeper understanding of the modeling process (e.g., by critically discussing design choices)
- Implements more advanced CRUD operations involving complex relationships (such as 1-to-many and many-to-many)
- Shows understanding of data validation with Pydantic
- Demonstrates deeper proficiency with API testing tools (Postman, FastAPI Swagger docs)
- Displays clear technical insight and precision with great confidence

Step 1 – Choosing a Partner or Working Alone

DO NOT JOIN SOMEONES GROUP WITHOUT ASKING.

You must decide whether to work in pairs (2 people) or alone.

Working in pairs is recommended since you'll get to practice Git collaboration, discuss challenges, and experience pair programming.

However, it's important to choose someone you can work well with so that you both actually learn something.

You can also try implementing parts individually (e.g. building the API), then synchronize and merge your work for the final product.

Example workflow:

1. Model the database together — but first, spend some time experimenting individually
2. Try building the API on your own — then spend a day or two refining it together
3. Before giving feedback to your assigned group, review their codebase and diagrams individually — then discuss them together

Classroom invite:

https://classroom.github.com/a/GioptpWk

Step 2 - Cases → Choose one

Once you've decided on your partner and agreed on a case, create a "thread" in the "laboration 2" channel on Discord by clicking on the channel → click on the "plus" → create "thread" → name the thread after your names + case, e.g "Hemnet - Martin G + Erik M"

The goal is to have some variety — not everyone choosing the same case. Take a look in the "laboration-2" channel make sure of it.

It's now time to model the database - you'll need to perform extensive research by testing out the product / application, to the best of your ability, and doing the best job you could possibly do in trying to understand how it works. This means looking around at the website. However, you are NOT allowed to ask a chatbot to create a complete solution - I want you guys to really use your own brain, as modelling a database based on a case is a SUPER COMMON interview strategy that tests our ability to reason in databases.

IMPORTANT:

You will be using miro.com when modelling your database - it’s an amazing tool for collaboration, where both of you can simultaneously work on the modelling / brainstorm. Include a link to your Miro board in the file miro.txt — do this early, not at the end

---

Case: Hemnet (Classifieds-type SaaS-product)

Description:

Hemnet is a platform for buying and selling homes, where individuals can advertise their properties in collaboration with real estate agents. Other users can then participate in bidding on these listings.

You have been tasked with modeling and implementing a simplified API that mirrors hemnet.se, as your manager wants to develop a competing product (since Hemnet has an almost monopolistic position in Sweden).

Do your own research!

As a first step, model relevant tables — you will likely end up with around 10 tables.

Example tables:

- house_listings
- listing_category (e.g. house, apartment)
- users (customers, realtors, administrators)
- realtor_agents (real estate agencies)
- favorites (a user can favorite a listing)
- images
- realtor_agent_reviews

More info: https://www.hemnet.se/kundservice/maklare

Case: Mentimeter (Digital Presentation / EdTech)

Mentimeter is a Swedish web app that lets you create interactive presentations where audiences can vote, respond to polls, and give real-time feedback — often used in classrooms, workshops, and meetings.

Example tables:

- users (presenters and participants)

- presentations
- slides
- questions
- responses
- sessions (live events with timestamps)
- participant_sessions (who joined which session)

More info: https://www.mentimeter.com

Case: Kahoot (EdTech / Game)

Kahoot is a Norwegian quiz-based learning platform where teachers, teams, and families can create and play quizzes live.

Participants join using a short code and answer on their phones, competing for points.

Example tables:

- users (teachers, hosts, players)
- quizzes
- sessions
- session_players

More info: https://www.kahoot.com

Case: Bokadirekt (Booking System)

Bokadirekt is a Swedish booking platform where customers can find and book appointments with salons, clinics, massage therapists, and other service providers — like “Booking.com for treatments.”

Example tables:

- users (customers and providers)
- businesses (salons, clinics, etc.)
- payments (optional – price, method, status)
- reviews (customer feedback)
- categories (wellness, beauty, health)

More info: https://www.bokadirekt.se

Case: Tradera (Auction Platform)

Tradera is a Swedish auction site for buying and selling second-hand items.

Users can list products for auction, bid on others, and complete purchases online.

Example tables:

- users
- listings
- bids
- reviews
- watchlist

More info: https://www.tradera.se

Case: Omniway / MyMoodle (Learning Management System)

Omniway, MyMoodle, and similar LMS platforms are used by schools and organizations for online education.

Users (students and teachers) can access courses, submit assignments, track progress, and communicate within the platform.

Example tables:

- users
- courses
- enrollments
- assignments

Step 3 – Implementing Your API

Implement your ERD diagram and expose it as an API using FastAPI + psycopg2 or psycopg3. Using an ORM (e.g sqlalchemy) is not allowed, we'll do that later in upcoming courses.

You are free to expand the file structure or adjust functions as needed.

Start from the provided GitHub repository (see the README and file descriptions).

Here are some expectations / reasonable size of the API:

- At least 5-10 GET-endpoints
- At least 5-10 POST-endpoints
- At least 5-10 PUT-endpoints
- At least 5 DELETE-endpoints
- At least 1-2 PATCH endpoints

A lot of them will look similar, so when you end up demo:ing, perhaps focus on the more interesting / unique ones.

Step 3.5 - BONUS (not required, only if you have time):

If you want to get a sneak peak of creating a fullstack application - try to AI-generate, e.g using cursor, claude code or codex, a frontend-application in react using react-vite. We will learn properly how this works in an upcoming course, but why not test the waters? It's a lot more fun when your API is used by a frontend that actually uses the data.

Step 4 – Video Presentation

Record a ~30-minute video (it's okay if it's slightly shorter or longer) where you present and demo your application. If you are working alone, I still expect at least 30 mins.

You should both appear on camera (either together or side by side).

Do not use PowerPoint — focus on the case, the database, and the API.

Use your Miro board as your visual base when presenting.

Suggested structure:

1. Introduction (≈5 min)

Talk about your case based on your research and experience modeling it.

Explain how the real application seems structured and what challenges you identified.

2. Database Modeling (5–10 min per person)

Both of you discuss your database model using the Miro board.

Speak freely — don't read from a script.

Show that you can discuss and reason about your design decisions.

3. Code & Functionality (≥7 min per person)

Both participants should explain different parts of the code and functionality of the API.

When you've finished the video

Post the video and your miro-board to the thread 2025-12-17, and make sure you give feedback on the 2025-12-19 and no later.

Tips:

- It's fine to edit the video slightly (e.g., small retakes).
- Present as if you're explaining your work to another developer.
- Explain the structure, reasoning, and technical decisions clearly.

Video Requirements:

- Your diagram and code must be clearly visible (minimum 720p)
- Both participants must be clearly audible
- Both must have sufficient speaking time
- Review the video before submitting
- Add the video link in videolink.txt, and make sure it is publicly accessible (no login or request needed)
- Do a test recording first! Make sure sound and video quality are good

Step 5 - Giving feedback

You are to give feedback to another group (either you combine both of your feedback, or you add feedback individually), that has not yet received feedback. You simply enter a message in the other groups thread saying "Group X will be giving feedback to you". If for some reason no group exists to give feedback to, you will choose an already taken group. If you want to, you could give feedback / discuss other groups as well, if you see they've already received their first wave of feedback. Here's how you should think about reviewing their project & code

- I will look at your feedback and see if I agree with it or not. So think about it carefully and do your best.
- Focus on giving feedback on the modelling step. Do you think their choices seem reasonable? If not, what could they have tried instead? Are there any trade-offs? Your feedback should 80% - 100% be about the modelling stage.
- You are free to comment on the API, but it's not the focus and you are not required to comment it

- It’s OK to give feedback on presentational things, such as “I would have appreciated XYZ when demo:ing”, but there should be very little focus on the technique of the presentation. It’s not required.
- Always give constructive feedback in a positive way - and if you really feel like its necessary to be critical, use the “sandwich” model, meaning you start by giving positive feedback, you put something critizing in the middle, and you finish off with something nice 😃 🥪. At the end of the day, we’re all just trying to learn and get better, so getting feedback means we care about someone elses success - but still, when giving feedback, be nice.
- Think of the other group as your colleagues - if you have any advice, life-hack, something that you know worked for you, share it with them! Help them get better. Perhaps you noticed them struggling with something, perhaps they mentioned something in the video you know a good answer to - help them!

Final words

Don’t overcomplicate this assignment. You’re supposed to model a database, just like in lab 1, but it’s larger and more ambitious. You then create an API based on it, to the best of your ability. You then record a video, and finally spend 1-2 hours giving feedback to someone else. I repeat: DO NOT OVERCOMPLICATE THIS IN YOUR HEAD.