CSCI 341

Database Management Systems

Fall 2025

Assignment 3

Due Date: Monday, 24 November 2025, at 23:59

This assignment may be completed in groups of up to three students (working individually is also allowed).

Project Description

In this project, you will design a database and implement a practical application of the subject. The scenario below comes from a real product description. You are free to enrich and revise the properties of the given project ideas. User interface of the projects will not be graded. The database model that you have created and the criteria you are asked to satisfy matter. The aim of this project is not to create a competitor application but to teach you how to design an application which utilizes databases.

Case Study: An Online Caregivers Platform

Platforms for caregiver services have become popular for finding trustful caregivers. Caregivers on several categories (e.g., babysitters, for elderly care) register to the platform, and families who seek a caregiver may search among the candidates and make an appointment via the platform. In this project, you are required to implement such a platform with the following features:

1. Caregivers register to the platform with the following information:
    a. Name
    b. Surname
    c. Type of caregiving (one of the categories: babysitter, caregiver for elderly, playmate for children)
    d. Gender
    e. Photo
    f. Mail address
    g. Phone number
    h. City
    i. Price requested per hour
    j. Biography and personal information: A short description of the caregiver such as education information, working experience
    k. Password
2. (Family) Members who seek a caregiver also register to the system with the following information:
    a. Name
    b. Surname
    c. Mail address
    d. Phone number
    e. Password

f. City

g. Address

h. Information about the person in need of nursing: A short description about the family member for whom the caregiver is needed (e.g., I have a 5-year old son who likes painting…)

i. House rules: Description about the house rules that the caregiver should follow (e.g., the caregiver should pay attention to hygiene…)

3. Family members can search caregivers based on the caregiving type and city.

4. Family members can post job advertisements to express their need for caregiving. Caregivers can then search through these announcements and apply as needed.

a. Since posting an advertisement is a more detailed process, family members explicitly state the criteria that should be met by the caregiver. For example, the announcement should include details such as the person's age who requires caregiving, preferred time intervals (e.g., 09:00-12:00, 12:00-15:00), and the frequency of caregiving services (e.g., weekly, daily, weekends only).

b. Members can view the profiles of applicants who applied in response to the advertisements they have posted.

5. Members can view profiles of the potential caregivers that satisfy their conditions. On the profile, they can send a message to the caregiver, or they can allow the caregiver to contact with them.

6. Members can make an appointment with the caregivers on their profiles.

a. For making an appointment, members state

i. Date of the appointment

ii. Hour of the appointment

iii. Total number of hours for caregiving service (e.g., 3 hours, 5 hours)

b. If a member creates an appointment with a caregiver, caregiver should confirm or decline the request.

c. If the appointment is confirmed, the contact information is shown to both members and caregivers.

d. Members can see their appointments on their profiles.

Part 1 (20 points)

Construct a physical database according to the given specifications (schema) below. To create the physical database, it is required to use a database management system. You may use one of the free DBMSs -- PostgreSQL or MySQL.

NOTE: Certain aspects of the case study may not map directly onto the provided schema. You are free to make reasonable assumptions and explain your design decisions in your report.

The schema is:

USER (user_id, email, given_name, surname, city, phone_number, profile_description, password)
CAREGIVER (caregiver_user_id, photo, gender, caregiving_type, hourly_rate)
MEMBER (member_user_id, house_rules, dependent_description)
ADDRESS (member_user_id, house_number, street, town)
JOB (job_id, member_user_id, required_caregiving_type, other_requirements, date_posted)
JOB_APPLICATION (caregiver_user_id, job_id, date_applied)
APPOINTMENT (appointment_id, caregiver_user_id, member_user_id, appointment_date, appointment_time, work_hours, status)

After constructing the database model, you should create tables and insert data instances in the tables (**at least 10 instances for each table**). Please note that the result set of the queries from **Part 2** should be non-empty. (This means that you must think of data instances you are going to insert).

Part 2 (40 points)

In this part, you are going to connect to your (PostgreSQL or MySQL) database and execute some queries and updates. You are asked to use Python3 and SQLAlchemy library to connect and interact with your database. Please refer to the SQLAlchemy documentation (https://docs.sqlalchemy.org/en/14/intro.html). You can use SQLAlchemy ORM or execute the raw SQL scripts using Textual SQL (https://docs.sqlalchemy.org/en/13/core/tutorial.html#using-textual-sql). You may choose any approach you prefer as long as it is easy to demonstrate. You can also use IDEs to help you organize your work. Some of the options are DataGrip and Pycharm with Database extensions.

SQL queries:

1. Create SQL Statements

You will create all the tables according to the schema described above.

You can assume that tables will be created before executing any other database operation. Do not forget to define primary keys and foreign keys while you are creating tables.

2. Insert SQL Statements

You will insert data into appropriate tables. Make sure that each query listed in items 5,6,7, and 8 (i.e., Simple Queries, Complex Queries, Query with a Derived Attribute, and View Operation) is not empty and produces at least one answer.

3. Update SQL Statement

3.1 Update the phone number of Arman Armanov to +77773414141.

3.2 Add $0.3 commission fee to the Caregivers' hourly rate if it's less than $10, or 10% if it's not.

4. Delete SQL Statement

4.1 Delete the jobs posted by Amina Aminova.

4.2 Delete all members who live on Kabanbay Batyr street.

5. Simple Queries

5.1 Select caregiver and member names for the accepted appointments.

5.2 List job ids that contain ‘soft-spoken’ in their other requirements.

5.3 List the work hours of all babysitter positions.

5.4 List the members who are looking for Elderly Care in Astana and have “No pets.” rule.

6. Complex Queries

6.1 Count the number of applicants for each job posted by a member (multiple joins with aggregation)

6.2 Total hours spent by care givers for all accepted appointments (multiple joins with aggregation)

6.3 Average pay of caregivers based on accepted appointments (join with aggregation)

6.4 Caregivers who earn above average based on accepted appointments (multiple join with aggregation and nested query)

7. Query with a Derived Attribute

Calculate the total cost to pay for a caregiver for all accepted appointments.

8. View Operation

View all job applications and the applicants.

Part 3 (40 points)

In this task, you will design a web application with essential CRUD (Create, Read, Update, Delete) operations to access and manage the database contents. This will extend your experience with database management by integrating web development and full-stack application concepts.

Requirements:

- **CRUD Functionality**: The application should support core database operations for each table, enabling the addition, retrieval, updating, and deletion of records.
- **Development Frameworks**: You may choose a framework like Django or Flask for your development.

- **Deployment**: Deploy the web application on a publicly accessible platform, such as PythonAnywhere (free for one web app, as we know) or Heroku (not free). For additional free hosting options, consider Amazon's educational-tier servers, accessible via your university email (@nu.edu.kz).

Video presentation and executive summary:

You need to prepare a video where you demonstrate your implementation, clearly separating Part 1, Part 2, and Part 3, in that order. In the video, you need to show how you created your tables, how you populated them, and show the code of your queries and their results. Please try to keep your videos as concise as possible, ideally within the 10-minute limit. Upload your video presentation to Google Drive and provide a shareable link in your submission.

Moreover, you need to submit an executive summary of your project - at most one page and it should include a brief description of what you managed to complete and what you could not do.

Submission:

Export your database in .sql format (you can use pg_dump to do this).

Submit a zip file on Moodle that will consist of the following files:

1. .sql file of your database.
2. .py file that will connect to your database and execute all the queries.
3. Executive summary.
4. A (text) file containing the link to your video presentation (do NOT submit the video itself).

Regulations:

- The assignment may be completed in groups of up to three students (working individually is also allowed).
- Copying from other groups is strictly prohibited.
- Your report must be well written, and all figures should be clear and easy to read.
- Some groups may be asked to present or defend their assignment live on Zoom or in class.

NOTE: To avoid many unnecessary questions, please read the requirements very carefully (multiple times) and implement everything based on your understanding of the requirements. This document is self-explanatory!