

Faculty of Informatics

Department of Information Systems

P170B114 Fundamentals of Information Systems Coursework Report

Project topic: GymMembership

Submission date:

**Lecturers:** Assoc. Prof. Drungilas Vaidotas

Jr. Assist. Prof. Jaruševičius Pranas

**Students:** Anna Levchenko

Vladyslav Levchenko

Rustam Habibov

[Nojus Butrimavičius](https://moodle.ktu.edu/user/view.php?id=22719&course=533)

KAUNAS, 2024

Contents

[Introduction 3](#_Toc81207181)

[Team composition 3](#_Toc81207182)

[1. Purpose of the Project 3](#_Toc81207183)

[System description 3](#_Toc81207184)

[Implementation details 3](#_Toc81207185)

[Distribution of work 3](#_Toc81207186)

[2. Requirements model 4](#_Toc81207187)

[Use case model 4](#_Toc81207188)

[Use case activity diagrams 4](#_Toc81207189)

[Domain entity class diagram 5](#_Toc81207190)

[Non-functional Requirements 5](#_Toc81207191)

[3. Requirements analysis model 6](#_Toc81207192)

[Robustness diagrams 6](#_Toc81207193)

[User interface model 6](#_Toc81207194)

[4. System design 7](#_Toc81207195)

[System architecture 7](#_Toc81207196)

[Logical database schema 7](#_Toc81207197)

[System implementation 7](#_Toc81207198)

[Conclusions 8](#_Toc81207199)

[Literature 8](#_Toc81207200)

## Introduction

The problem domain is a gym membership management system, aimed at efficiently handling the day-to-day operations of a gym. The system must address key challenges, such as membership subscriptions, class scheduling, workout progress tracking, and trainers management.

## Team composition

Our team has four people: Anna Levchenko, Vladyslav Levchenko, Rustam Habibov, [Nojus Butrimavičius](https://moodle.ktu.edu/user/view.php?id=22719&course=533). A legend for identification of each team member's work:

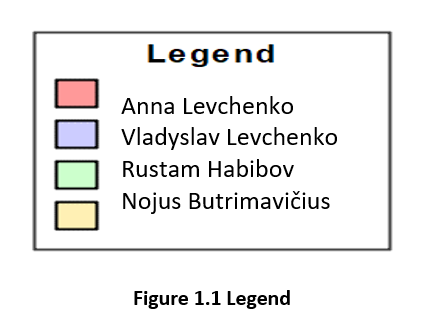


Figure 1.1 Legend

# Purpose of the Project

## System description

Description of the system: system aims, its subsystems, introduction to system user types and an outline actions they may perform. (The description of the system should take up at least 1 page.)

List objectives or goals of the project explaining what is expected when the System is developed and implemented.

Provide the details about the people that will be using the System as well as certain characteristics of their environment that can constrain the solution under development.

The Gym Membership Management System (GMMS) is designed to streamline the daily operations of a gym, providing members with a user-friendly interface to manage their subscriptions, book classes, track workouts, and access trainers - ensuring seamless coordination.

The system caters to several distinct **user groups**, each with unique roles and interactions within the system:

**- Gym Members:** These are the primary users of the system. Members sign up for gym services and interact with the system daily for booking classes, managing subscriptions, and tracking workout progress.

- Trainers: Manage their trainer profiles, certifications, and availability. They can schedule and cancel sessions. Trainers are assined to specific classes, activities, or one-on-one sessions.

- Admin: Use the system to ensure smooth operations within the gym. Handling class schedules, setting attendance limits, and managing membership plans. Monitoring trainer availability and assigning trainers to specific classes or activities.

**Objectives and Goals:**

The system must provide an intuitive and user-friendly interface to simplify the process of booking classes, tracking workout progress, and managing subscriptions. It should efficiently manage class bookings, ensure that class sizes are limited, and optimize the use of trainer availability, minimizing scheduling conflicts and maximizing gym resource utilization. Members should find it easy to navigate the system on both desktop and mobile devices, increasing overall engagement with gym services.

**Constraints:**

-We might encounter a learning curve when working with unfamiliar technologies or frameworks.

-Balancing the complexity of features with our team’s skills and time constraints for keeping the project manageable.

Our system consists of four subsystem:

1. Membership Subscription Subsystem:

The Membership Subscription Subsystem is designed to handle the comprehensive management of gym memberships, including the creation, selection, updating, and cancellation of membership plans. It serves both administrative staff and gym members, ensuring seamless interaction with the system.

- Admin can create new membership plans by defining key attributes such as the plan name, price, duration, and any special promotions. This allows for flexibility in offering various membership options to meet different needs.

- Members can select a new membership plan or switch from an existing one. The system displays available plans with details on pricing, duration, and benefits.

- Members have the ability to cancel their active memberships. The system updates the membership status to reflect the cancellation.

- Admin can modify existing membership plans by updating details such as pricing or duration. This ensures that the membership plans remain current and relevant.

- Both members and admin can check the current status of a membership, such as whether it is active, expired, or suspended. This helps in monitoring and managing membership statuses effectively.

- The system automatically renews memberships that are set to expire within the next 3 days. It ensures continuous membership without manual intervention.

2. Class Booking Subsystem

**3. Workout Plan Subsystem**

**4. Trainer Managment Subsystem**

Class Booking System:  
This allows members to book gym classes, view class schedules, and manage their bookings. It shows available classes, limits class size.

Workout Plan and Progress Tracking System

Enables users to create, view, and follow personalized workout plans. Tracks workout history, exercises, sets, reps, and weights lifted. Provides data analytics to show progress toward fitness goals.

Trainer and Staff Management System

Manages trainer profiles, certifications, and availability. Allows trainers to schedule and cancel sessions. Assigns trainers to specific classes, activities, or one-on-one sessions.

Table 1.1 Users’ specification table

|  |  |  |  |
| --- | --- | --- | --- |
|  | Admin | Gym Member | Trainer |
| Aims | - Manage membership plans, user data, schedules. | - Manage personal data, select a membership, book classes, do workout plan. | - Manage personal data and schedules and sessions with members. |
| Common Characteristics | - High-level access and control over system functions. | - Limited access focused on personal membership and class bookings. | - Access to scheduling and member management features. |
| Work Environment | - Office or administrative setting. | - Gym or home (via web/mobile app). | - Gym, office, or home (via web/mobile app). |
| Equipment Used | - Desktop or laptop computers. | - Smartphones, tablets, or computers. | - Smartphones, tablets, or computers. |
| Subject Matter Experience | - Extensive knowledge of gym operations and membership management. | - General understanding of gym memberships and services. | - Expertise in fitness training and session management. |
| IT Experience | - Intermediate to advanced knowledge of IT systems and tools. | - Basic to intermediate familiarity with web and mobile apps. | - Basic to intermediate familiarity with scheduling tools and apps. |
| Priority | - High priority on system stability, security, and administrative functions. | - High priority on ease of use, accessibility, and clear information on memberships and bookings. | - High priority on efficient scheduling, member interactions, and personal management. |

## Implementation details

The selected implementation technologies:

* Programming language, back-end frameworks, or libraries: **Django** is a high-level web framework for building web applications using Python.
* Database management system: **MySQL** is an open-source relational database management system.
* User interface, front-end frameworks, or libraries: **React** is a JavaScript library for building user interfaces. React allows us to create reusable UI components.

## Distribution of work

The task backlog and the distribution of responsibilities in a table format, using the legend to outline each team member contribution.

Table 1.2. Distribution of work

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Anna Levchenko | Vladyslav Levchenko | Rustam Habibov | [Nojus Butrimavičius](https://moodle.ktu.edu/user/view.php?id=22719&course=533) |
| Registration |  | + |  |  |
| Log in |  | + |  |  |
| Create New Membership Plan | + |  |  |  |
| Build form fields for entering Plan Name, Price, Duration, and Special Conditions. | + |  |  |  |
| Create UI for accessing "Membership Plans." | + |  |  |  |
| Implement selectection of their desired plan for members | + |  |  |  |
| Implement confirmation messages and update membership status. | + |  |  |  |
| Implement functionality to process membership cancellation. | + |  |  |  |
| Develop UI for selecting the membership to cancel. | + |  |  |  |
| Create UI for selecting the plan to update for admin. | + |  |  |  |
| Develop functionality to modify plan details (price, duration, etc.). | + |  |  |  |
| Develop UI for accessing "Membership Status." | + |  |  |  |
| Create functionality for entering or selecting membership IDs | ++ |  |  |  |
| Build functionality to fetch and display membership status and additional details. | + |  |  |  |
| Develop logic to identify memberships expiring within the next 3 days. | + |  |  |  |
| Implement functionality to renew memberships | + |  |  |  |
| Create system notifications to inform members of successful renewals | + |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Requirements model

## Use case model

Use case diagram and description. All use cases should correspond to the legend.

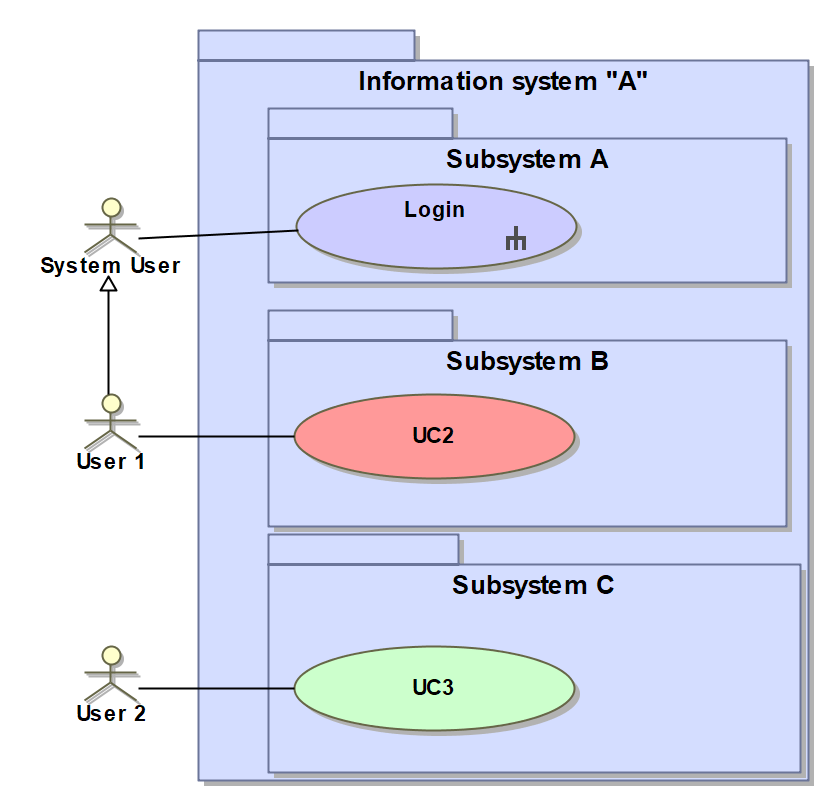


Figure 1.1 Use case diagram *(Example includes snippet only)*

## Use case activity diagrams

Each use case must have an activity diagram, and a specification provided. The activity diagram should denote the interaction between user and system.

Table 1.1. Use case "Login" specification

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | | **Login** |
| **Goal.** Log in to the system | | |
| **Description.** User enters credentials, and is logged into system | | |
| **Precondition** | | **User is not logged in** |
| **Actors** | | User |
| **Relations** | **Include** | **-** |
| **Extend** | **-** |
| **Generalization** | **-** |
| **Postcondition** | | **User is logged in** |

**SUBSYSTEM MEMBERSHIP SUBSCRIPTION MANAGEMENT:**

**Use Case:** **CREATE NEW MEMBERSHIP PLAN**

**Actor:** Admin

**Description:** The admin creates a new membership plan in the system by defining its features, such as price, duration, and any special conditions or promotions.

**Precondition:** The admin is authenticated and has the necessary permissions to manage membership plans.

Steps:

The admin logs into the system.

The admin navigates to the "Membership Plan Management" section.

The system presents an option to "Create New Plan."

The admin enters the following details for the new plan: Plan Name (e.g., Gold Plan, Summer Promotion), Price (e.g., $50/month or $500/year), Duration (e.g., 1 month, 3 months, 1 year).

The admin confirms the creation of the plan.

The system saves the new plan in the database and makes it available for members to select once it is activated.

A confirmation message is displayed, and the new plan is listed under the available membership plans.

**Postcondition:**

The new membership plan is successfully created and available in the system for future subscriptions. Members can view and select this plan.

**Use Case:** **SELECT MEMBERSHIP**

**Actor:** Gym Member

**Description:** A new or existing member selects a membership plan (e.g., monthly, yearly, or special promotion) to subscribe to or switch from an existing plan.

**Precondition:** The actor is logged as a member.

**Steps:**

The actor logs.

The actor navigates to the "Membership Plans" section.

The system displays available membership options, including plan details such as pricing, duration, benefits, and any current promotions.

The actor selects a membership plan.

A confirmation message is displayed with the membership details (start date, expiration date, etc.).

**Postcondition:**

The actor successfully selects or switches to a membership plan, and the membership status is updated

Use Case: CANCEL MEMBERSHIP

Actor: Gym Member

Description: A member cancels an active membership.

Precondition: The actor has permission to cancel the membership.

Steps:

The actor selects the membership to cancel.

The actor confirms the cancellation request.

The system sets the membership status to "Cancelled."

Postcondition: The membership is canceled, and the member no longer has access to gym services.

in the system.

Use Case: UPDATE MEMBERSHIP INFORMATION

Actor: Admin

Description: Admin updates information such as price or duration.

Precondition: The actor is authenticated.

Steps:

The actor selects the membership they want to update.

The actor updates the relevant field.

The system saves the updated information.

Postcondition: The updated information is successfully stored in the system.

Use Case: GET MEMBERSHIP STATUS

Actor: Gym Member/Gym staff

Description: The member or gym staff checks the current status of a membership to see if it is active, expired, or suspended.

Precondition: The actor is authenticated and has permission to view membership details.

Steps:

The actor logs in to the system (either as a member or gym staff).

The actor navigates to the "Membership Status" section.

The actor enters the membership ID (or selects the membership from a list if the actor is the member).

The system retrieves the membership details and checks the current status (e.g., Active, Expired, Suspended).

The system displays the current membership status along with additional information such as the expiration date and suspension reason (if applicable).

Postcondition: The actor successfully views the current status of the membership.

**Use Case:** **AUTO-RENEW EXPIRING MEMBERSHIPS**

**Actor:** System (automated process)

**Description:** The system automatically renews memberships expiring within the next 7 days and sends notifications to the members. If payment information is available, it processes the renewal fee.

**Precondition:** The system contains valid payment information for auto-renewal, and memberships are configured for auto-renewal.

**Steps:**

The system identifies all memberships expiring in the next 3 days.

After Extend the membership expiration date by the appropriate period (monthly/yearly)

**Postcondition:** Expiring memberships are successfully renewed, and notifications are sent to members.

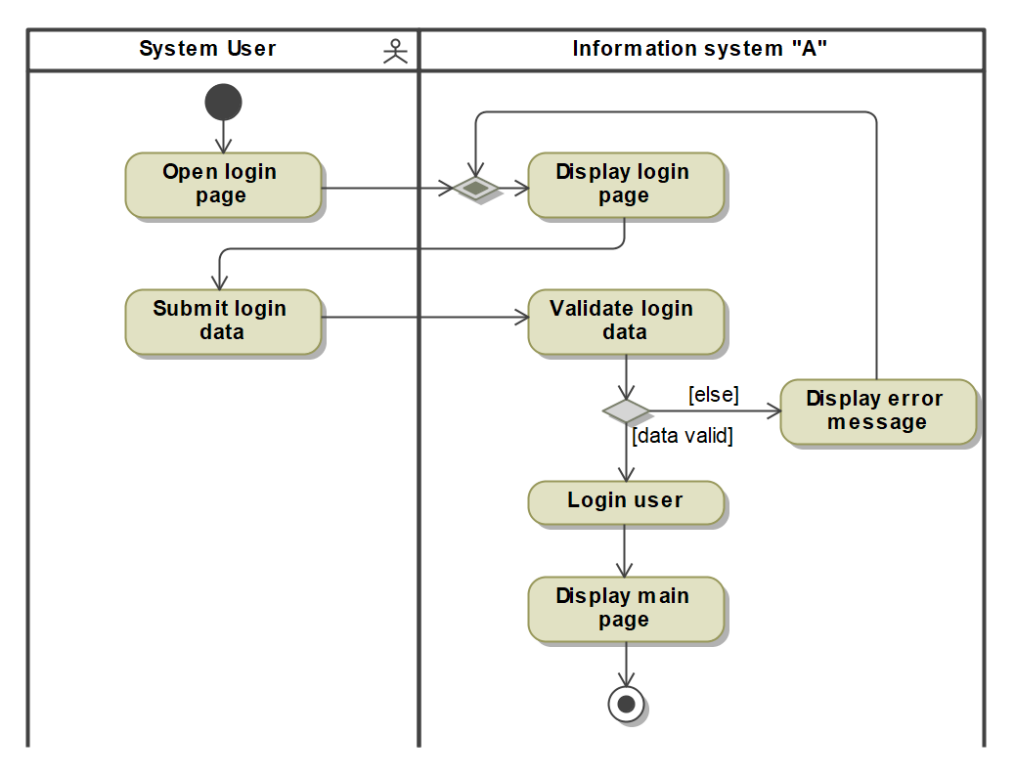


Figure 1.2 UC "Login" activity diagram

## Domain entity class diagram

Domain entity class diagram with a short description, classes must have a stereotype *«Entity»*. All elements in the chart must match the color legend.

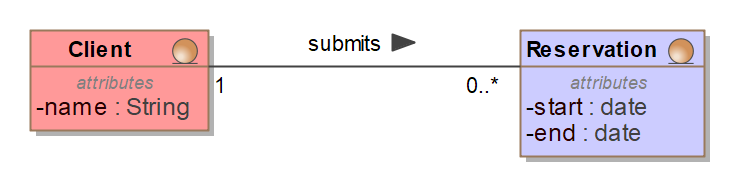


Figure 1.3 Domain entity class diagram *(Example includes snippet only)*

## Non-functional Requirements

In this section, list any non-functional requirements you find to be relevant. It is enough to list their formulations. However, make sure that the formulations are specific and unambiguous, – if this is not possible, include separate fit criterion(-s) next to a specific requirement. It is also advised to include the requirement type.

Non-functional requirements can be presented in plain text. Encoding each requirement is highly recommended, for example:

1. Type. Requirement1 formulation.
2. Type. Tequirement2 formulation.
3. …

# Requirements analysis model

## Robustness diagrams

Robustness diagrams are used during requirements analysis to determine what classes are required for implementation of a specific use case. A robustness diagram must be provided for each use case.

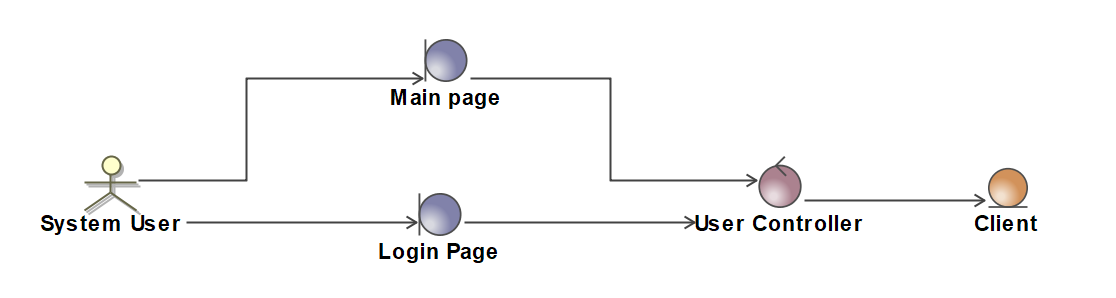


Figure 2.1 UC “Login” analysis diagram 1*(Example includes snippet only)*

## User interface model

Navigation plan presented in a class diagram with stereotype *«boundary»*.

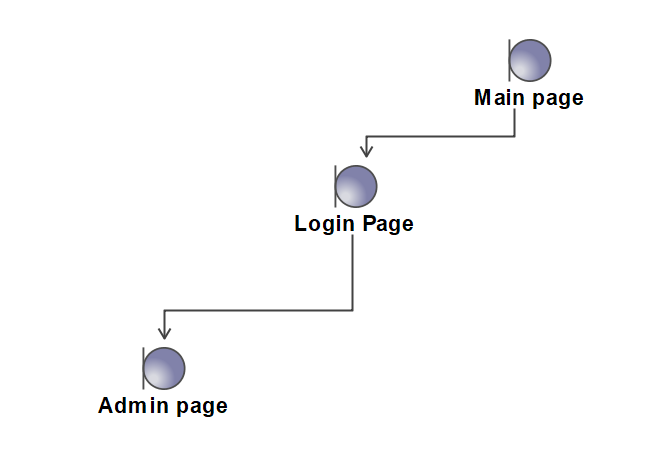


Figure 2.2 System navigation plan *(Example contains snippet only)*

# System design

## System architecture

Package diagram and description of the architecture of the system.

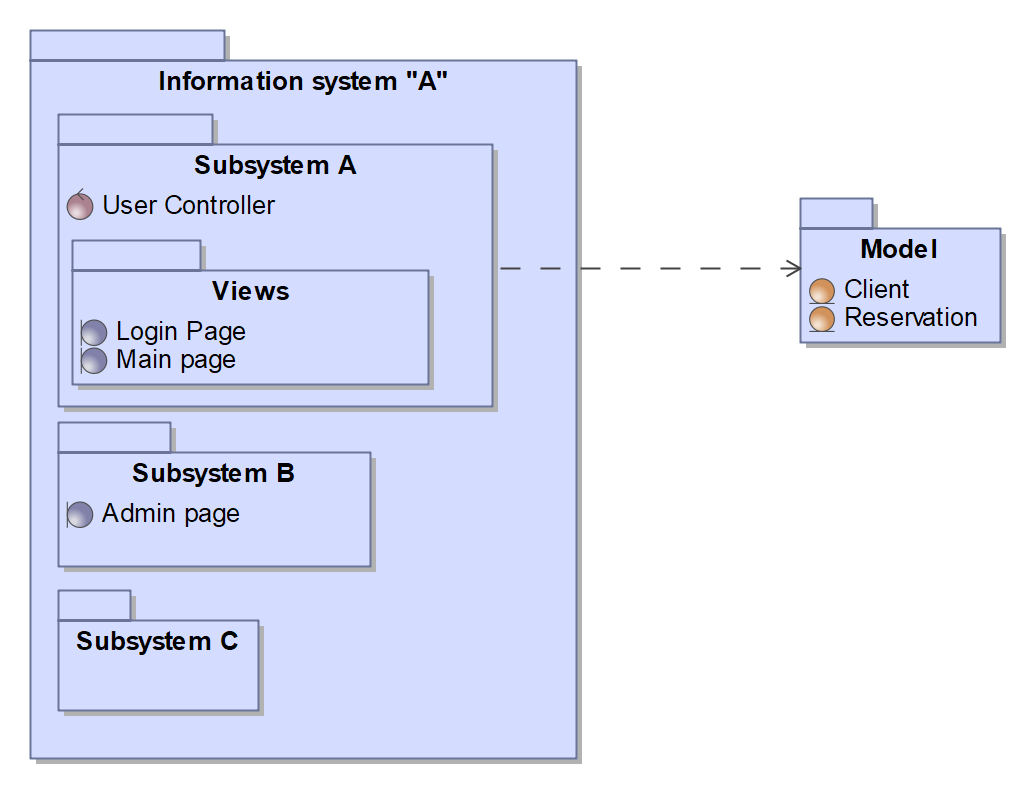


Figure 3.1 System architecture 1*(Example contains snippet only)*

## Logical database schema

Database model and its description (for each table). The diagram can be produced by transforming domain model to database schema. Stereotypes *«table», «PK», «FK» are used.* All tables must match the color legend specified in the description of the introduction.

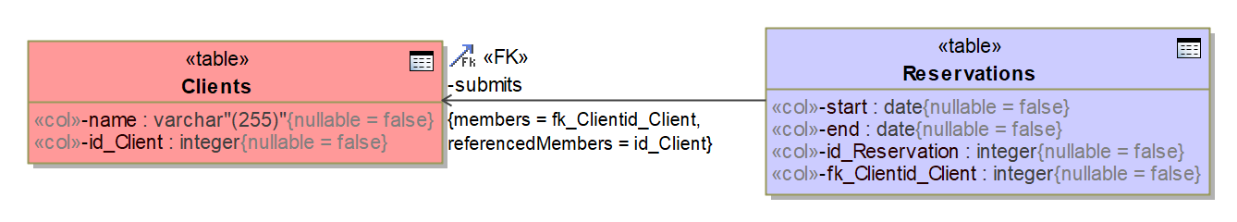


Figure 3.2 Logical database schema *(Example contains snippet only)*

## System implementation

Images and descriptions of all completed user interface pages, forms and reports. Screenshots should be provided with the test data.

# Conclusions

Present final conclusions covering the entire project. Draw conclusions with emphasis on qualitative (or at least quantitative) criteria. A fact is not a conclusion if it is not justified.

Examples of conclusions include:

1. Selected … domain because ...
2. Selected... (technology, composition of tasks, etc.), because ...
3. The system is designed for ... consumer types because ...
4. The system will have to ensure ... functions that ...
5. In the future, it would be useful to improve the system, as ...

# Literature

Provide the literature references used during the work (if necessary).