**1. Requirements Analysis and Conceptual Modelling - The business is Fitness First Gym.**

Q1. A set of requirements in natural language (English). This is a short text explaining what your system needs to do. The text needs to be concise and rigorous.

1. The gym system must manage member registration and payment information.

2. The gym system must provide access control to the gym premises.

3. The gym system must track member attendance.

4. The gym system must generate reports on gym usage and revenue.

5. The gym system must allow for the scheduling of classes and personal training sessions.

6. The gym system must manage equipment inventory and maintenance.

7. The gym system must provide a user-friendly interface for both gym staff and members.

8. The gym system must ensure data privacy and security.

9. The gym system should allow for online member registration and payment.

10. The gym system should provide mobile access control to the gym premises.

11. The gym system should send notifications to members about their attendance and upcoming classes or personal training sessions.

12. The gym system should integrate with fitness trackers or other health devices or track members progress.

13. The gym system could offer online classes and personal training session reservations.

14. The gym system could integrate with social media platforms to promote gym events and specials.

15. The gym system could provide personalised workout plans for members.

16. The gym system could offer virtual classes or personal training sessions.

17. The gym system won’t provide medical advice or diagnosis.

18. The gym system won’t guarantee member results or outcomes.

19. The gym system won’t allow access to unauthorised individuals.

Q2. An essential class diagram in UML. The essential class diagram identifies the entities (with one class per entity) in the domain of interest, as well as the relationships amount them (represented as associations with their multiplicities) but leaving out the operations and the attributes. This constitutes a high-level conceptual model of your system, representing the domain of interest of said system.

Diagram

Description automatically generated

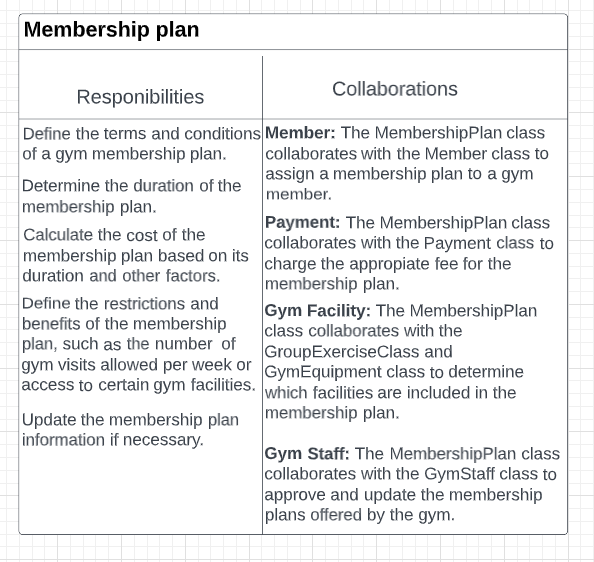
**2. Responsibilities with CRC Cards**

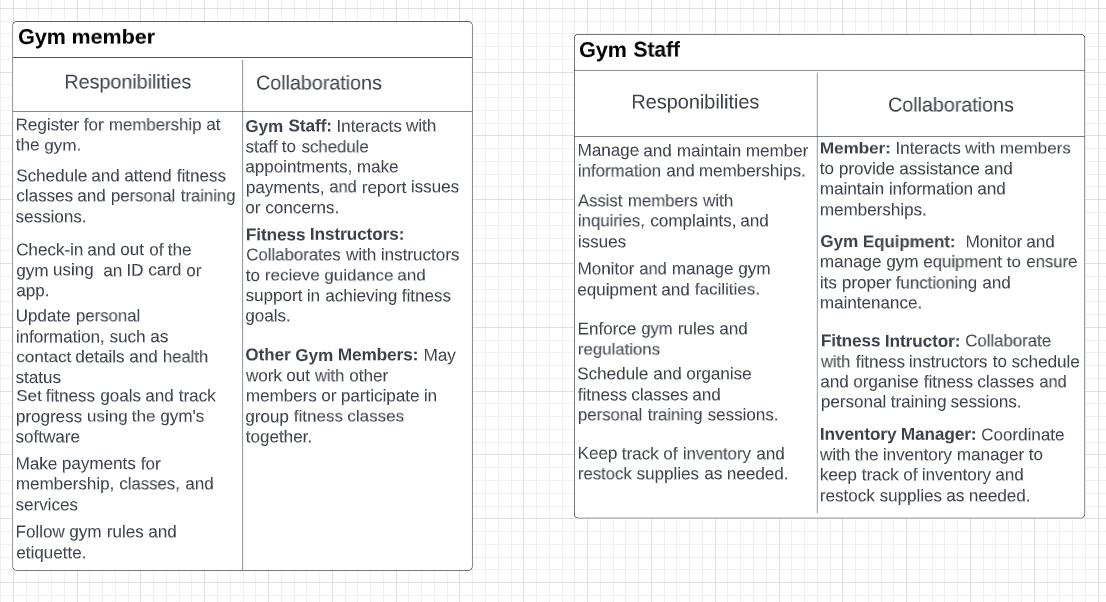
Q1. A use case diagram in UML. This should be derived straight from the functional part of the requirements you wrote in Part 1.

Diagram

Description automatically generated

Q2. CRC Cards for all classes (entities) in the essential class diagram you wrote in Part 1. The cards are produced by “crossing” the use case diagram with the essential class diagram.





Table

Description automatically generated

Table

Description automatically generated

**3. Operation Specification**

Q1. The list of operations to be implemented in the system, at a finer grain than the use cases. Each operation will be documented by a few words that explain what the operation does. You should use names that are as explanatory as possible for the operations.

1. AddMember(): Add a new member to the gym system.

2. RemoveMember(): Remove a member from the gym system.

3. UpdateMemberInfo(): Update the information of a member.

4. AddMembership(): Add a membership plan to a member (e.g. name, address, phone number, bank details).

5. RemoveMembership(): Remove a membership plan from a member’s account.

6. RenewMembership(): Renew a membership plan for a member.

7. CheckMembershipStatus(): Check the status of a member’s membership plan.

8. CheckIn(): Records that a member has checked in at the gym.

9. CheckOut(): Records that a member has checked out of the gym.

10. RecordPayment(): Records a payment made by a member (e.g. for a membership plan, personal training session).

11. ScheduleClass(): Schedule a group fitness class.

12. CancelClass(): Cancel a scheduled group fitness class.

13. RegisterForClass(): Register a member for a group fitness class.

14. CancelClassRegistration(): Cancel a member’s registration for a group fitness class.

15. SchedulePersonalTraining(): Schedule a personal training session for a member.

16. CancelPersonalTraining(): Cancel a scheduled personal training session for a member.

17. RecordPersonalTraining(): Record that a personal training session has taken place and the progress made.

18. ViewMemberActivityHistory(): view a member’s activity history (e.g. check-in and check-out times, class attendance, personal training sessions).

19. ViewClassSchedule(): View the schedule of group fitness classes.

20. ViewPersonalTrainingSchedule(): View the schedule of personal training sessions.

21. ViewMembershipPlan(): View the available membership plans and their details.

22. ViewMemberDetails(): View the details of a specific member’s account.

Q2. The detailed specification for at least three, non-trivial operations. Each specification can be given in the form of a decision table, an activity diagram or in a controlled English / pseudo-code.

This is pseudo-code for membership plan.

**add** registerNewMember(personalDetails, membershipPlanDetails) {

**get** uniqueMembershipID = generateUniqueMembershipID();

**set**("members." + uniqueMembershipID + ".personalDetails", personalDetails);

**set**("members." + uniqueMembershipID + ".membershipPlanDetails", membershipPlanDetails);

**get** sendConfirmationEmail(uniqueMembershipID);

**get** uniqueMembershipID;

This is pseudo-code for group exercise class.

**add** scheduleFitnessClass(classDetails) {

**if** (fitnessClass is full): {

**set** reservedSpots = reserveSpots(classDetails);

**set** instructor = assignInstructor(classDetails);

**set**("classes." + classDetails.date + "." + classDetails.time, {

"classType": classDetails.classType,

"instructor": instructor,

"reservedSpots": reservedSpots

});

**get** sendConfirmationMessage(reservedSpots, instructor);

} **else:** {

**get** sendClassFullNotification();

}

}

This is pseudo-code for gym equipment.

**add** checkEquipmentAvailability(equipmentType, dateTime) {

**if** (theGym is open) {

**set** availableEquipment = getAvailableEquipment(equipmentType, dateTime);

**if** (availableEquipment.length > 0) {

**get** availableEquipment;

} **else:** {

**get** print**(**"No equipment available");

}

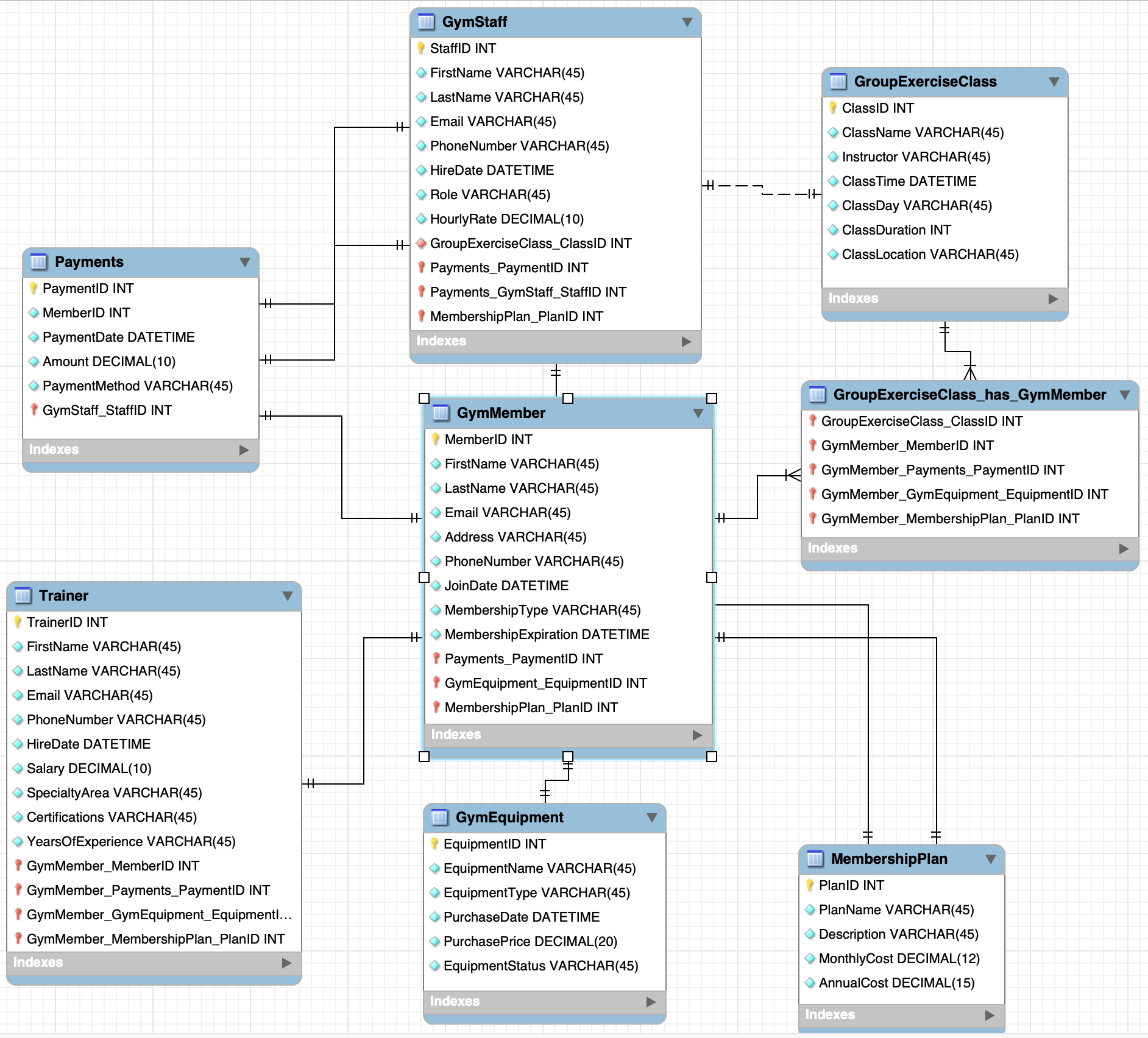
} **else:** {

**get** print("Gym is closed at this time");

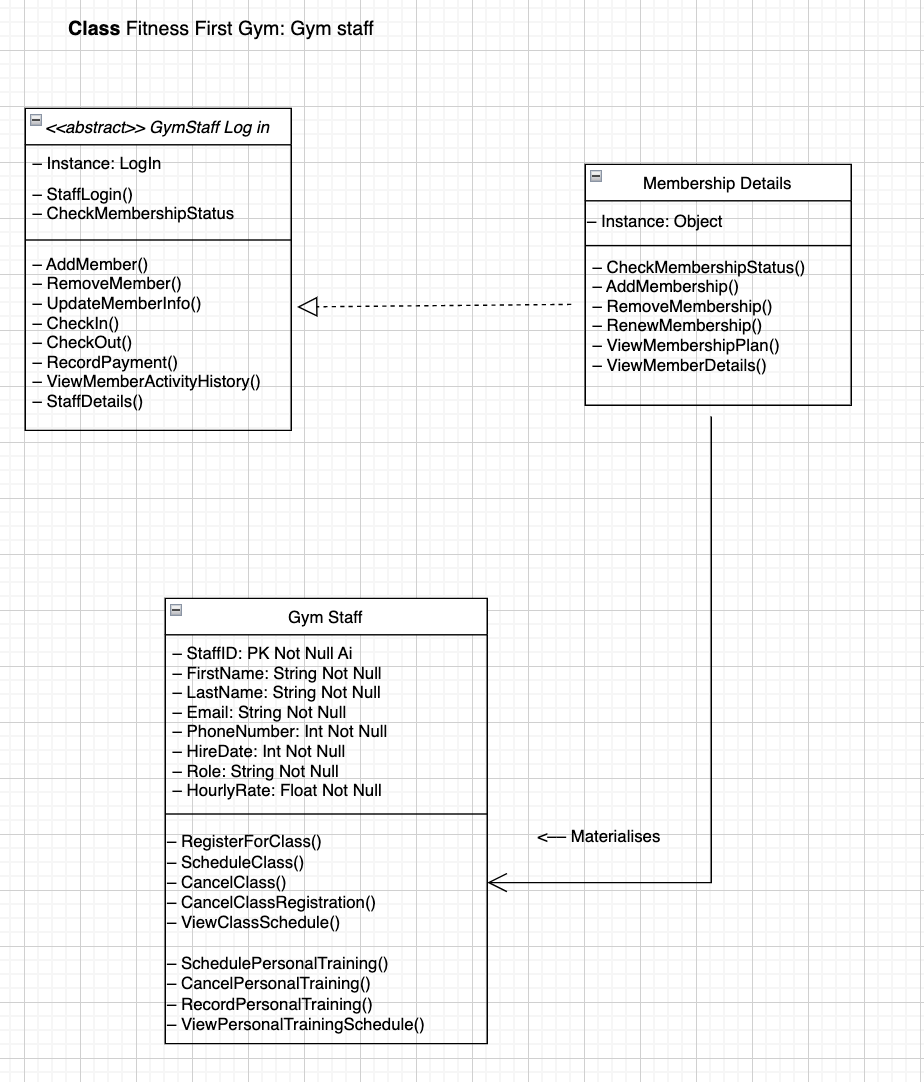
}}

**4. Persistence Design**

Q1. The relational schema for the persistent data that your system needs to store. This will be a reflection mostly of the essential class diagram you produced in Part 1, obviously with the representation of all properties/attributes of the class (which are not represented yet in the essential class diagram), whose information is found in the requirements set of Part 1.

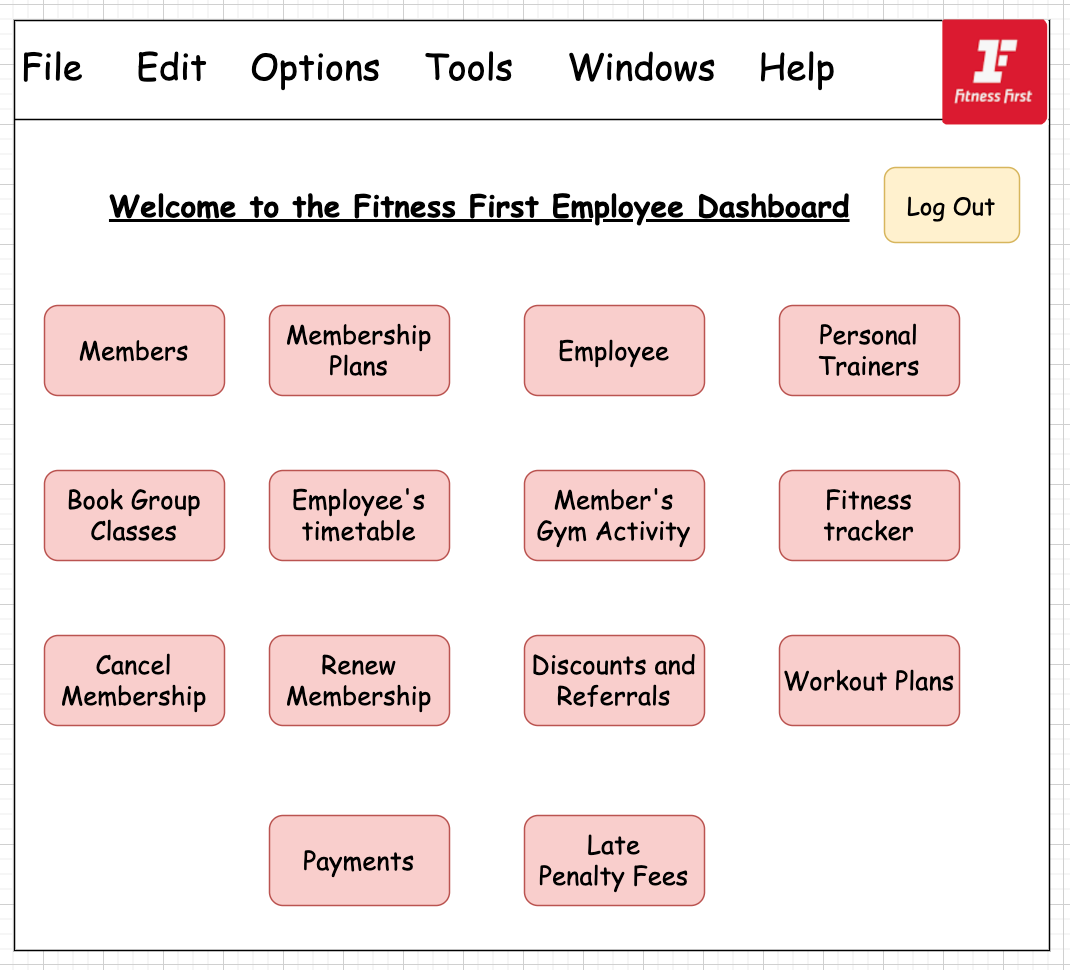


Q2. One broker class associated to an entity class of your choice among those in the essential class diagram. This must be represented in UML, and must have individual operations, which will have to reflect the responsibilities of the entity class, among those identified in Parts 2 and 3, that operate on persistent data



**5. User Interface Design**

Q1. An essential UI prototype for one major UI element of your system. This would be normally a paper-based prototype, with sticky notes moved around until UI elements are placed, without details, in the design of the UI; in your case, you will include a figure (produced with tools of your choice, including hand-drawing) in the report?



Q2. A high-level UI flow diagram related to the part of the UI necessary to one of the operations you have specified in Part 3.

Diagram

Description automatically generated