An Illustration of Systems Analysis Deliverable

Pro Health Club

Table of Contents

MS-1 Systems Analysis —Requirements Specification……………………………2

Introduction …………………………………………………………………2

Problem Description …………………………………….……………….….3

Business Activities of the Pro Health Club …………………………..4

System Requirements Specification ………………………………………...5

Function Model ………………………………….………...…………5

Use Case Model …………………………………….…………….….6

Data Model …………………………………….……………………11

Data Objects in the Pro Health Club ………………………………..11

Domain Class Diagram …………………………………….……….12

Database Query Requirements …………………………………….………13

Validation of Function and Data Models …………………………………..13

Non-Functional Requirements of the Firm …………………………..……14

Usability …………………………………….……………………....14

Performance …………………………………….…………………..15

Scaling …………………………………….………………………...15

Security …………………………………….……………………….15

MS-2 Database Design Deliverable........................................................................16

Introduction ..................................................................................................16 Logical Data Model ......................................................................................16

Initial RDM for the Real Estate Firm .................................................16

Definition of relations.........................................................................17 Normalization of Relations ................................................................24

Final Relational Data Model for the Real Estate Firm........................27

Validation of the Relational Data Model.................................................................28

MS-3 Real Estate Firm System UI Design (Deliverable #3) ..................................32

Introduction ............................................................................................................32

Master Layout (Template) for the webpages .........................................................33

Menu bar Design.....................................................................................................35

Webpage Design for each Function (use case) .......................................................37

MS-4 System Component Design for the Real Estate Firm....................................41 Introduction ............................................................................................................41 Use Case Realization ..............................................................................................42 Design Class Diagram.............................................................................................45 System Component--Internal Structure...................................................................46

Views.............................................................................................................46 Controllers.....................................................................................................46 Models ..........................................................................................................47

Deployment Model .................................................................................................47

MS-1 Systems Analysis —Requirements Specification

Introduction

In this document, we will analyze the activities of the Pro Health Club. This analysis allows will list and explain the functional and non-functional requirements needed to operate the Pro Health Club.

The outputs of the system analysis are described below:

1. Function models that outline the club’s major activities.

2. Data models that outline the type of data needed for daily operations.

3. Non-functional requirements which include: usability, performance, scaling, and security requirements.

All functional requirements will be depicted using case models, a class diagram, and matrix. Non-functional requirements will be described.

Problem Description

Pro Health Club is in the works of promoting health and fitness. It offers a wide variety of classes, instructors, and equipment for members.

Pro Health Club is situated in Lubbock, Texas. The organization is made up of classes. Attributes of the classes include ClassID (identifier), ClassName, and ClassFee.

The classes are operated by one or more managers. Managers are in charge of hiring instructors, creating class schedules, and monitoring attendance. Attributes of the manager include ManagerID (identifier) and contact information.

For every class, there is at least one instructor. An instructor can also be a personal trainer. Members who enroll in personal classes must pay an additional fee.

Members can enroll in general classes that are covered by their membership. Senior members and students have a discounted fee. This discount can be found under the attribute MembershipStatus. A member can also leave a review which is stored in the system.

Payments made by the member are stored in a card verification system. This system’s attributes include CardName (identifier), CardNumber, ExpirationDate, and Code.

As of now, Pro Health Club is without a working digital system. With one, it plans to increase security, efficiency, and maintenance.

Business Activities of the Pro Health Club

In the form of Use Case Models, the club can be described by its major and associated activities:

1.      Maintain Members

a.      Enroll members

b.      Receive feedback

c.      Register for general classes

d.      Register for personal classes

e. Review feedback

2.      Maintain Classes

a.      Process payment

b.      Maintain class schedule

c.      Offer classes

e.      Monitor class participation

3.      Maintain Instructors

a.      Hire instructor

b.      Assign instructor to classes

c.      Monitor instructor

d.      Assign trainer to personal class

System Requirements Specification

For Pro Health Club, the system has a need for functional and non-functional requirements to operate as planned. These functions are further explained by our case models and class diagram.

Function Model

Our case models depict the functional activities our system must perform. These models are shown in figures 1-6.

Use Case Model

Our collection of use case models describes the club’s order of activities. Pro Health Club has four interactive actors that participate in the club’s day-to-day functions. These actors include our workers (managers and instructors), our clients (members), and our card verification system. To start, Figure 1 shows the club at its broadest. All four actors are included as without them, the club would fail to function as intended. In Figure 2, our use case diagram breaks down our operations into three distinct case models: Maintain Members, Maintain Classes, and Maintain Instructors. These cases are found at level 0. Figures 3-6 are at level 1. The case models in these figures expand the activities listed in Figure 2, starting with Maintain Classes depicted in Figure 3. The activities of Maintain Instructors are depicted in Figure 4 and Maintain Membership is shown in Figure 5.

Diagram

Description automatically generated

Figure 1. Context Use Case Diagram for Pro Health Club

Diagram

Description automatically generated

Figure 2. Level 0 UCD of Club

Diagram

Description automatically generated

Figure 3. Level 1 UCD - Maintain Classes

Diagram, schematic

Description automatically generated

Figure 4. Level 1 UCD - Maintain Instructors

Diagram

Description automatically generated

Figure 5. Level 1 UCD - Maintain Membership

Data Model

Our data model is represented by our Domain Class Diagram (Figure 6). It shows our data objects, or entities, in our system that store information.

Data Objects in the Pro Health Club

For an effective club model, our entities need to store data and manage relationships between objects. Our objects include:

1.      Club

2.      Manager

3.      Class

4.      Instructor

5.      Member

6.      Payment Method

7.      Review

Each data object has attributes that store user inputs. These include a member’s payment information or a manager’s phone number. All attributes are needed to help organize and secure club data. Additional objects, like Class Schedule, are associated classes. They carry unique attributes that are affected by relating data objects, such as manager and class.

Relationships between objects are shown by a line connecting one object to another. These lines are labeled by an action that describes the relationship. For instance, the ‘train’ relationship connects a personal trainer to a member.

Domain Class Diagram

Figure 6 presents a visual representation of our data model. This model includes our data objects, their relationships, and their attributes.

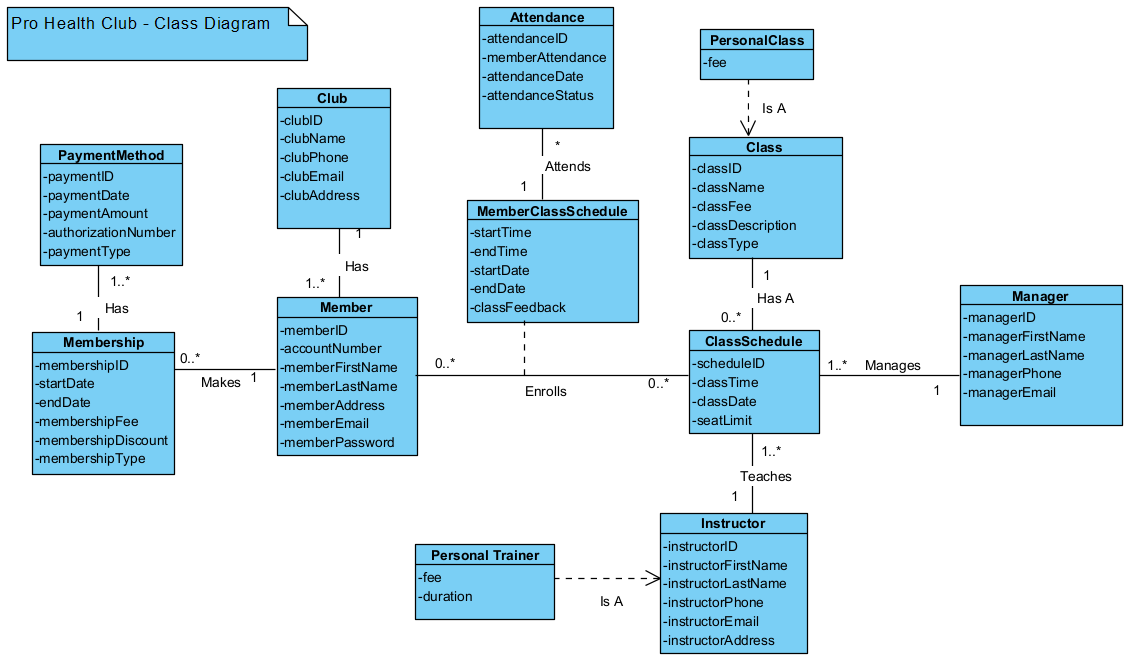


Figure 6. Class Diagram of the Club

Database Query Requirements

Our system stores the data found in the domain class diagram in a database. This information is needed to operate and manage the club effectively.

The queries needed include:

1.      List all general class schedules

2.      List all managers

3.      List all class schedules for a given period

4.      List all instructors and their classes

5.      List all personal trainers and their schedules

6.      List all personal class schedules

7.      List all members their class attendance

8.      List membership enrollment

9.      List member feedback and class ratings

10.   Compute all membership fees

11.   Compute all personal class fees

12. Display current revenue

13. Display the previous year’s revenue

Validation of Function and Data Models

The class diagrams and use case models must work with each other to ensure consistency and usability. The Matrix found in Table 1 shows the use of each data object in relation to the club’s activities found in Figures 1-5. Each data object, referenced in Figure 6, will have an associated case to prove its use to the system and vice versa.

Table 1. Data Object function Matrix of the Club

Chart, table, scatter chart

Description automatically generated

Non-Functional Requirements of the Pro Health Club

Non-functional requirements analyze the operations of the system outside of its case models and class diagram. For now, we will discuss usability, performance, scaling, and security.

Usability

The software system put in place will be simple enough for any user to access and navigate. All members and employees of the Pro Health Club will find the system efficient to use.

Performance

The system will be able to handle many users at once and have a desired response time. All users will be able to support up to 100 transactions per second without notable lag or system failure. Performance will not be an issue encountered by management and members.

Scaling

The software will be designed to hold all data involving instructors, classes, members, and managers. To do so, the system will have the capacity to store:

1. One million classes
2. Two million members
3. Two thousand clubs
4. Ten thousand instructors

Modifications can be made to account for future growth.

Security

The system will be accessed via website. This will require a secured login to ensure the safety of club data. Confidential information regarding all members, instructors, and managers with be stored within the system. This includes IDs, class schedules, and payment information. All users will require a username and password before accessing any information. Additional security measures include double verification, which will require the user’s phone number or private email.

To ensure privacy, the user will be limited in their access to data unrelated to themselves or their activities. There will be different levels of permissions to ensure this. For example, a manager can reach member and instructor information, but instructors and members can only access their own data.

MS-2 Database Design Deliverable

Introduction

In this section, we document the steps of the database component design for the Pro-Health Club.

The database component design involves:

1. Logical Data Model

2. Physical Data Model

3. A My SQL Implementation of the physical data model

* Only the logical database design will be included in this deliverable.

Logical Data Model

Our relational data model (RDM) is Pro-Health Club’s logical data model. The class diagram shown in Figure 6 was used to create an equivalent RDM. This translation between the diagram and the relational data model was achieved through the use of conversion rules. By doing so, we can ensure that the relations are in third normal form.

Initial RDM for the Real Estate Firm

Our initial RDM can be seen below in figure 7. Each relation has attributes, primary keys, and zero or more foreign keys. We also defined each relation and its attributes below.

Text

Description automatically generated

Definition of relations

Here, attributes are specified according to their relation.

1. Club(**ClubID**, ClubName, ClubPhone, ClubEmail, ClubAddress)

The relation is used to store details of the Club. The following table Defines the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| ClubID | Alphanumeric (5 characters long) | Stores Primary key for clubs’ relation. |
| ClassName | Alphanumeric (Up to 30 characters long) | Stores Club name. |
| ClubPhone | Alphanumeric (11 characters long) | Stores Club Phone number. |
| ClubEmail | Alphanumeric (Up to 30 characters long) | Stores Club Email. |
| ClubAddress | Alphanumeric (Up to 50 characters long) | Stores Club Address. |

1. Member(**MemberID**, MemberLastName, MemberAddress, MemberEmail, MemberPassword, AccountNumber, *ClubID*)

The relation Member holds data about the members of the club. The following table defines the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| MemberID | Alphanumeric (5 characters long) | Stores Primary key for the member ID. |
| MemberLastName | Alphanumeric (Up to 30 characters long) | Stores the last name of the member. |
| MemberFirstName | Alphanumeric (Up to 30 characters long) | Stores the first name of the member. |
| MemberAddress | Alphanumeric (Up to 50 characters long) | Stores the member's address. |
| MemberPassword | Alphanumeric (Up to 30 characters long) | Stores the member's Password |
| AccountNumber | Alphanumeric (5 characters long) | Stores the Account Number |
| ClubID | Alphanumeric (5 characters long) | Stores Primary key for clubs’ relation as a foreign key |

1. Membership(**MembershipID**, StartDate, EndDate, MembershipFees, MembershipDiscount, MembershipType, *MemberID*)

This relation holds the data about the Membership’s relation. The following table describes the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| MembershipID | Alphanumeric (5 characters long) | Stores Primary key for Membership ID. |
| StartDate | Date (MM/DD/YYYY) | Stores Start Date of the membership. |
| EndDate | Date (MM/DD/YYYY) | Stores End Date of the membership. |
| MembershipFees | Decimal (money) | Stores the price of the Membership. |
| MembershipDiscount | Decimal (money) | Stores Discount of Membership. |
| MembershipType | Alphanumeric (20 characters long) | Description of what type of member they are |
| MemberID | Alphanumeric (5 characters long) | Stores Primary key for the member ID as a foreign key. |

1. Manager(**ManagerID**, ManagerFirstName, ManagerLastName, ManagerPhone, ManagerEmail)

This relation holds the data about Manager attributes, data type, and description. The following table describes its attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| ManagerID | Alphanumeric (5 characters long) | Stores Primary key for the Manager ID. |
| ManagerFirstName | Alphanumeric (Up to 30 characters long) | Stores the First name of the Manager. |
| ManagerLastName | Alphanumeric (Up to 30 characters long) | Stores the Last name of the Manager. |
| ManagerPhone | Alphanumeric (11 characters long) | Stores Manager Phone number. |
| ManagerEmail | Alphanumeric (Up to 30 characters long) | Stores the Managers Email. |

1. Class(**ClassID**, ClassName*,* ClassFee, ClassDescription, ClassType)

This relation stores data about classes in Pro Health Club. The attributes are described in the table below.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| ClassID | Alphanumeric(5 characters long) | Stores primary key for class ID. |
| ClassName | Alphanumeric(Up to 30 characters long) | Stores class name. |
| ClassFee | Decimal (money) | Stores the classes cost. |
| ClassDescription | Alphanumeric(Up to 50 characters long) | Describes activities the class holds. |
| ClassType | Alphanumeric(Up to 20 characters long) | Stores the type of class available. |

6. PersonalClass(**ClassID**, Fee)

This relation stores data for costs to join a personal class. The table below describes the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| Fee | Decimal (money) | Stores the Personal Class cost. |
| ClassID | Alphanumeric (5 characters long) | Stores the primary key for Class |

1. Instructor(**InstructorID**, InstructorFirstName , InstructorLastName, InstructorPhone, InstructorEmail, InstructorAddress)

This relation stores data about the instructors of Pro Health club. The following table describes the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| InstructorID | Alphanumeric(5 characters long) | Stores primary key for Instructor ID. |
| InstructorFirstName | Alphanumeric(Up to 30 characters long) | Stores Instructors first name |
| InstructorLastName | Alphanumeric(Up to 30 characters long) | Stores Instructors last name |
| InstructoPhone | Alphanumeric (11 characters long) | Stores Instructor Phone number |
| InstructorEmail | Alphanumeric(Up to 30 characters long) | Stores Instructor Email |
| InstructorAddress | Alphanumeric(Up to 50 characters long) | Stores Instructor's address |

1. PaymentMethod(**PaymentID**, PaymentDate, PaymentAmount, AuthorizationNumber, PaymentType*, MembershipID*)

This relation holds data about the payment method. The following table describes the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| PaymentID | Alphanumeric(5 characters long) | Stores primary key for Payment ID. |
| PaymentDate | Date (MM/DD/YYYY) | Stores payment Date of the membership |
| PaymentAmount | Decimal (money) | Stores the payment amount for membership fee. |
| AuthorizationNumber | Alphanumeric(Up to 10 characters long) | Stores the authorization number for the payment made. |
| PaymentType | Alphanumeric(10 characters long) | Stores the type of payment made (debit/credit) |
| MembershipID | Alphanumeric(5 characters long) | Stores Foreign key for Payment Method relating to Membership |

1. Attendance(**AttendanceID,** MemberAttendance, AttendanceDate, AttendanceStatus, *ScheduleID*)

This relation stores data about the attendance in Pro Health club, the attributes are described in the table.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| AttendanceID | Alphanumeric(5 characters long) | Stores primary key for attendance. |
| MemberAttendance | Numeric(3 characters long) | Stores how many classes a member has attended. |
| AttendanceDate | Date (MM/DD/YYYY) | Stores attendance Date |
| AttendanceStatus | Alphanumeric(5 characters long) | Stores the attendance status of members |
| ScheduleID | Alphanumeric(5 characters long) | Stores foreign key for Attendance relating to MemberClassSchedule |

1. MemberClassSchedule(***ScheduleID, MemberID***, StartTime, EndTime, StartDate, EndDate, ClassFeedback)

This relation holds data about the membership class schedule. The following table describes the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| Schedule ID | Alphanumeric(5 characters long) | Stores the primary key from ClassSchedule as a foreign and primary key |
| MemberID | Alphanumeric(5 characters long) | Stores the primary key from Member as a foreign and primary key |
| StartTime | TimeStamp(HH:MM:SS) | Stores Start Time of the membership class schedule |
| EndTime | TimeStamp(HH:MM:SS) | Stores End Time of the membership class schedule |
| StartDate | Date (MM/DD/YYYY) | Stores Start Date of the membership class schedule |
| EndDate | Date (MM/DD/YYYY) | Stores End Date of the membership class schedule |
| ClassFeedback | Alphanumeric(Up to 150 characters long) | Stores the Class Feedback for the membership Class Schedule. |

1. PersonalTrainer**(InstructorID,** Fee, Duration)

This relation stores data on  the personal Trainer details of Pro Health Club. The table below describes the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| InstructorID | Alphanumeric(5 characters long) | Holds Primary key for Personal Trainer |
| Fee | Decimal (money) | Stores the Personal Trainer cost. |
| Duration | TimeStamp(HH:MM:SS) | Stores Duration of the Personal Trainer |

1. ClassSchedule(**ScheduleID**, ClassTime, ClassDate, SeatLimit, *ClassID, InstructorID, ManagerID*)

This relation stores data about the class schedule details for the Pro Health Club. The table below describes the attributes.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| ScheduleID | Alphanumeric (5 characters long) | Stores primary key for SheduleID |
| ClassTime | DateTime(HH:MM:SS) | Stores ClassTime of the class schedule |
| ClassDate | Date (MM/DD/YYYY) | Stores Class Date of the Class Schedule |
| SeatLimit | Numeric(3 characters long) | Stores the seat limit of the Class Schedule. |
| ClassID | Alphanumeric(5 characters long) | Holds the foreign key for Class |
| InstructorID | Alphanumeric(5 characters long) | Holds the foreign key for Instructor |
| ManagerID | Alphanumeric (5 characters long) | Holds Primary key for the Manager ID as a foreign key |

Normalization of Relations

To ensure ease of maintenance and modification, all relations in the RDM must be in third normal form. Doing so requires normalization or simplification of each relation, as shown below. The various forms (1NF, 2NF, 3NF) are based on the type of dependency each attribute has with each other and their relation.

First normal forms have non-atomic attributes, meaning that every row must only have one value per column. For second normal forms, relations have no partial dependency between their primary key and non-key attributes. Finally, a relation is in third normal form if there is no transitive dependency between any key attributes. Below, each relation is being tested for the third normal form.

**Relation**: Club(**ClubID**, ClubName, ClubPhone, ClubEmail, ClubAddress)

The Club is in 1NF because each record or tuple has a single value for attribute. It also meets the qualifications for 2NF because there are no partial dependencies between the attributes and the primary key. There are also no transitive dependencies between non-key attributes, thus making it 3NF.

**Relation**: Member(**MemberID**, MemberLastName, MemberFirstName, MemberAddress, MemberEmail, MemberPassword, AccountNumber, *ClassID*)

The Member class is in 1NF because all of its attributes have a single value. It is also 2NF because each attribute is entirely dependent on the primary key. Member is also in 3NF because non-key attributes are not dependent on each other.

**Relation**: Manager(**ManagerID**, ManagerFirstName, ManagerLastName, ManagerPhone, ManagerEmail)

The Manager relation is in 1NF since each attribute has its own value. The relation is also in 2NF because every non-key depends fully on the Manager attribute. It is also in 3NF since non-key attributes do not depend on each other.

**Relation**: Membership(**MembershipID**, StartDate, EndDate, MembershipFees)

The Membership relation’s attributes all have a single value, making it 1NF. However, two of the attributes (membershipType, membershipDiscount) had a mutual dependency. To fix this, a new relation had to be made. After doing such, Membership was able to reach 2NF as all of the attributes were only dependent on the primary key. This also allowed it to be in 3NF.

**Relation**: MembershipTypeCategory(**membershipType**, membershipDiscount)

The MembershipTypeCategory was created to achieve 3NF in the Membership relation. All attributes have single values, thus making it 1NF. MembershipDiscount is dependent on membershipType, making it 2NF. Because there is only one non-key attribute, the relation is in 3NF.

**Relation**: Instructor(**InstructorID**, InstructorFirstName, InstructorLastName, InstructorPhone, InstructorEmail, InstructorAddress)

The Instructor relation is in 1NF since each attribute stores a single value for each tuple. The relation is also in 2NF because all the non-key attributes fully depend on the primary key InstructorID. There is also a relation in 3NF because all the non-key attributes are independent on their own.

**Relation**:PaymentMethod(**PaymentID**, PaymentDate, PaymentAmount, AuthorizationNumber, PaymentType*, MembershipID*)

The PaymentMethod relation is in 1NF since each attribute has its own value. The relation is also in 2NF because every non-key depends fully on the PaymentMethod attribute. It is also in 3NF since non-key attributes do not depend on each other at all.

**Relation**: Attendance(**AttendanceID**, MemberAttendance, AttendanceDate, AttendanceStatus, *ScheduleID*)

The Attendance relation is in 1NF because each attribute stores a single value for each tuple. It is also in 2NF because the primary key is not a composite key. None of the non-key attributes are mutually dependent on each other. So, the relation is also in 3NF.

**Relation**: MemberClassSchedule(***MemberID***, StartTime, EndTime, StartDate, EndDate)

The MemberClassSchedule relation is in 1NF since each attribute has its own value. The relation was not in 2NF, however, because the Times and Dates are unique to the Member, not the class schedules. To fix this, a new relation had to be made. Now it is in 2NF and 3NF since non-key attributes do not depend on each other.

**Relation:** MemberClassScheduleFeedbackCategory(***ScheduleID,*** ClassFeedback)

This relation is in 1NF as each attribute has its own value. It is also in 2NF as ClassFeeback there is only one non-key attribute. This also has it qualify for 3NF.

**Relation**: Class(**ClassID**, ClassName, ClassDescription)

The Class relation met all 1NF requirements as each attribute had single values. However, it did not achieve 2NF as ClassType and ClassFee had a transitive dependency. As such, the attributes were moved to their own relation. This allowed Class to achieve 3NF as none of the remaining non-key attributes were dependent on one another.

**Relation**: ClassTypeCategory(**ClassType**, ClassFee)

The ClassTypeCategory was created to prevent transitive dependecies in the Class relation. Because fees are dependent on class type, a new relation was made. As both attributes have single values, the relation is in 1NF. It is also in 2NF and 3NF as there is only one non-key attribute that is dependent on the composite key.

**Relation**: ClassSchedule(**ScheduleID**, ClassTime, ClassDate, SeatLimit,*ClassID*, *InstructorID, ManagerID*)

The ClassSchedule relation is in 1NF since each attribute has its own value. The relation is also in 2NF because every non-key depends fully on the ScheduleID attribute. It is also in 3NF since non-key attributes do not depend on each other.

**Relation**: PersonalTrainer(**InstructorID**, Fee, Duration)

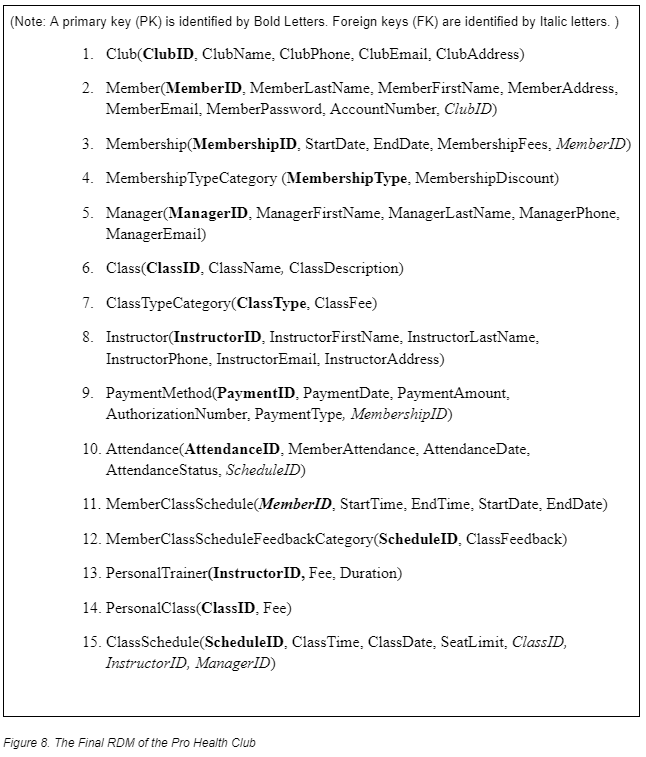
The PersonalTrainer relation is in 1NF and 2NF as per the definitions—each attribute holds only a single value, and the non-key attributes are fully dependent on the primary key. There is no dependency between the Fee and Duration. So, it is also in 3NF

**Relation**: PersonalClass(**ClassID**, Fee)

The PropertyClass relation is in 1NF since there is a single value in PersonalClass for each tuple. It is also in 2NF because Fee is fully dependent on the composite primary key. The relation is also in 3NF since there is no issue of mutual dependency.

Final Relational Data Model for the Pro Health Club

There was no change in the initial RDM due to normalization. So, the final relational data model is the same as the initial RDM. It is shown in figure 8.



Validation of the Relational Data Model

Here, we cross-check the user queries against the relational data model to make sure that we have appropriate relations and attributes included in the model to support the user queries and

other requirements. The database user queries were specified as part of the functional requirements.

For each query, we show the sequence of relations and the attributes required for each query and make sure that these relations and the attributes are indeed present in the RDM.

**Query #1**. List all Memberships.

**Given:** Memberships are not given as an input, but all memberships can be listed.

**Relations:** Membership(**MembershipID**, StartDate, EndDate, MembershipFees, *MemberID*)

We note that the relation and the required attributes are present in the RDM.

**Query #2.** List all the memberships sold in a given period.

**Given:** Memberships sold in a given period are an input found in the StartDate attribute.

**Relations:** Membership(MembershipID, StartDate, EndDate, MembershipFees, *MemberID*)

We check and see that the relation and the required attributes are present in the RDM.

**Query #3**. List all the Members.

**Given:** No inputs are given. Member details are found in the Members relation.

**Relations:** Member(MemberID, MemberLastName, MemberFirstName, MemberAddress, MemberEmail, MemberPassword, AccountNumber, *ClubID*)

The query does not require any changes to the RDM.

**Query #4**. List all Instructors assigned to a Class.

**Given:** Instructors and their classes need to be given as an input in the query. We have to list ClassSchedule details to find the assigned instructors.

**Relations:** ClassSchedule (**ScheduleID**, ClassTime, ClassDate, SeatLimit, *ClassID*, *InstructorID, ManagerID*)

Instructor (InstructorID, InstructorFirstName, InstructorLastName, InstructorPhone, InstructorEmail, InstructorAddress)

The query does not require any changes to the RDM. All the required data is present in RDM.

**Query #5**. List all personal trainers and their schedules.

**Given:** The given criteria are personal trainers, who are also instructors with assigned class schedules. Personal trainers are connected to the instructor through InstructorID, which is also a foreign key found in the class schedule.

**Relations:** ClassSchedule (**ScheduleID**, ClassTime, ClassDate, SeatLimit, *ClassID*, *InstructorID*)

Instructor (**InstructorID**, InstructorFirstName, InstructorLastName, InstructorPhone, InstructorEmail, InstructorAddress)

PersonalTrainer (**InstructorID**, Fee, Duration)

The query does not require any changes to the RDM. All the required data is present in RDM.

**Query #6**. List all the personal class schedules

**Given:** Personal Class schedules are a given input that can be found in the ClassID foreign key.

**Relations:** PersonalClass(**ClassID**,Fee)

ClassSchedule(**ScheduleID**, ClassTime, ClassDate, SeatLimit, *ClassID, InstructorID, ManagerID*)

The query does not require any changes to the RDM. All the required data is present in RDM.

**Query #7.** List all members of their class attendance.

**Given:** Given inputs are found in the member relation, which is connected to the membership class schedule through Member ID. Attendance is connected to Member Class Schedule through its primary composite key, scheduleID.

**Relations:** Attendance (**AttendanceID** MemberAttendance, AttendanceDate, AttendanceStatus, *ScheduleID*)

MemberClassSchedule (***ScheduleID***, ***MemberID*,** StartTime, EndTime, StartDate, EndDate, ClassFeedback)

Member (**MemberID**, MemberLastName, MemberFirstName, MembershipID, MemberAddress, MemberEmail, MemberPassword, AccountNumber*, ClubID*)

The query does not require changes to the RDM.

**Query #8**. List membership enrollment.

**Given:** Membership enrollment is a given input from StartDate found in the Membership relation.

**Relations:** Membership(**MembershipID**, StartDate, EndDate, MembershipFees, *MemberID*)

The query does not require any changes to RDM. All the required data is present in RDM.

**Query #9**. List member feedback and class rating

**Given:** The inputs given are found in theMembers relation, which is connected to MemberClassSchedule by MemberID.

**Relations:** MemberClassSchedule (***ScheduleID***, ***MemberID***, StartTime, EndTime, StartDate, EndDate, ClassFeedback)

Member(**MemberID**, MemberLastName, MemberFirstName, MembershipID, MemberAddress, MemberEmail, MemberPassword, AccountNumber, *ClubID*)

The query does not require any changes to RDM. All the required data is present in RDM.

**Query #10**. Compute all membership fees.

**Given:** Membership fees are an input found in the Membership relation.

**Relations:** Membership(**MembershipID**, StartDate, EndDate, MembershipFees, *MemberID*)

The query validation did not require any changes/modifications to the final relational data model.

**Query #11.**   Compute all personal class fees.

**Given:** The fee attribute in Personal Class is a given input.

**Relations:** PersonalClass(**ClassID**, Fee)

The query validation did not require any changes/modifications to the final relational data model.

**Query #12**. Display current revenue.

**Given:** Revenue is discoverable through membership fees. We can determine the current revenue through the StartDate found in the Membership relation.

**Relation:** Membership(**MembershipID**, StartDate, EndDate, MembershipFees, *MemberID*)

The query validation did not require any changes/modifications to the final relational data model.

**Query #13**. Display the previous year’s revenue.

**Given:** Revenue is discoverable through membership fees. We can tell if the revenue was from a previous year based on the input found in the Membership relation’s StartDate.

**Relation:** Membership(**MembershipID**, StartDate, EndDate, MembershipFees, *MemberID*)

The query validation did not require any changes/modifications to the final relational data model.

MS-3 Pro Health Club System UI Design (Deliverable #3)

Introduction

It is a web-based system that describes a user-interface design of the Pro-Health Club. We describe the master layout of the web pages. We use the storyboard to show the website navigation and the interaction between the users and the system... A storyboard specifies how a system responds in reaction to users’ actions. The storyboard can include drawings and pictures. The UI design includes a menu bar design and the webpage layout for each function (use case). This deliverable shows only a few web pages for illustration purposes.

Master Layout (Template) for the webpages

For the webpages Figure 9 shows an overall template to be used by the system. The webpage layout consists of the following major subsections:

1.  Menu bar. The menu bar contains the menu items.

2.  Header. This subsection includes a heading and, optionally, pictures/photos.

3.  Webpage Title. Each webpage will have its own title.

4.  Work Area. This subsection shows the content.

5.  Footer. This subsection is used to contain general information, status, and links.

Graphical user interface, text, application, email

Description automatically generated

Menu bar Design

A menu bar shows a link (menu items) for each use case in the use case model. The use case model is the major input to structuring the menu bar. For the Pro Health Club, the use case model below has the following use cases to group the lower-level use cases:

1. Maintain Memberships
   1. Enroll New Members
   2. Provide Feedback
   3. Register for General Classes
   4. Register for Personal Classes
   5. Review Feedback
   6. Process Payment
   7. List Members
2. Maintain Instructors
   1. Hire Instructors
   2. Assign Instructors to Classes
   3. Monitor Instructors
   4. Assign Trainers to Classes
   5. Monitor Trainers
   6. List Instructors
3. Maintain Classes
   1. Process Payment
   2. Maintain Class Schedule
   3. Offer Classes
   4. View Class Schedule
   5. Monitor Class Participation
   6. List Classes

The use case model provides the hierarchical structure for the menu bar. A storyboard can show how the various use cases are accessible to the user. Figure 10 shows the storyboard for the Pro-Health Club. Each node in the storyboard represents a webpage (screen). The storyboard shows what webpages are navigable from a given webpage.

Figure 10. A partial storyboard showing the website's navigation for the Pro Health Club.

Each web page (i.e., use cases) will be made available to users as menu choices in the menu bar. The menu bar is shown in figure 11, as part of the About page. The menu bar covers all the use cases in the use case model.

A picture containing graphical user interface

Description automatically generated

Web Page Design for each Function (use case)

Each webpage will be based upon the same homepage structure, as shown in figure 11. The webpage below (figure 12) displays the member details and allows an authorized user to add, edit, or delete member information. Figures 13, 14, and 15 depict similar webpage designs and functionality. All webpages made available for authorized users will allow Instructor, Member, and Class changes.

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

**MS-4 System Component Design for the Pro Health Club**

Introduction

The application presented here is the system (subsystem) internal design for the Pro Health Club. The system internal design consists of internal interactions between classes (software) and their subsystem components. The two collaborate to support responsibilities depicted in our use case diagrams documented by the UI objects. The use case realizations listed in the following section use the Model-View-Controller pattern. Interactions among various classes use sequence diagrams for each use case or system function. We then use a class diagram to show the software classes and their interdependencies.

Use Case Realization

The menu bar for the Pro-Health Club system has the following use cases as menu choices.

1. Maintain Memberships
   1. Enroll New Members
   2. Provide Feedback
   3. Register for General Classes
   4. Register for Personal Classes
   5. Review Feedback
   6. Process Payment
   7. List Members
2. Maintain Instructors
   1. Hire Instructors
   2. Assign Instructors to Classes
   3. Monitor Instructors
   4. Assign Trainers to Classes
   5. Monitor Trainers
   6. List Instructors
3. Maintain Classes
   1. Process Payment
   2. Maintain Class Schedule
   3. Offer Classes
   4. View Class Schedule
   5. Monitor Class Participation
   6. List Classes

We realize the following use cases for this deliverable:

1. List Members Details (referred to as ‘Member’ in the menu bar). When clicked on this menu choice, it displays a list of member details.
2. List Instructor Details (referred to as ‘Instructor’ in the menu bar). When selected, it displays a list of instructor details.
3. List Classes (referred to as ‘Class’ in the menu bar). When selected, it displays a list of classes.
4. List Class Schedule. (a sub menu item under ‘Class’ in the menu bar). When selected, it shows the class schedules.

Figures 16, 17, 18, and 19 show the use case realization of List Members, List Instructors, List Classes, and List Class Schedules. Figure 18 shows the interaction among the four objects—HomePage(Browser), MemberController, MemberModel and MemberView.

Diagram

Description automatically generated

*Text

Description automatically generated*

*Figure 16. A sequence diagram showing List Member Use Case realization.*

Diagram

Description automatically generated

*Figure 17. A Sequence diagram Showing List Instructor Use Case Realization.*

Diagram, box and whisker chart

Description automatically generated

*Figure 18. A sequence diagram showing List Classes Use Case Realization.*

Diagram

Description automatically generated

*Figure 19. A sequence diagram showing Class Schedule Use Case Realization.*

Design Class Diagram

Interdependencies and their software classes are shown in the design class diagram (figure 20). The software classes and their dependencies are derived directly from the sequence diagrams. The design class diagram covers the four use cases discussed above.

Diagram

Description automatically generated

*Figure 20. Design Class Diagram—Pro Health Club*

System Component--Internal Structure

Subcomponents or design subsystems are used to organize related software classes into groups. Organization is based on view, functionality, or any other basis that provides good overall architecture. Here, we will divide our software classes into three subsystems: Views (presentation layer (UI) subsystem), Controllers (business layer control classes), and Models (data access layer and Business logic software classes). Each subsystem is described below. Refer to the design class diagram depicted in Figure 20 for a set of software classes to be organized into components.

Views

All software classes dealing with UI are grouped together as Views. Based upon the use case realizations, we have the following UI-related classes:

1. MemberView

2. ClassView

3. InstructorView

4. ClassScheduleView

5. Homepage

All UI related classes will be organized into a single software module named Views.

Controllers

This subsystem contains software classes needed to implement the coordination between Views and Models. The controller classes are:

1. MemberController

2. ClassController

3. InstructorController

4. ClassScheduleController

All controller-related classes will be organized into a single, separate software module.

Models

This subsystem contains software classes to implement data access and business logic. The model classes are:

1. MemberModel

2. ClassModel

3. InstructorModel

4. ClassScheduleModel

All model-related classes will be organized into a single, separate software module.

Pro Health Club does not have a third-party system for software classes to communicate with.

Deployment Model

A deployment model shows the physical nodes (computer hardware) such as client and server machines used to host the various components of an application (software system). The deployment model for the Pro Health Club application is shown in figure 21. The application is a web-based application. The nodes communicate with each other on the Internet using TCP/IP protocols.

*Graphical user interface, diagram, application

Description automatically generatedFigure 21. Pro Health Club Application Deployment Model*

This model shows the hosting location for the various software components of the Pro Health Club Application.