Vanier College

Computer Science and Technology Department

System Development

420-436-VA

Deliverable 6

Orange Team

Daniel Levitin (Team Leader)

Jericho Nieva

Philip Dubé

Client: FavoriteDesign stretch ceilings

4/21/2023

We, the orange team, certify that this project is our own work.

Code learned and used from our eCommerce class will be used for features of our project.

I, Philip Dubé, student ID#2145451, confirm that I have contributed to this deliverable.

[illegible]

I, Daniel Levitin, student ID# 2162749, confirm that I have contributed to this deliverable.

[illegible]

I, Jericho Nieva, student ID# 6161653, confirm that I have contributed to this deliverable.

[illegible]

Table of contents

Executive Overview

This document focuses on essential documentation related to the upcoming development of our database system. Specifically, it covers the future database system.

The document starts by outlining the business problems that our client has encountered, which remain the same.

The document follows with an updated Narrative Description of the Future Information System presented as a narration that provides a comprehensive account of the system's purpose and the final product from the user’s perspective, particularly concerning the database and the stored information. The section elaborates on the database’s construction and how it will be operated.

Starting with appendix 1, it contains a data dictionary, filled with the tables’ fields, their data type, field size, constraints, description and an entry example.

Following appendix 1, appendix 2 includes an Entity-Relationship diagram of the database complete with cardinalities. It also contains the differences and similarities between the aforementioned Entity-Relationship diagram and the class diagram from the 3rd deliverable.

Next, appendix 3 explains the multiple ways used to optimize our database and explains how our database system conforms to the normalization standards up to 3NF.

Finally, appendix 4 explains the access speed required, how our design will allow it, the amount of times our database will be accessed and the necessary response time.

The Business Problem

Our client has a problem with how they keep a record of their clients, as they currently are doing physically through paper and pencil/pen. This is inefficient for adding/removing/editing information, looking for a specific record, and doesn't allow to filter the records, at least not in a scalable way. It also makes it unrealistic to create a backup, which is an important risk to consider for a business that plans to exist for a long time. Furthermore, sending data or records to someone would involve them either recopying them, scanning them with a printer, or taking pictures, so the communication between the two owners isn't very efficient. With an online Database Management System, communication about certain clients becomes very easy, convenient, and the owner who doesn't currently have the booklet of records doesn't need to wait for the other owner to send it.

Another problem encountered by our client is that they don't have a landing page/poster for their different services that they offer, which makes it difficult to advertise in newspapers, online ads, etc. So, one solution is to create a nice and compact landing page, where clients get all of the important information, while also getting sold on the product as much as possible with the limited words.

Narrative Description

Our database system has only one actor, the owner of the company Favorite Design, Alexander Levitin. He will need to be signed in to be able to perform numerous tasks within the system. These include viewing, adding, editing and deleting records. These records, accessible to the user, are split into several different tables.

Our system starts with the login and authentication of the user. Each user will have a user id automatically assigned to them. Information such as their username as well as their password hash will be stored in the database for authentication purposes.

One of the tables accessible to the user is the Project table. To add a new project record, the user must fill out several fields pertaining to the new record. These fields include the job, the project cost, the start and end date, whether the project is done or not, the surface area, the amount of lights and more. When a new project is added, a unique project id will be automatically assigned to it as well as a client record, with linking to old clients through the client's name and address.

To keep track of the payments, the database will store payment information records. Information such as the payment id, the project's id, the payment method, the amount, the user's id and the date will be kept.

Within the database, information on the client will be stored in a table such as their client id, which is automatically assigned, their name and their address.

Another table accessible to the user is the Trip table. To add a new trip record, the user must fill out several fields pertaining to the new record. These fields include the address, the name of the client, as well as the distance traveled. When a new trip is added, a unique project id will be automatically assigned to it.

The last table that the user will be able to access is the Expense table. To add a new expense record, the user must fill out several fields pertaining to the new record. These fields include the supplier's name, the total expense, details, and which owner paid. Upon the addition of the record, a unique id will be assigned to it.

Appendix I - Data Dictionary

User					
Field Name	**Data Type**	**Field Size**	**Constraints**	**Description**	**Example**
user_id	int	11	Primary Key	A unique Id is assigned to each user	5
username	varchar	20	Not Null	A username that allows the user to log in	admin
password_hash	varchar	72	Not Null	A password hash that allows the user to log in	sdak12asdad 3a35atrpo7

Expense					
Field Name	**Data Type**	**Field Size**	**Constraints**	**Description**	**Example**
expense_id	int	11	Primary key	A unique Id is assigned to each expense	65
user_id	int	11	Foreign key	A unique Id is assigned to each user	5
supplierName	varchar	50	Not null	The name of the supplier	Rona
totalExpense	numeric	7	Not null	The total expense	249.99

details	text	500	Not null	The details of the expense	Screws, PVC film…

Trip					
Field Name	**Data Type**	**Field Size**	**Constraints**	**Description**	**Example**
trip_id	int	11	Primary Key	A unique Id is assigned to each trip	1
project_id	int	11	Foreign key	A unique Id is assigned to each project	67
client_id	int	11	Foreign key	A unique Id is assigned to each client	5
distance	int	3	Not null	The distance of the trip	200

PaymentInformation					
Field Name	**Data Type**	**Field Size**	**Constraints**	**Description**	**Example**
payment_id	int	11	Primary key	A unique Id is assigned to each paymentInformation	2
project_id	int	11	Foreign key	A unique Id is assigned to each project	67
user_id	int	11	Foreign key	A unique Id is	5

				assigned to each user	
paymentMethod	varchar	30	Not null	The payment method of the payment	Cash
amount	numeric	7	N/A	The total amount of the payment	1000.50
date	Date	10	N/A	The date the payment was made	04/22/2023

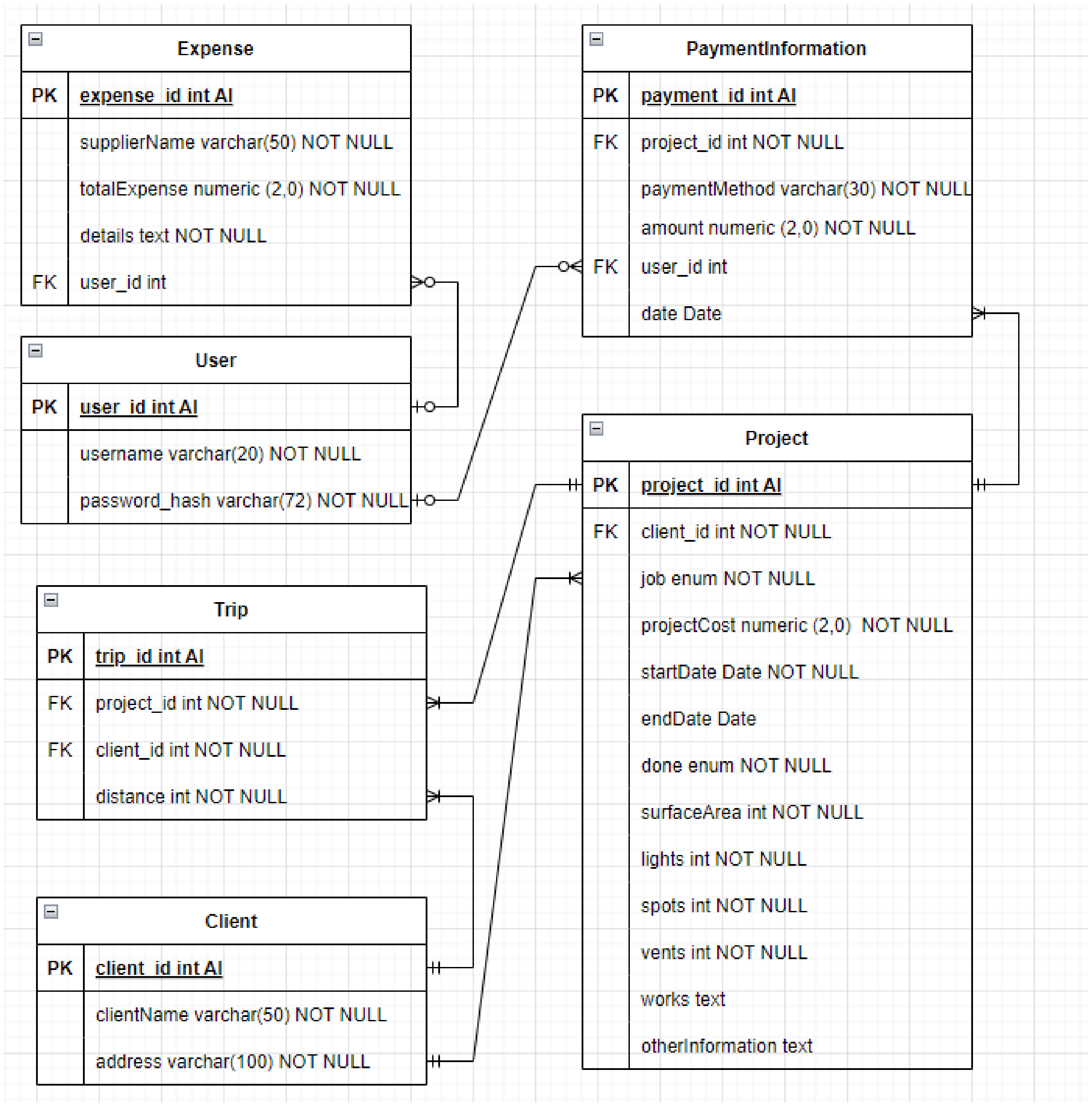
Client					
Field Name	**Data Type**	**Field Size**	**Constraints**	**Description**	**Example**
client_id	int	11	Primary Key	A unique Id is assigned to each client	5
clientName	varchar	50	Not null	The name of the client	John Doe
address	varchar	100	Not null	The address of the client	328 rue Laplante

Project					
Field Name	**Data Type**	**Field Size**	**Constraints**	**Description**	**Example**
project_id	int	11	Primary Key	A unique Id is assigned to each project	67
client_id	int	11	Foreign key	A unique Id is assigned to each	5

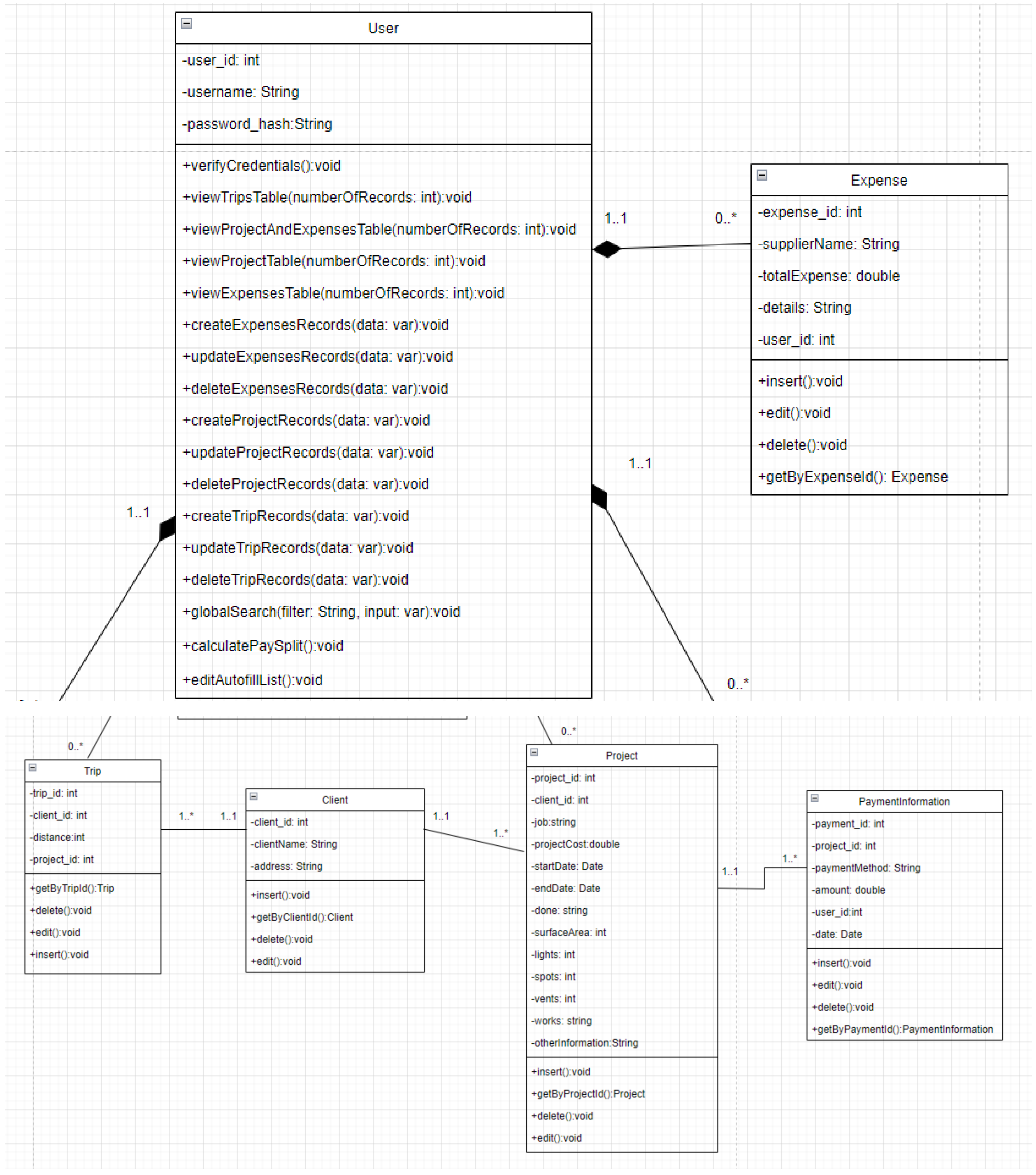
				client	
job	enum	3	Not null	The job done in the Trip	Service
projectCost	numeric	7	Not null	The total cost of the project	2001.00
startDate	date	10	Not null	The start date of the job	04/20/2023
endDate	date	10	N/A	The end date of the job	04/22/2023
done	enum	2	Not null	Whether the project is done	Not done
surfaceArea	int	3	Not null	The surface area of the project	200
lights	int	11	Not null	The number of lights used for the project	15
spots	int	11	Not null	The number of spots used for the project	7
vents	int	11	Not null	The number of vents used for the project	5
works	text	500	Not null	Extra works (more specific than “other information”)	Fix electricity
otherInformati on	text	500	Not null	Additional information regarding the project	Spare lights at the end

Appendix II - ERD Diagram | Class Diagram | Changes and Differences

ERD Diagram



Class Diagram



Similarities

The first similarity that can be pointed out between the two diagrams is that they share a common base of data, meaning that the data members in the class diagram are the same as the field in the entity-relationship diagram.

The second similarity that can be observed here is that integrity constraints in both diagrams are respected, meaning that the foreign keys in the ERD are displayed as data members in the class diagram.

A third similarity that can be observed is that given that the two units being connected are the same in both diagrams, the cardinalities/entity relationships correspond to each other.

Differences

One difference that is displayed between the two diagrams is the way that data is presented. In the ERD, data is represented as a whole, while in the class diagram, it is represented by its specific value data types. An example of this is that enum values in the ERD are represented as Strings in the class diagram.

One key difference that lies in the definition of the diagrams, is that the class diagram contains the methods that are going to be used for the website, which the ERD does not.

Finally, another difference that can be pointed out is that certain pieces of data have the same data types but have different names. An example of this is that values which are presented as double in the class diagram are presented as numeric in the ERD.

Appendix III - Optimisation and Normalization

Optimisation

In our database, we are going to avoid using “SELECT *” statements wherever possible and are going to instead specify the needed tables instead.

We’re also going to be using “JOIN” statements where needed, to allow for displaying of different types of records. This is mainly going to be needed in the main page, where we are going to be displaying expense and project records simultaneously.

Normalization

1NF

There are 2 requirements to having a database which conforms to 1NF standards:

- Each table cell should contain a single value.
- Each record needs to be unique.

In our case, we’ve made sure that each cell does indeed include only a single value. An example of this is the “startDate” and “endDate” in the trip table, where we separated those two values instead of keeping them as one. Another example where we made sure that each of our cells would contain only one value is the payment of projects, where we made a separate entity to represent the payment information, because not only is there a likely chance for there to be more than one payment per project, there’s also more than one piece of information attached to each payment. As for including other information about records that are exceptional, we added a text field for other information where needed, which counts as a single value.

As for making sure every record is unique, we’ve made sure of that by checking that each entity, record and field is broken down as much as it needs to be, as per the examples in the previous paragraph, and by using auto-increment values for our primary keys.

2NF

There are 2 requirements to having a database which conforms to 2NF standards:

- Be in 1NF
- All non-key attributes are fully functional dependent on the primary key

As determined in the previous section, our database is in accordance with the standards of 1NF, so the first requirement is met.

As for the second requirement, a partial dependency requires a table to contain a composite key. In our case however, we don't have any tables that fulfill this condition. Therefore, our database conforms to the two requirements of 2NF and to the 2NF standard.

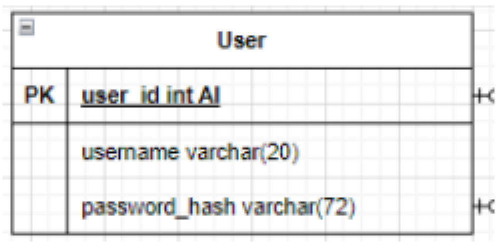
3NF

There are 2 requirements to having a database which conforms to 2NF standards:

- Be in 2NF
- Have no transitive functional dependency

A transitive functional dependency would mean that field A is dependent on field B which isn't part of the primary key, and field B is dependent on the primary key. In order to make sure this isn't the case in our database, I'll go through each table in our database.

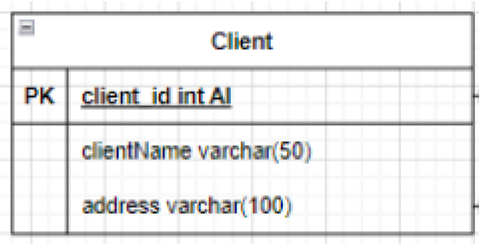
- User



User	
PK	user_id int AI
	username varchar(20)
	password_hash varchar(72)

The username and user password are indeed both dependent only on user_id.

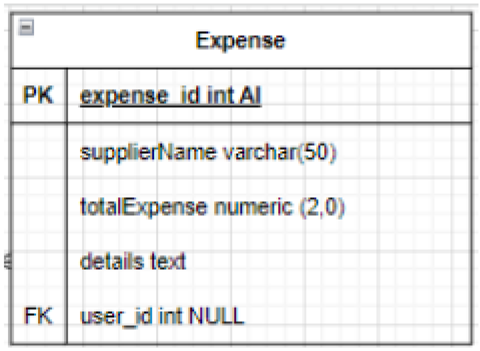
- Client



Client	
PK	client_id int AI
	clientName varchar(50)
	address varchar(100)

The client name is indeed dependent only on client_id. As for the client address, it's also dependent only on client_id, since the client name field isn't unique.

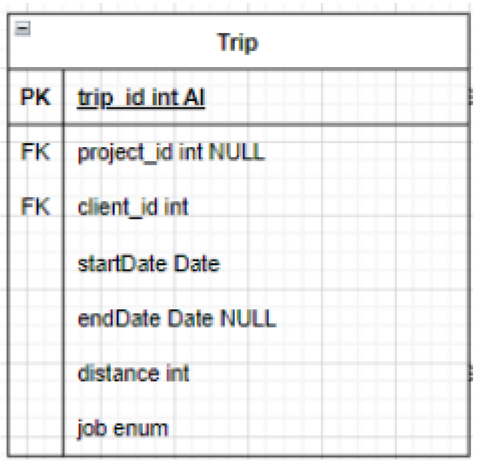
- Expense



	Expense
PK	expense_id int AI
	supplierName varchar(50)
	totalExpense numeric (2,0)
	details text
FK	user_id int NULL

The supplier name, the total expense, the details and the user_id, the last of which refers to one of the owners, all do indeed depend solely on the expense_id.

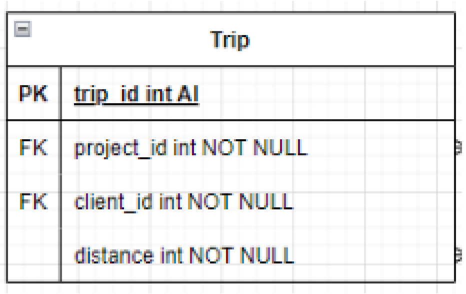
- Trip (before 3NF)



	Trip
PK	trip_id int AI
FK	project_id int NULL
FK	client_id int
	startDate Date
	endDate Date NULL
	distance int
	job enum

The project_id and the client_id, both of which are foreign keys in the trip table, are indeed both solely dependent on the trip_id in order to determine which specific record is being referenced. The distance is clearly dependent solely on the trip_id as well. However, the same cannot be said about the type of job and the start and end date, since they can be determined by the project_id, and are therefore dependent on it, causing a transitive functional dependency. In order to fix this dependency, we will need to move these data members into the project table, where the primary key is the project_id.

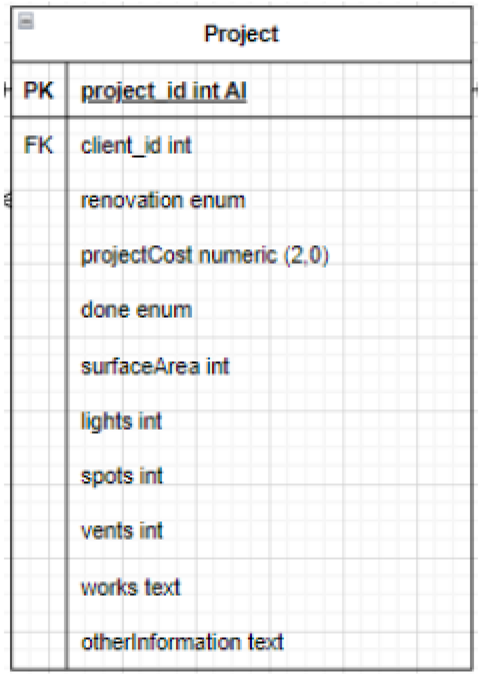
- Trip (after 3NF)



Trip	
PK	trip_id int AI
FK	project_id int NOT NULL
FK	client_id int NOT NULL
	distance int NOT NULL

Now that the start and end dates, as well as the type of job have been moved to the project entity, the rest of the data members are dependent only on the trip_id.

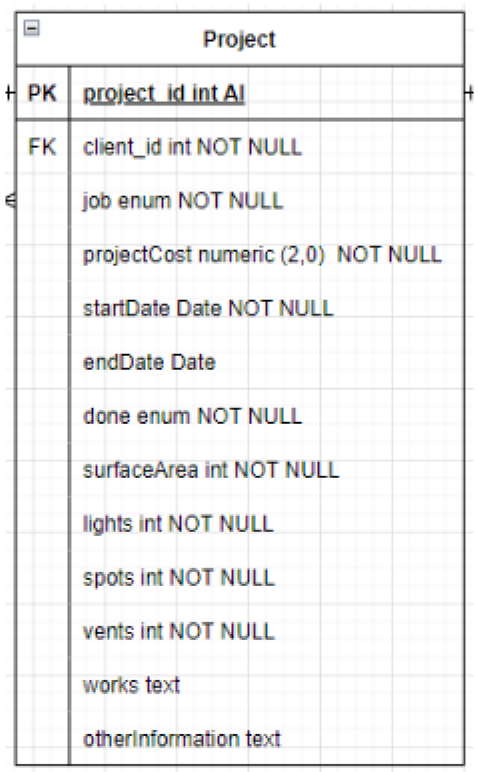
- Project (before 3NF)



Project	
PK	project_id int AI
FK	client_id int
	renovation enum
	projectCost numeric (2,0)
	done enum
	surfaceArea int
	lights int
	spots int
	vents int
	works text
	otherInformation text

The client_id, which is a foreign key in the project table, is dependent only on the project_id. The type of renovation, the project cost, the status of the project (done enum), the surface area, the number of lights, spots and vents, the works, and the other information field all depend only on the project_id and nothing else, as they are all non-unique details about a specific project.

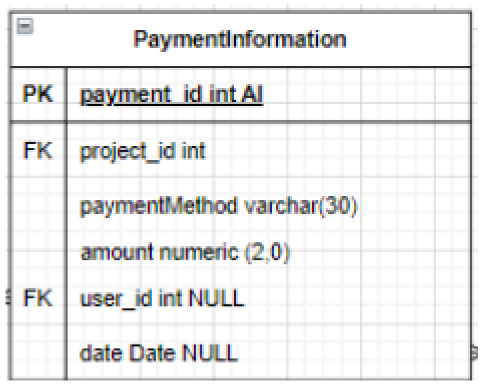
- Project (after 3NF)



	Project
PK	project_id int AI
FK	client_id int NOT NULL
	job enum NOT NULL
	projectCost numeric (2,0) NOT NULL
	startDate Date NOT NULL
	endDate Date
	done enum NOT NULL
	surfaceArea int NOT NULL
	lights int NOT NULL
	spots int NOT NULL
	vents int NOT NULL
	works text
	otherInformation text

Since certain data members in the trip table didn't correspond with the standards of 3NF as they were dependent on the project_id, which isn't part of the primary key in the trip table, they were moved to the project table. These new fields are the startDate, endDate, and job, the last of which replaced the renovation field, and are all dependent solely on the primary key project_id.

- Payment Information



	PaymentInformation
PK	payment_id int AI
FK	project_id int
	paymentMethod varchar(30)
	amount numeric (2,0)
FK	user_id int NULL
	date Date NULL

The project_id and the user_id (here referring to one of the owners), which are the foreign keys in the payment information table, both depend only on the payment_id field. The same can be said about the payment method, the amount and the date, which are all non-unique details about a specific payment. It should be noted that while the amount will usually be equal to half of the project cost, which can be determined by the project_id, this doesn’t always apply due to extra fees which can occur for a variety of reasons.

Appendix IV - Access Speed

In the case of our database, the access speed doesn't need to be super fast, since our only client is likely going to be Daniel's dad, the owner who takes care of administration. Therefore, we can afford for there to be a slight wait, although an excessive amount of waiting time that impedes on the efficiency of the record keeping is to be avoided. It should also be considered that our client's machine isn't equipped with the latest and fastest components, meaning that the system shouldn't be too overbearing in order to keep a good access speed. Our design will allow for the access speed to be reasonable using the optimizations of queries mentioned in the earlier section, as well as loading only what needs to be shown or accessed at the moment.

The database will be accessed as soon as the client logs in, since the home page is a combination of records from two tables joined together. As for the rest of the pages, the ones that have to do with viewing records will also require database access. With this we can conclude that most of the pages will require interaction with the database, since the whole system revolves around record keeping.

Mediography

(n.d.). Stretch Ceilings and Wall Decor in Montreal. Favorite Design. Retrieved February 23, 2023, from https://www.favoritedesign.ca/en/