Logo

Description automatically generated with medium confidence

**QuickGig**

Opportunities Ahead

*By:*

Mathew Migliore, Peter Rudolf, Tyler Segovia, and Kevin Romero

*Presented to:*

Logo, company name

Description automatically generated

*Date:*

2021-02-13

Table of Contents

Introduction2

Mission Statement3

Objectives4

Use Case Preface5

Use Case6

System Boundary Preface7

System Boundary8

Activity Diagram Preface9

Activity Diagram10

3NF Preface20

3NF List21

3NF Diagram23

Functional Dependencies Preface24

Functional Dependencies25

Data Dictionary Preface29

Data Dictionary30

Final Word33

Dictionary34

Index35

# **Introduction**

QuickGig is an app that allows users to post and search for job requests and make bids on small jobs. Users can post a picture and specifications of the job they want and make bids on the job based on how much they are willing to do the job for. The employer will choose the best bid. Employers will be able to allow their gig to been seen on the map and whether users can make bids on that job. The app will also notify you about jobs based on your profile and jobs that are posted that you would be interested in. Transactions done through the app will work with the employer puts down their balance, and when the job is completed, the worker will post a photo of the job and request the money. Both the employer and the employee will have a layer of security for them to get paid when the job is complete, or the employer to ensure that their money is not stolen if the job is not done, or something has been done incorrectly.

# **MISSION STATEMENT**

The purpose of QuickGig is to manage and promote job listings to simplify the process of finding a job for anyone.

# **OBJECTIVES**

- To maintain (enter, update, delete) data on users

- To maintain (enter, update, delete) data on posts

- To maintain (enter, update, delete) data on jobs

- To perform searches on users

- To perform searches on posts

- To perform searches on jobs

- To track the status of users

- To track the status of posts

- To track the status of jobs

- To report on users

- To report on posts

- To report on jobs

# **Use Case Preface**

For the use case we considered what the user should be able to see, there are two different types of employees to take into consideration. The employer and gig seeker, below we will explain of their roles and their views. Please make note that any of the two can be both a gig-seeker and an employer. The only moment where they differ is when they either post a gig or bid.

## **Employer:**

The employer’s main function is to post gigs that they want gig-seekers to fulfill. The employer will post a gig, they will write the description of the gig, an image, the location, and the budget. The employer can also accept or deny a bidder, we also consider the option of allowing the employer to maybe ask to raise or lower a bid. Once they have accepted their gig-seeker, they can see their jobs dashboard to see ongoing jobs, done jobs, and finalized jobs. They can also review a gig-seeker, or they can dispute a job.

## **Gig-Seeker:**

The gig-seeker’s main function is to place bids and accept posts, they look for posts throughout the site and will interact with posts with either their preference or miscellaneously. They can then post bids on the gigs they prefer for the amount that they would like. Similarly, to the employer once their bid has been accepted, they can see on going, done, and finalized jobs on the job dashboard. They can also review employers and dispute jobs.

Diagram

Description automatically generated

# **System Boundary Preface**

The system boundary was the starting point for our capstone, the team sat down and considered all the possible scenarios on how the users would interact with the Quickgig system. This meant that we would have to think of the different types of users that were going to be using the system, how they would relate to eachother, and each step that they would take through the page. Then we had to think about which pages would make this process the most efficient in its use as possible, This was the result.

# Diagram, schematic Description automatically generated**SYSTEM BOUNDARY**

**Activity Diagrams Preface**

**Introduction:** The activity diagrams were our second step after the system boundary, once we had thought of the pages we would be using, we thought about how each page would work and each of their functions. We had to think programmatically how the front page would interact with the back end, and the back end with the database.

**What do the diagrams explain?**

The diagrams explain several aspects of the website, most importantly API calls done to the backend with the frontend. They also explain database errors, form input errors, and backend errors. Lastly, they explain functional processes, such as the ranking algorithm for pay per review.

**How are they labeled?**

The diagrams are color labelled according to each page and functionality. Purple diagrams are for messages, pink diagrams are for replies, brown diagrams are for bids, black diagrams are for the profile, Blue diagrams are for general website functionalities, orange diagrams are for login/logout/register pages, green diagrams are for post pages, and yellow diagrams are for reviews.

**What do the colored boxes inside the diagrams represent?**

The boxes color represents the site’s stack. The cyan boxes represent the front-end (ReactJS), Green boxes represent the back-end (Express), orange containers represent the database (Sequelize), and the white boxes represent errors.

**Activity Diagrams**Diagram

Description automatically generated

**A picture containing text, screenshot, sign

Description automatically generated**

**Diagram

Description automatically generated**

**Activity Diagrams**

A screenshot of a computer

Description automatically generated with medium confidence

Diagram

Description automatically generated

Diagram

Description automatically generated

**Activity Diagrams**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Chart, diagram, funnel chart

Description automatically generatedA picture containing funnel chart

Description automatically generatedDiagram

Description automatically generated**

**Diagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

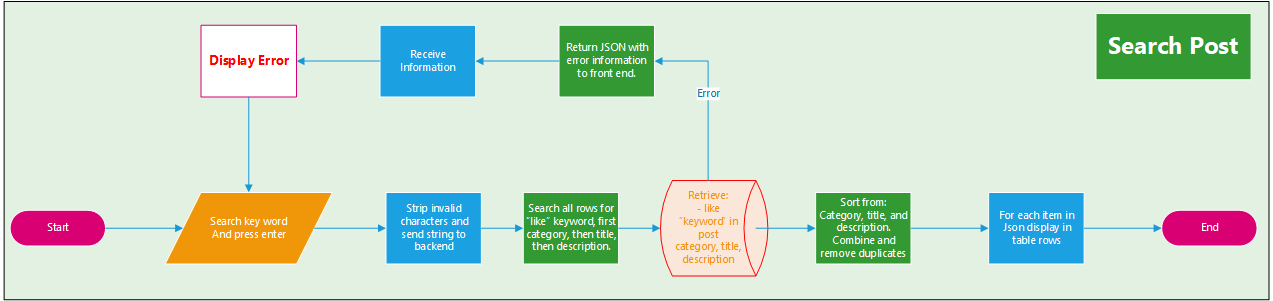
**Diagram

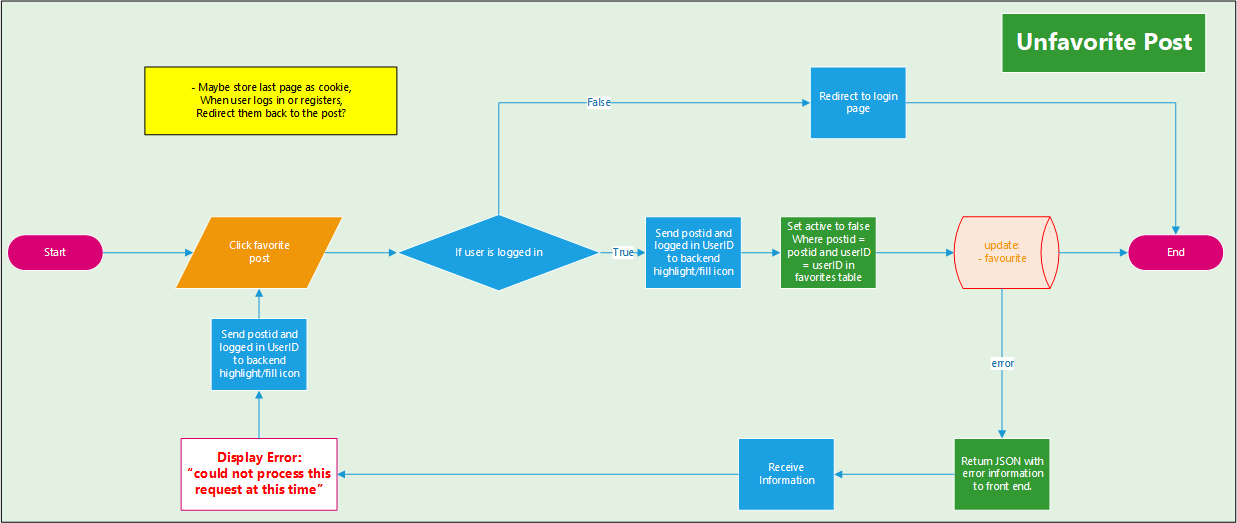
Description automatically generated**

**Diagram

Description automatically generatedDiagram

Description automatically generated**

**Activity Diagrams**

****

**Diagram

Description automatically generatedDiagram, Teams

Description automatically generated**

**3NF Preface**

**Introduction:** The 3NF diagram was an essential part to visualizing how data would be stored in the database and how it would interact with other data. We developed the 3NF diagram before the list because it displays relationships and makes it easier to understand how data will be created, modified, and deleted. This diagram was revised a number of times in order to ensure data normalization and remove the possibility of errors.

**How did we come up with the design?**

The design process started with the most important tables, such as ‘posts’ and ‘users’, and branched out from there. We had to discuss this process as a group and run through scenarios together to ensure that we did not miss anything. There were many revisions to this diagram throughout the creation of this report.

**What were the biggest challenges?**

The biggest challenge was ensuring that we did not miss anything while staying within the scope of the project. We came up with a lot of ideas that we had to mark down as ‘Extras’.

**Why did our group choose those tables?**

Everything in this diagram is essential to achieving our goal and vision for this project.

**Legend**

**Table Name (Primary Key (PK), Foreign Key (FK),** Attribute Name…**)**

Profile:

**users**(**userID (PK)**, **addressID (FK)**, **userStatusID (FK)**, email, password, firstName, lastName, phone, dateOfBirth, city, country, bio, preferencesIndex)

**favourites**(**favouriteID (PK),** **postID (FK), userID (FK)**, active, date)

**userStatus**(**userStatusID (PK)**,status)

**userPhotos**(**photoID (PK),** **userID (FK)**)

**loginHistory**(**loginID (PK)**, **userID (FK)**, date)

Posts:

**posts**(**postID (PK)** , **userID (FK)**, **addressID (FK)**, **postStatusID (FK)**, **catergoryID (FK),** postTitle, postDescription, setPrice, budget, postedDate, visible) optional: (views)

**contract**(**contractID (PK), userID (FK), postID (FK), contractStatusID (FK**), dateStart, dateEnd)

**postStatus**(**postStatusID (PK),** name) (open, accepted, complete, deleted)

**contractStatus**(**contractStatusID (PK)**, name) (open, hidden, accepted, disputed, done, finalized, deleted)

**postPhotos**(**photoID (PK),** **postID (FK)**)

**contractCategories**(**categoryID (PK),** categoryName, categoryDescription)

**bids**(**bidID (PK),** **postID (FK),** **userID (FK)**, price, priceType, **bidStatusID (FK)**, createdDate, responseDate, )

**bidStatus**(**bidStatusID (PK),** status)

**priceType**(**priceTypeID (PK),** priceType)

Reviews:

**review**(**reviewID (PK), contractID(fk), userID(fk), reviewerID(PK)**, comment, rating, date)

Messages:

**messages** (**messageID (PK), senderID(fk), sendToID(fk),** title, message, date)

Others:

**addresses**(**addressID (PK)**, line1, line2, city, postalCode, province, country)

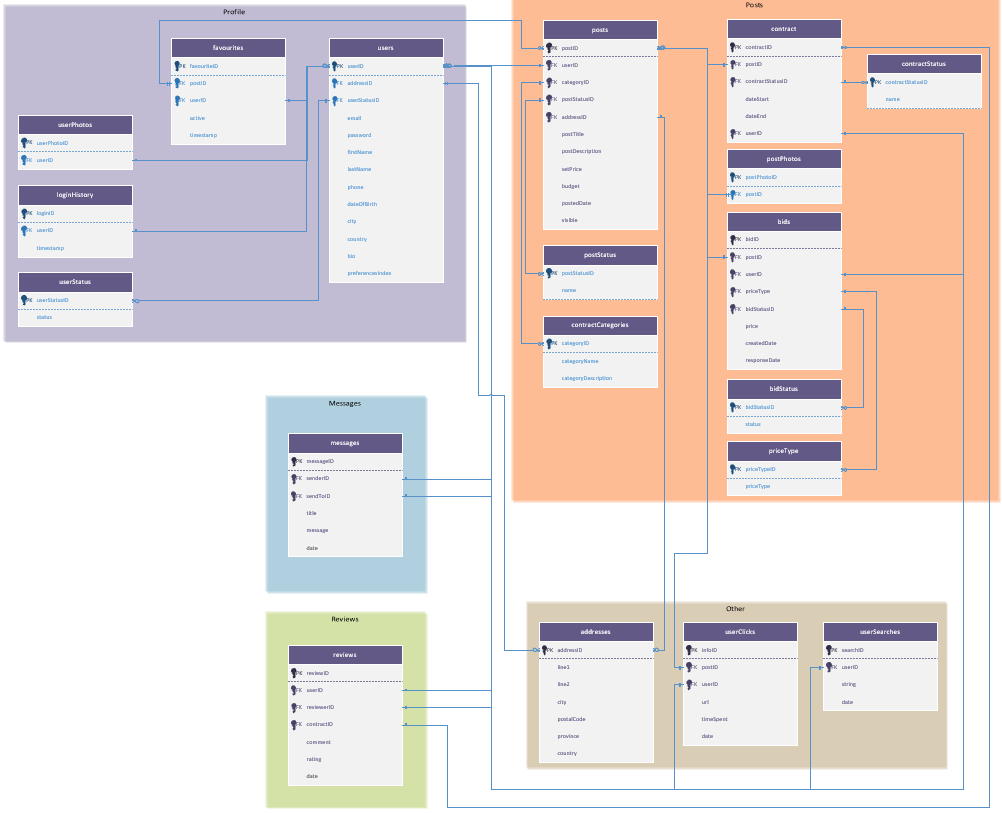
**userClicks**(**infoID(PK)**, **userID(fk)**, **postID(fk)**,url, timeSpent,date)

**userSearches(searchID(PK), userID(fk)**,string, date**)**

Extra:

* ~~Track what they’re searching for~~
* ~~Track the jobs the user bids on~~
* ~~Track time spent on page~~
* ~~Create personalized suggestions~~
* ~~Track number of clicks/visits a post gets~~
* ~~Hide reviews~~

# **3NF Diagram**



**Functional Dependencies Preface**

**Introduction:** For the functional dependencies we thought about which attributes would logically determine the others within the database, this is where the graph we made for the 3NF design came in handy, we were able to visualize within a table each attribute’s role within the database.

**How did we come up with the design?**

The standard Functional Dependency format was followed.

**Legend**

**Table Name:**

{Determining Attribute/s (X)} -> {Functionally Dependent Attribute/s (Y)}

**Functional Dependencies**

**users:**

{userID} -> {addressID, userStatusID, email, password, firstName, lastName, phone, dateOfBirth, city, country, bio, prefrencesIndex}

{email} -> {userID, addressID, userStatusID, password, firstName, lastName , dateOfBirth, city, country, bio, phone, prefrencesIndex}

{firstName, lastName, phone} -> {userID, addressID, userStatusID, password, dateOfBirth, city, country, bio, email, prefrencesIndex}

{firstName, lastName, addressID} **-> {**userID, userStatusID, password,email, phone, dateOfBirth, city, country, bio, prefrencesIndex}

**favourites:**

**{**favouriteID**}**  {postID, userID, active, date}

{postID, userID}-> {favouriteID, active, date}

userStatus:

**{**userStatusID**}->** {status}

userPhotos:

**{**photoID**}->** {userID}

loginHistory:

**{**loginID**} ->** {userID**,** date}

{userID, Date} -> {loginID}

**posts:**

**{**postID**}** -> **{**userID, addressID, postStatusID, categoryID, postTitle, postDescription, setPrice, budget, postedDate, visible}

**{**userID**,** postedDate**}** -> **{**postID, userID, addressID, postStatusID, categoryID, postTitle, postDesecription, setPrice, budget, postedDate, visible}

**contract:**

**{**contractID**}** -> **{**contractID, userID, postID, contractStatusID, startDate, endDate}

**{**userID**,** postID**} ->** {contractID, userID, postID, contractStatusID, startDate, endDate}

**{**userID**,** dateStart**}** **->** {contractID, userID, postID, contractStatusID, startDate, endDate}

postStatus:

**{**postStatusID**} -> {**name**}**

**{**name**} -> {**postStatusID**}**

contractStatus:

{contractStatusID} -> {name}

{name} -> {contractStatusID}

postPhotos:

{photoID} -> {photoID, postID}

contractCategories:

{categoryID} -> {categoryName, categoryDescription}

{categoryName} -> {categoryID, categoryName, categoryDescription}

bids:

{bidID} -> {bidID, postID, userID, bidStatusID, price, priceType, createdDate, responseDate}

{postID, userID, createdDate} -> {price, priceType, createdDate, responseDate}

bidStatus:

{bidStatusID} -> {status}

{status} -> {bidStatusID}

priceType:

{priceTypeID} -> {priceType}

{priceType} -> {priceTypeID}

review:

{reviewID} -> {reviewID, contractID, userID, reviewerID, comment, rating, date}

{contractID, userID, reviewerID} -> {reviewID, contractID, userID, reviewerID, comment, rating, date}

messages:

{messageID} -> {messageID, senderID, sendToID, title, message, date}

{senderID, sendToID, date} -> {messageID, senderID, sendToID, title, message, date}

addresses:

{addressID} -> {line1, line2, city, postalCode, province, country}

{line1, city, postalCode, province, country} -> {addressID}

userClicks:

{infoID} -> {userID, postID, url, timeSpent, date}

{date, userID} -> {infoID, userID, postID, url, timeSpent, date}

userSearches:

{searchID} -> {searchID, userID, string, date}

{userID, date} -> {searchID, userID, string, date}

**Data Dictionary Preface**

**Introduction:** This data dictionary is the blueprint to creating the database. We had to carefully design every field in every table to support the right data types. This data dictionary is based completely off of the 3NF diagram that we created and was updated many times to keep our design consistent. It also helped us identify and fix errors early on to avoid problems later on.

# **Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Data Type** | **References** | **Default** | **Not Null** |
| **< users >** | userID | INT |  | Identity | Y |
| **< users >** | addressID | INT | <addresses>.addressID |  | Y |
| **< users >** | userStatusID | INT | <userStatuses>.userStatusID |  | Y |
| **< users >** | email | VARCHAR |  |  | Y |
| **< users >** | password | VARCHAR |  |  | Y |
| **< users >** | firstName | VARCHAR |  |  | Y |
| **< users >** | lastName | VARCHAR |  |  | Y |
| **< users >** | phone | VARCHAR |  |  | Y |
| **< users >** | dateOfBirth | DATETIME |  |  | Y |
| **< users >** | city | VARCHAR |  |  | Y |
| **< users >** | country | VARCHAR |  |  | Y |
| **< users >** | bio | VARCHAR |  |  | N |
| **< users >** | preferencesIndex | VARCHAR |  |  | Y |
| **< favourites >** | favouriteID | INT |  | Identity | Y |
| **< favourites >** | postID | INT | <posts>.postID |  | Y |
| **< favourites >** | userID | INT | <userStatuses>.userStatusID |  | Y |
| **< favourites >** | active | BOOL |  |  | Y |
| **< favourites >** | date | DATETIME |  |  | Y |
| **< userStatus >** | userStatusID | INT |  | Identity | Y |
| **< userStatus >** | status | VARCHAR |  |  | Y |
| **< userPhotos >** | photoID | INT | <postPhotos>.photoID |  | Y |
| **< userPhotos >** | userID | INT | <user>.userID |  | Y |
| **< loginHistory >** | loginID | INT |  | Identity | Y |
| **< loginHistory >** | userID | INT | <user>.userID |  | Y |
| **< loginHistory >** | date | DATETIME |  |  | Y |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Data Type** | **References** | **Default** | **Not Null** |
| **< posts >** | postID | INT |  | Identity | Y |
| **< posts >** | userID | INT | <user>.userID |  | Y |
| **< posts >** | addressID | INT | <addresses>.addressID |  | Y |
| **< posts >** | postStatusID | INT | <postStatus>.postStatusID |  | Y |
| **< posts >** | categoryID | INT | <contractCategories>.categoryID |  | Y |
| **< posts >** | postTitle | VARCHAR |  |  | Y |
| **< posts >** | postDescription | VARCHAR |  |  | Y |
| **< posts >** | setPrice | MONEY |  |  | Y |
| **< posts >** | budget | MONEY |  |  | Y |
| **< posts >** | postedDate | DATETIME |  |  | Y |
| **< posts >** | visible | BOOL |  |  | Y |
| **< posts >** | views | INT |  |  | N |
| **< contract >** | contractID | INT |  | Identity | Y |
| **< contract >** | userID | INT | <user>.userID |  | Y |
| **< contract >** | postID | INT | <post>.postID |  | Y |
| **< contract >** | contractID | INT | <contractCategories>.categoryID |  | Y |
| **< contract >** | dateStart | DATETIME |  |  | Y |
| **< contract >** | dateEnd | DATETIME |  | Identity | Y |
| **< postStatus >** | postStatusID | INT |  | Identity | Y |
| **< postStatus >** | name | VARCHAR |  |  | Y |
| **< contractStatus >** | contractStatusID | INT |  | Identity | Y |
| **< contractStatus >** | name | VARCHAR |  |  | Y |
| **< postPhotos >** | photoID | INT |  | Identity | Y |
| **< postPhotos >** | postID | INT |  |  | Y |
| **< contractCategories >** | categoryID | INT |  | Identity | Y |
| **< contractCategories >** | categoryName | VARCHAR |  |  | Y |
| **< contractCategories >** | categoryDescription | VARCHAR |  |  | Y |
| **< bids >** | bidID | INT |  | Identity | Y |
| **< bids >** | postID | INT | <posts>.postID |  | Y |
| **< bids >** | userID | INT | <users>.userID |  | Y |
| **< bids >** | price | MONEY |  |  | Y |
| **< bids >** | priceType | VARCHAR |  |  | Y |
| **< bids >** | bidStatusID | INT | <bidStatus>.bidStatusID |  | Y |
| **< bids >** | createdDate | DATETIME |  |  | Y |
| **< bids >** | responseDate | DATETIME |  |  | Y |
| **< bidStatus >** | bidStatusID | INT |  | Identity | Y |
| **< bidStatus >** | status | VARCHAR |  |  | Y |
| **< priceType >** | priceTypeID | VARCHAR |  | Identity | Y |
| **< priceType >** | priceType | VARCHAR |  |  | Y |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Data Type** | **References** | **Default** | **Not Null** |
| **< review >** | reviewID | INT |  | Identity | Y |
| **< review >** | contractID | INT | <contractCategories>.categoryID |  | Y |
| **< review >** | userID | INT | <user>.userID |  | Y |
| **< review >** | reviewerID | INT |  | Identity | Y |
| **< review >** | comment | VARCHAR |  |  | N |
| **< review >** | rating | VARCHAR |  |  | N |
| **< review >** | date | DATETIME |  |  | Y |
| **< messages >** | messageID | INT |  | Identity | Y |
| **< messages >** | senderID | INT |  | Identity | Y |
| **< messages >** | sendToID | INT |  | Identity | Y |
| **< messages >** | title | VARCHAR |  |  | Y |
| **< messages >** | message | VARCHAR |  |  | Y |
| **< messages >** | date | DATETIME |  |  | Y |
| **< addresses >** | addressID | VARCHAR |  | Identity | Y |
| **< addresses >** | line1 | VARCHAR |  |  | Y |
| **< addresses >** | line2 | VARCHAR |  |  | N |
| **< addresses >** | city | VARCHAR |  |  | Y |
| **< addresses >** | postalCode | VARCHAR |  |  | Y |
| **< addresses >** | province | VARCHAR |  |  | Y |
| **< addresses >** | country | VARCHAR |  |  | Y |
| **< userClicks >** | infoID | INT |  | Identity | Y |
| **< userClicks >** | userID | INT | <user>.userID |  | Y |
| **< userClicks >** | postID | INT | <post>.postID |  | Y |
| **< userClicks >** | url | VARCHAR |  |  | Y |
| **< userClicks >** | timeSpent | DATETIME |  |  | Y |
| **< userClicks >** | date | DATETIME |  |  | Y |
| **< userSearches >** | searchID | INT |  | Identity | Y |
| **< userSearches >** | userID | INT | <user>.userID |  | Y |
| **< userSearches >** | string | VARCHAR |  |  | Y |
| **< userSearches >** | date | DATETIME |  |  | Y |

# **Final word**

We hope that the reader enjoys the report and the website, which can be found at (<https://www.quickgig.ca>), the entire team put a lot of work into the report, planning, programming, and organizing. If you have any question you can also find our contact information through our “About” page. Thank you for your attention.

Best Regards,

Mathew Migliore, Peter Rudolf, Tyler Segovia, & Kevin Romero.

**Dictionary**

* **DBMS:** Short for Database Management System. The system that manages the storage and querying of a database.
* **Query:** A logical request made to the database to return specified results.
* **Mission Statement:** Defines an understanding of what the mission of the project is and what the company wants to accomplish from this project.
* **Mission Objectives:** Defines in a simple format what the objectives of our team are for development in order to deem the project successfully

completed.

* **System Boundary:** Defines the boundaries that are internally in the facility and externally. It also defines the actors, items and other systems that interact with the company’s system.
* **Use Case:** Defines the objects and functions that the actors within the system interact with in each systematic cycle.
* **3NF:** Third Normal form, is the most common used normalization form to design a database which uses principles to reduce the duplication of data, avoid data anomalies, ensure referential integrity, and simplify data management.
* **Functional Dependencies:** defines a relationship between two attributes, typically between the PK and other non-key attributes within a table.
* **Data Dictionary:** a set of information describing the contents, format, and structure of a database and the relationship between its elements, used to control access to and manipulation of the database.
* **Attribute:** In terms of databases it refers to as a field. Used interchangeably with the term “column”
* **Column:** **:** In terms of databases it refers to as a field. Used interchangeably with the term “Attribute”
* **Primary Key:** The field used to Identify a record.
* **Foreign Key:** The field that reference the primary key of another table.

**Index**

3NF 21-23

3NF Preface 20

**A**

Activity Diagram 10-19

Activity Diagram Preface 9

**D**

Data Dictionary 30-32

Data Dictionary Preface 29

Dictionary 34

**F**

Final Word 33

Functional Dependency 25-28

Functional Dependency Preface 24

**I**

Introduction 2

**M**

Mission Statement 3

**O**

Objectives 4

**S**

System Boundary 8

System Boundary Preface 7

**U**

Use Case 6

Use Case Preface 5

**Logo

Description automatically generated with medium confidence**

**Disclaimer**

Quickgig is a fictional company for the computer programmer and analysis program’s capstone project for Durham College. Use of names, logos, references, and resources are for learning purposes only. All resources, names, logos, slogans, etc. were created by the hardworking students in the “Quickgig” Project, any similarities to external sources is purely coincidental.