# Group Project Part 1

Borislav Sabotinov, Yash Patel, Nicky Panda, Jaswanth Reddy

[CS5332 - Group 8 - Project Part 1 0](#_Toc63844)

[1. Requirement Gathering 1](#_Toc63845)

[2. Pros and Cons of Analysed systems(SurveyMonkey, LimeSurvey & Qualtrics) 10](#_Toc63846)

[3. Business Rules/Constraints 12](#_Toc63847)

[5. E-R Design of Each Entity 15](#_Toc63848)

[6. Relational Schema Diagram 25](#_Toc63849)

[6.1 Relational Schema Definition and Domain of all attributes 27](#_Toc63850)

[7. Domains of all attributes 28](#_Toc63851)

[8. Design Decision and Domain of all Attributes (For E-R Diagram) 41](#_Toc63852)

[8.1 Description for mapping E-R diagram to Relational Schema Diagram 50](#_Toc63853)

[9. Problems faced during the design of E-R Diagram and its Mapping 52](#_Toc63854)

[10. Not Included 52](#_Toc63855)

[11. Group Log 52](#_Toc63856)

# 1. Requirement Gathering

* We started the project by analyzing the project specification provided by the professor.
* We read the project description carefully and noted all the keywords, which helped us to find the Entities and their attributes.
* As per the project specification, we had gone through some online survey system websites and analyzed them to collect all the required information.
* Example:

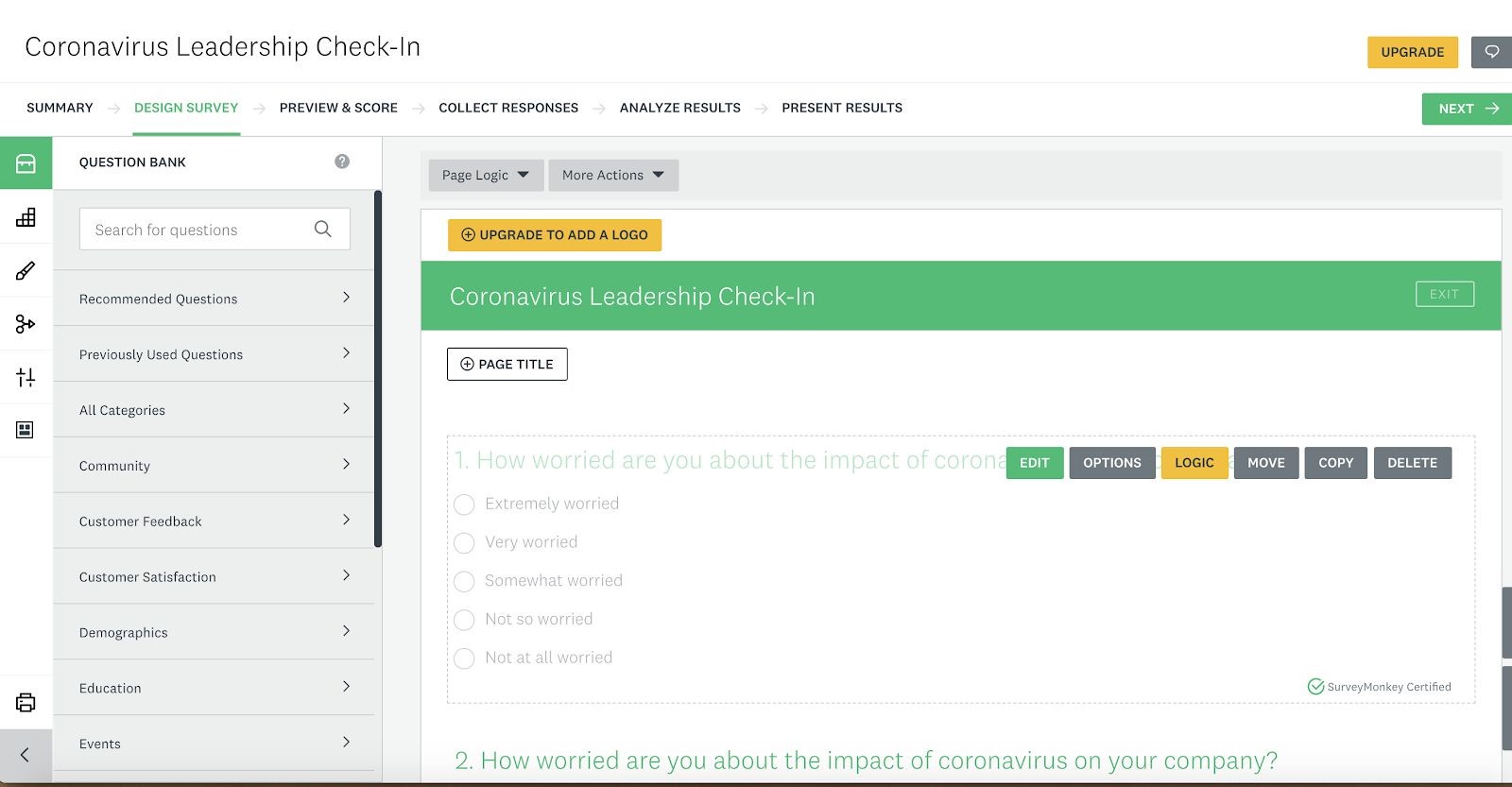
[**http://www.surveymonkey.com**](http://www.surveymonkey.com/)

**https://www.qualtrics.com/lp/survey-platform/ http://www.limesurvey.org**

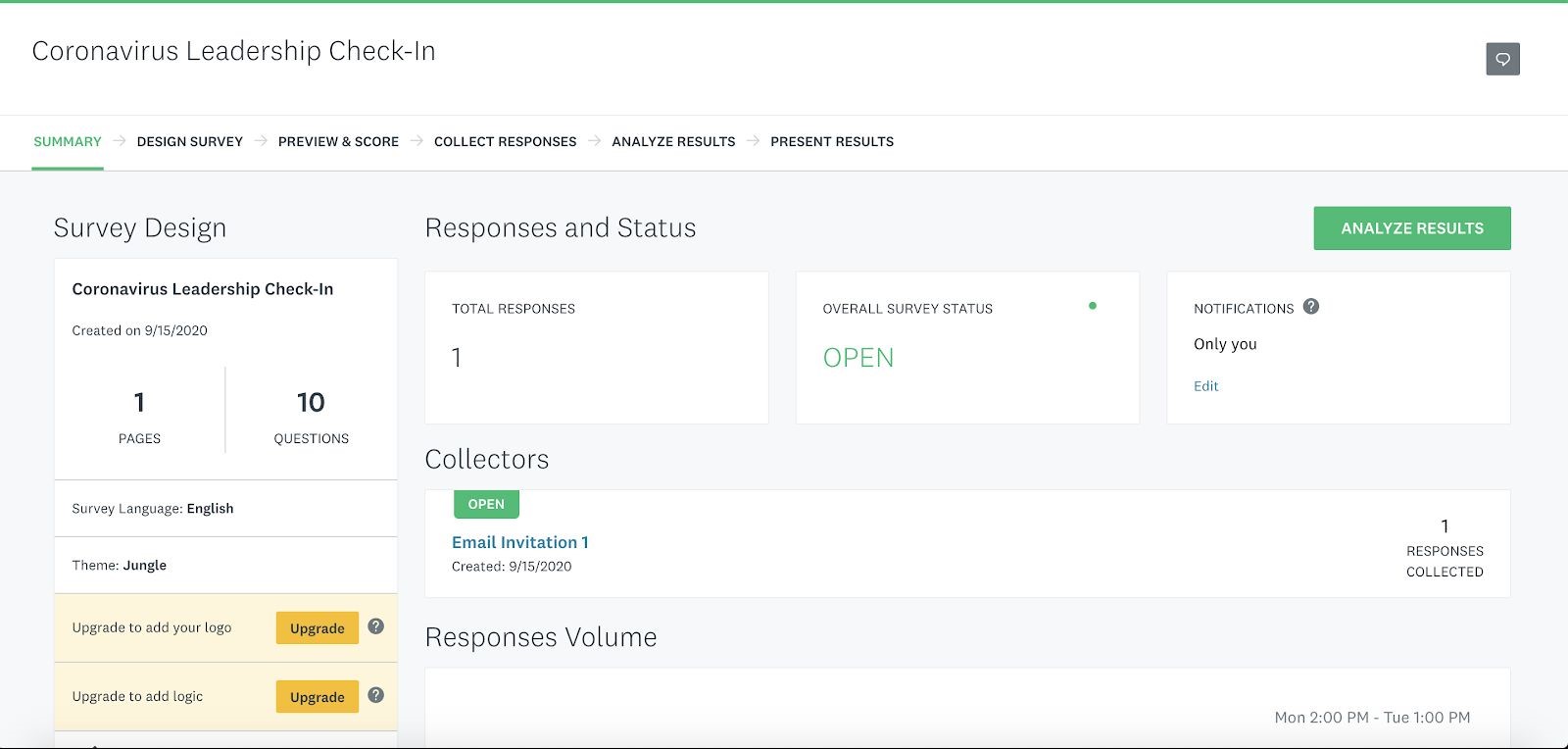
* We created an account in **http://www.surveymonkey.com**​ ​ website to create a sample survey to understand how an online survey system works and came across different types of questions and answers.
* We visited each section of the website to create a survey and with the help of knowledge we were able to identify each Entity and their attributes with values.
* To test the survey, we sent some invitations to get a responses to the survey we created
* We provided responses to the survey as an Admin and as a user to view how the individual response report looks.
* We also collected all the responses and viewed a summarized survey report.
* All the group members discussed project requirements and shared our ideas to finalized all the requirements for this project.

**Survey Monkey:**

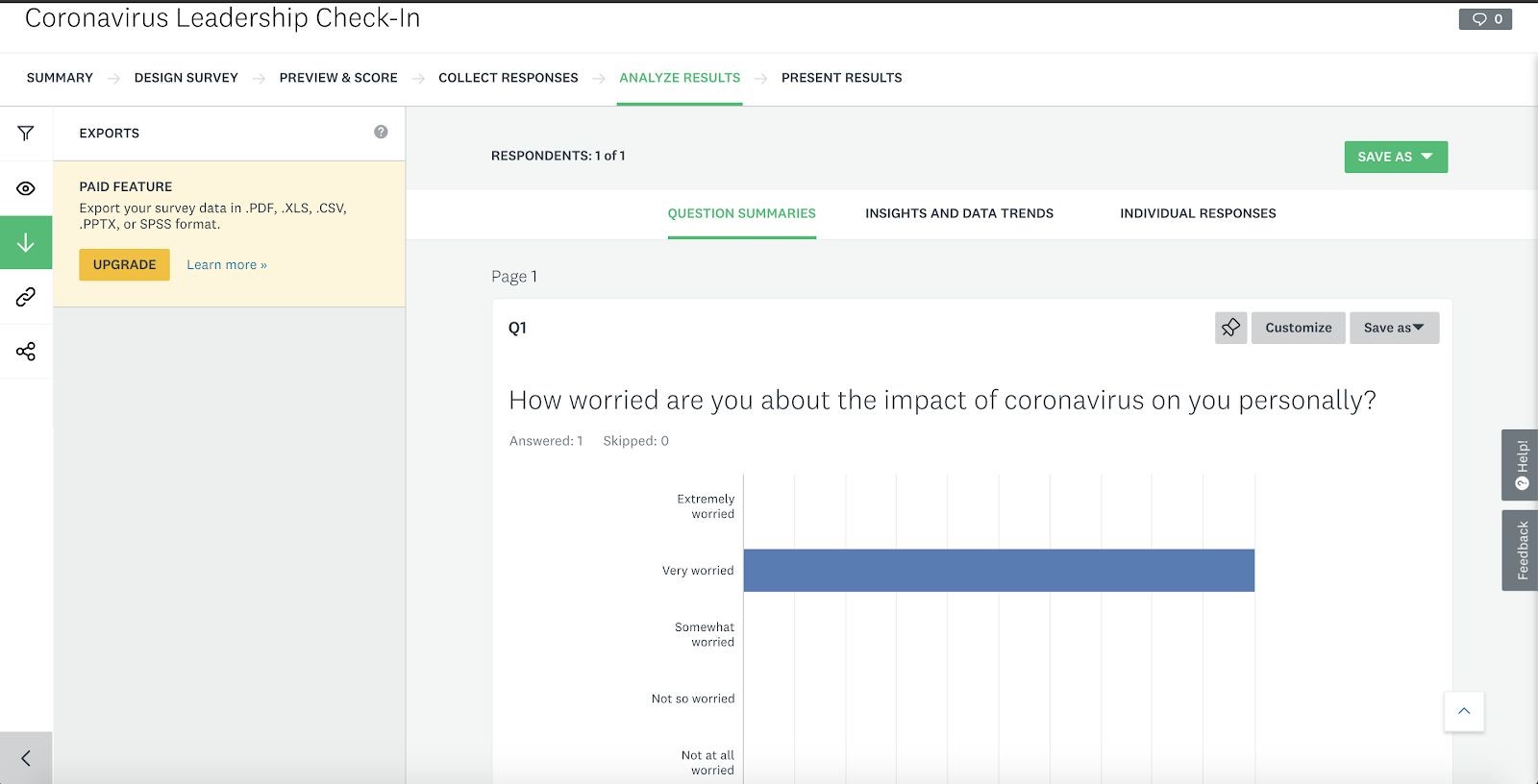
* Survey Monkey is an online survey application and makes it easy to take feedback from users. This webpage helps to create surveys with different options (i. using template or new surveys on their own).



Summary page showing the survey response and it's easy to analyze the results

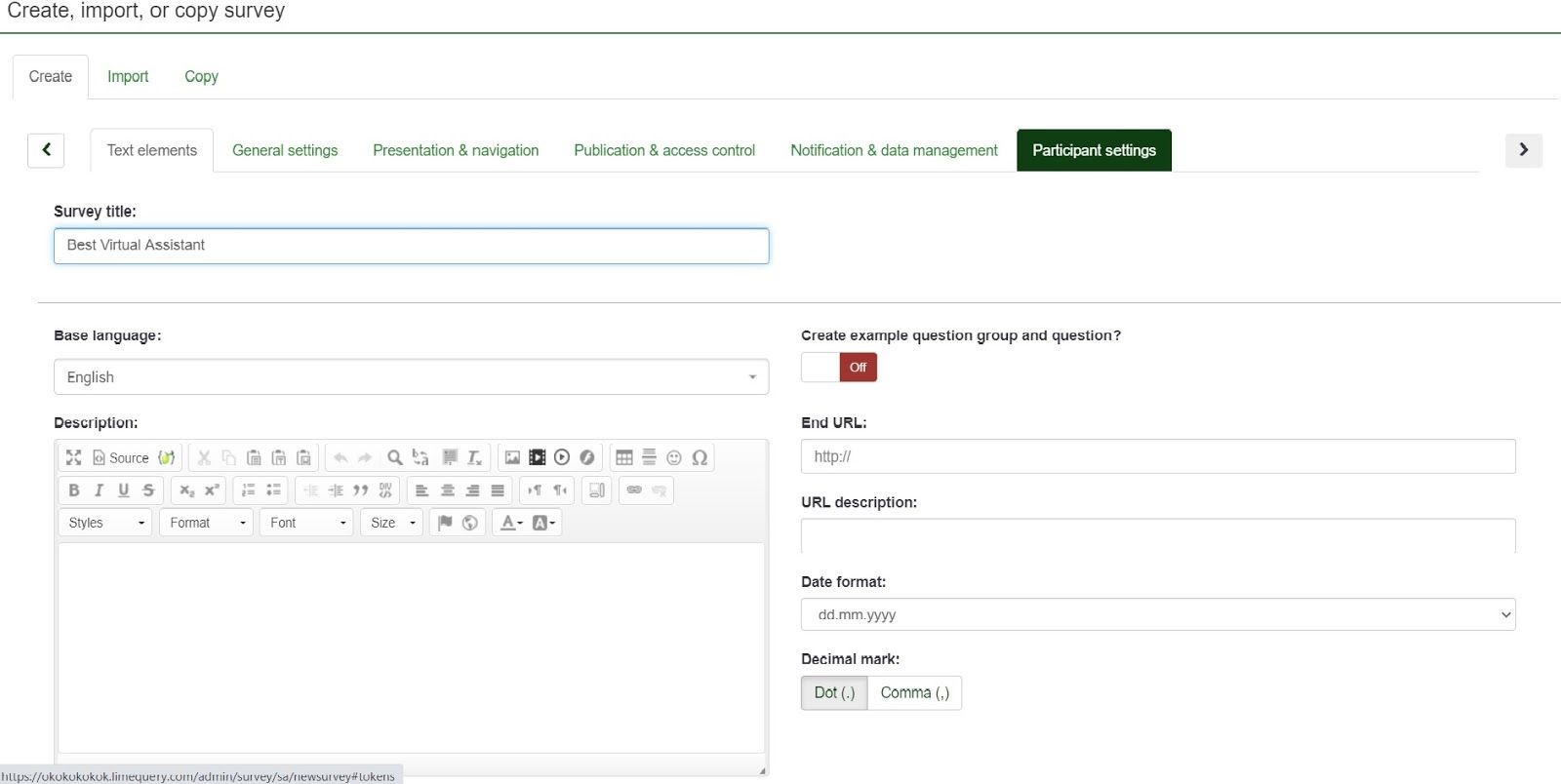


Summary Page shows the statistics and result for the response.

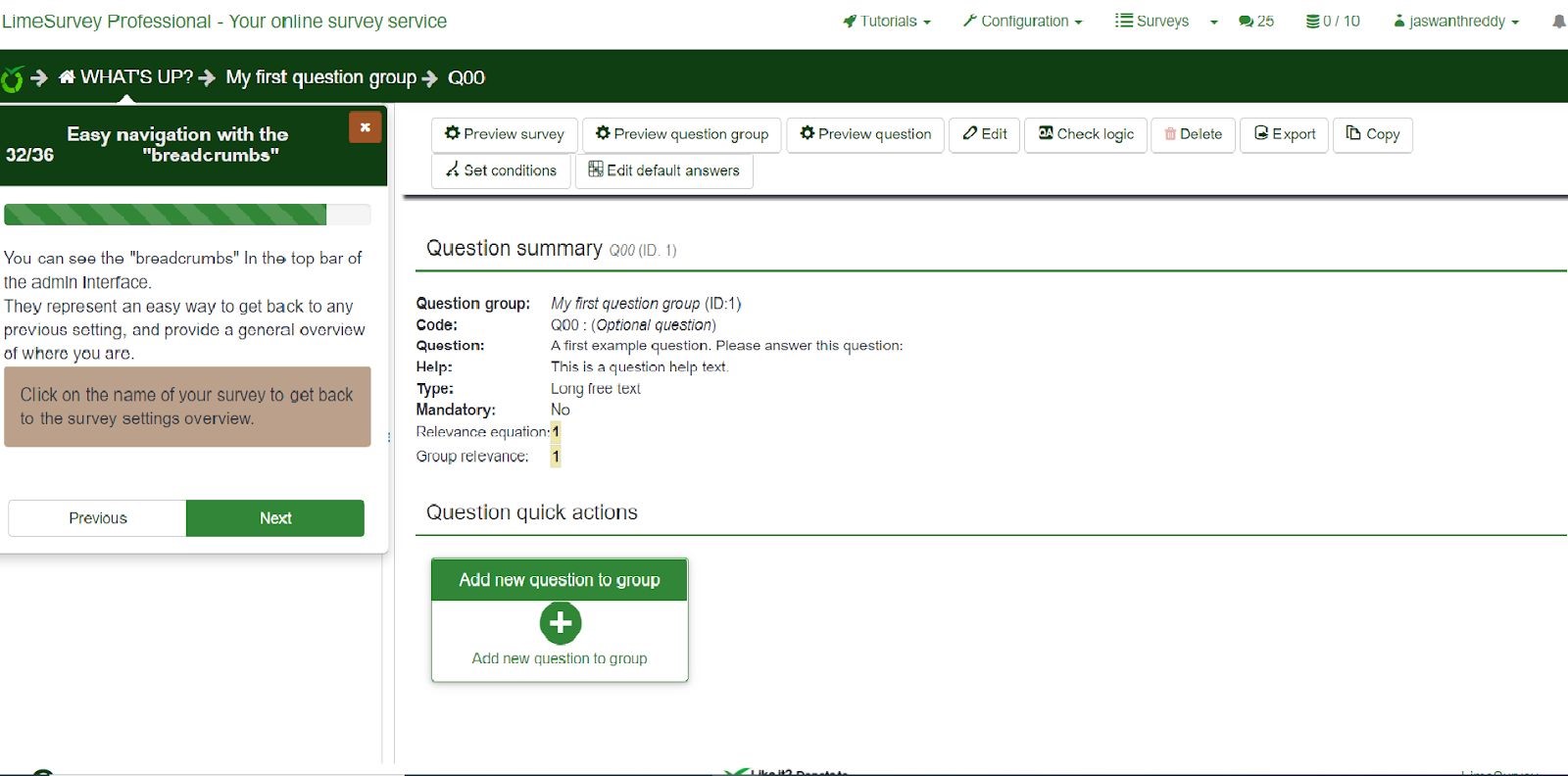


**LimeSurvey**

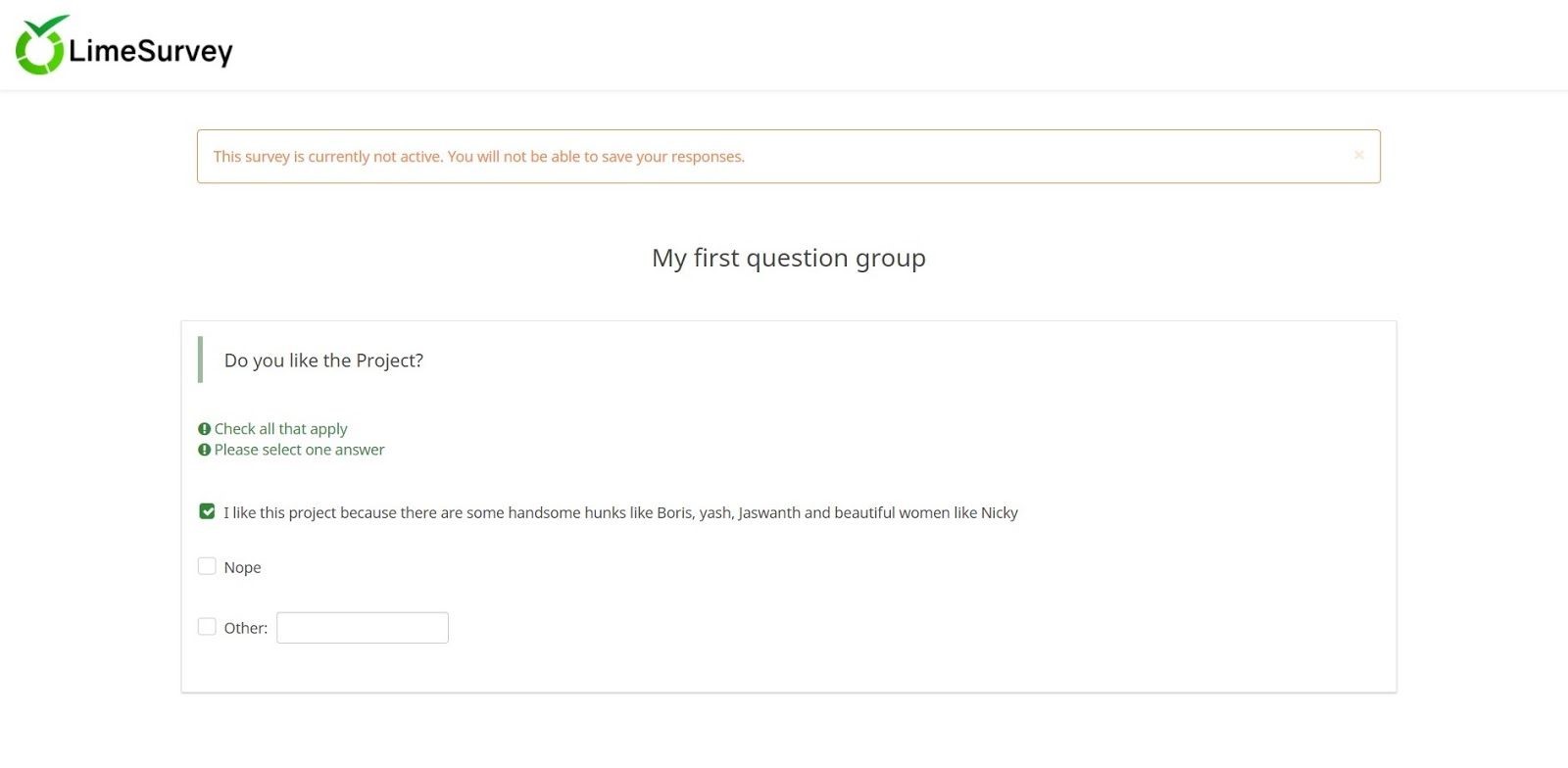
* LimeSurvey is a web based software that enables users using a web interface to​ develop and publish on-line surveys, collect responses, and create statistics. A similar GUI can be used to support users creating accounts and authenticating with the survey application. The information will be stored in our User and associated entities.



* Question summary can be easily previewed and changes can be done immediately.



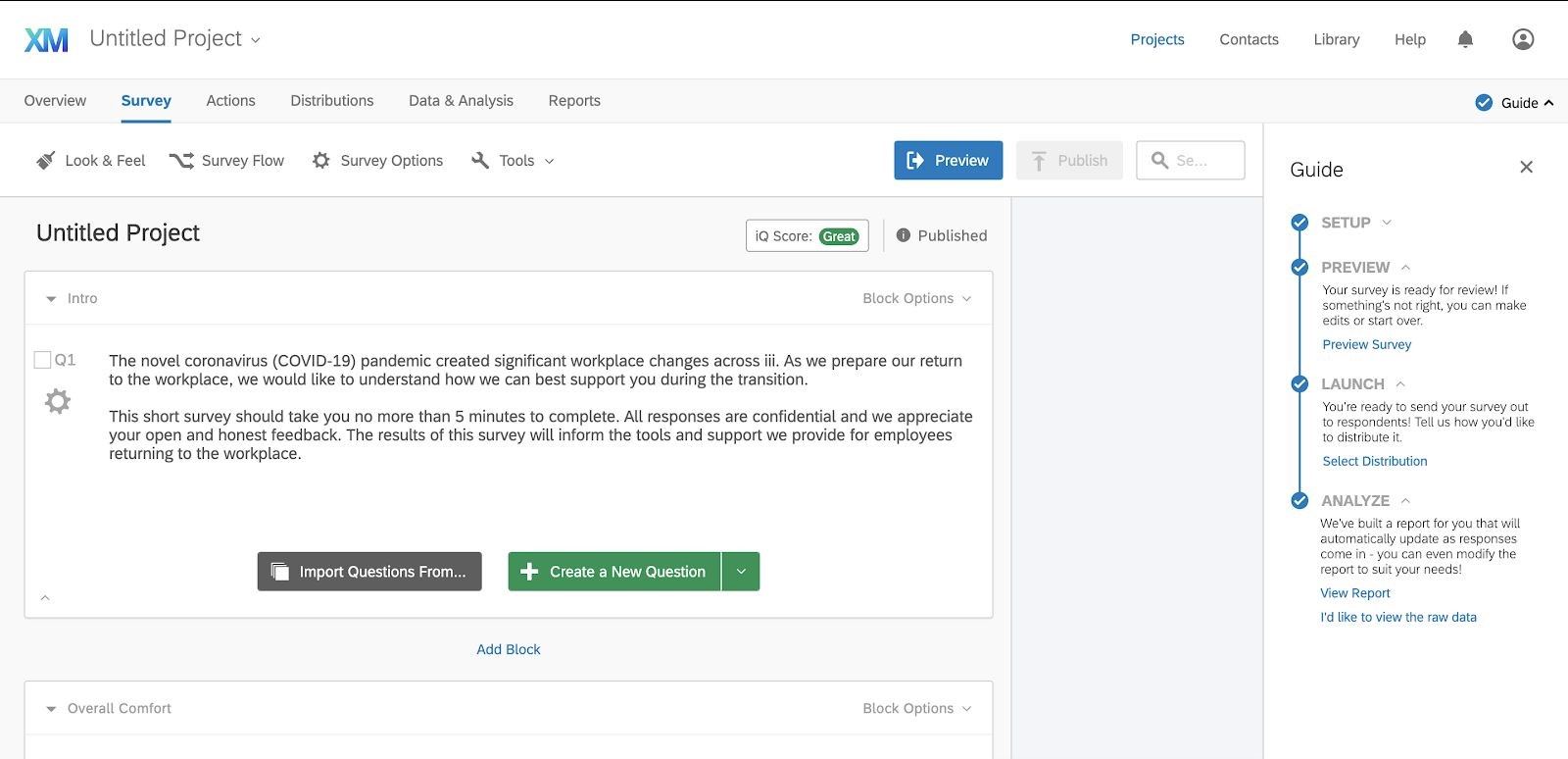
* Quicker access and simpler format for answering the questions.



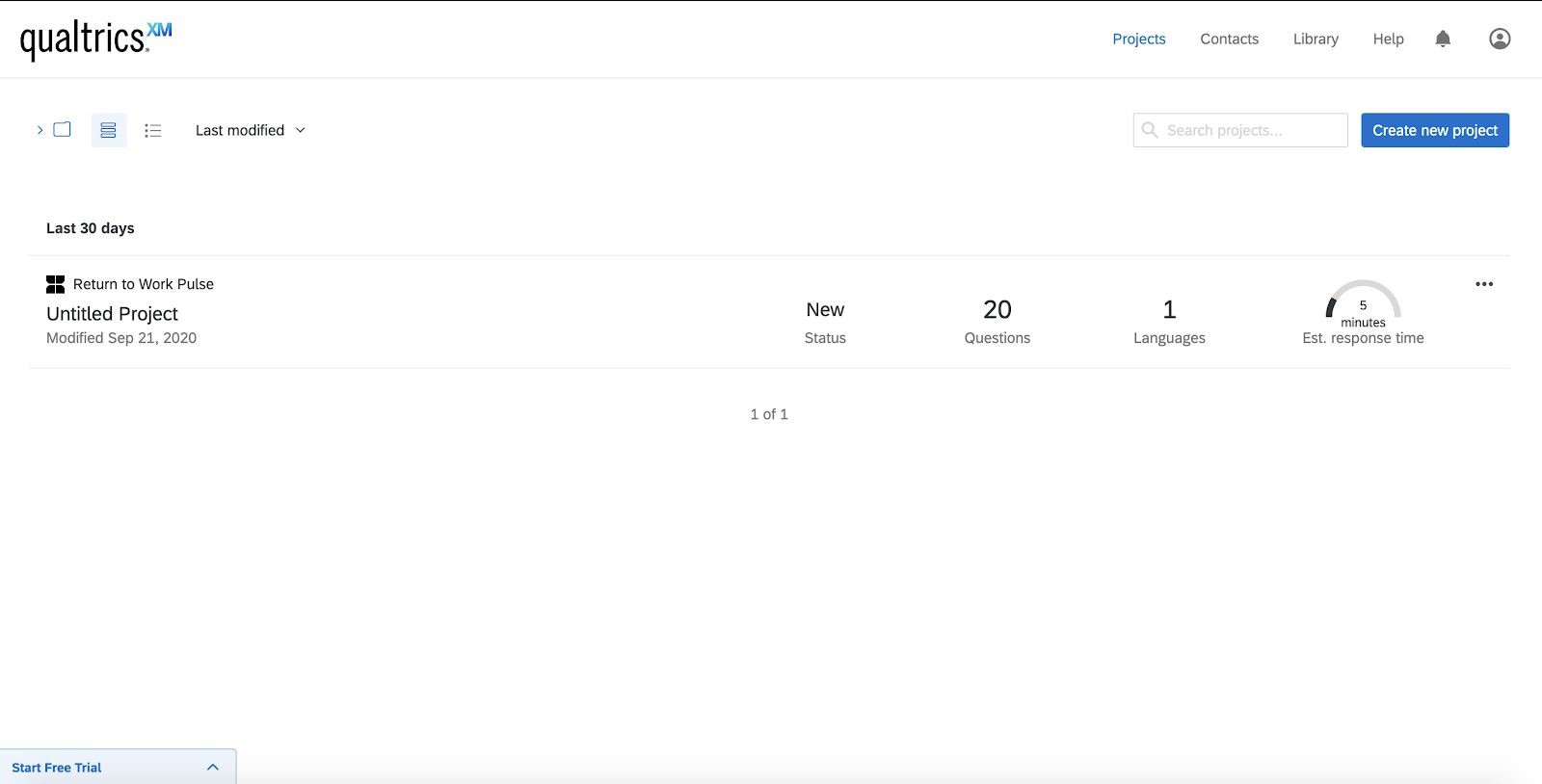
**Qualtrics:**

Qualtrics is an online survey tool that allows users to create surveys and analyze the survey reports.

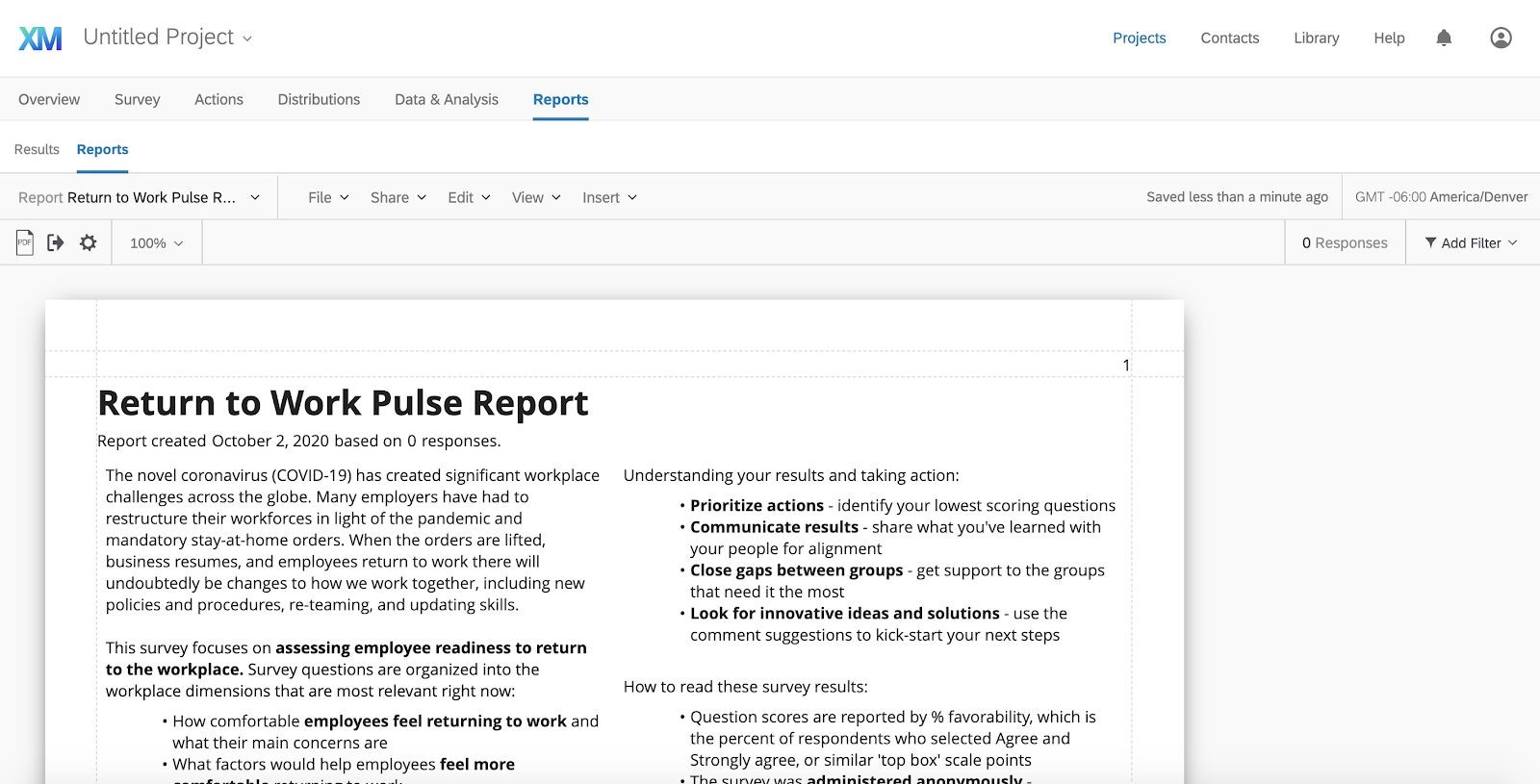
Survey page is for the survey creation and to preview the survey questions



Summary page shows the survey details with survey title, number of questions and response time.



Survey Report shows all the responses for the survey questions.



# 2. Pros and Cons of Analysed systems(SurveyMonkey, LimeSurvey & Qualtrics)

**SurveyMonkey**

**Advantages**​ :​

* Easy to use the interface.
* Surveys are created very easily.
* Easy to customize a survey.
* Seems to be purely Software-as-a-service (SaaS) - no need to manage your own server or worry about installing any software.
* Support for creating single and multiple surveys.
* Available for creating surveys by an individual or by a team(group of people can collaborate) as well as.
* Supports various platforms for sending surveys i.e through mail ,creating a web link and other social media platforms.
* Survey responses can be shared by creating a link.

**Disadvantages**:​

* It has a limitation with the integrations of applications.
* It needs a more interactive interface, its structure is somewhat boring.
* Survey supports few file formats.
* Some of the training modules are a bit lengthy and could be better served if broken into shorter topics.
* When waiting for results, since it is anonymous it doesn't tell you who has not completed the survey. If there was a way for it to constantly be in your inbox to prompt the person to complete it quickly, that'd be great.
* Offering more integrations with other sites and services would be helpful.
* Additional design functionalities would be beneficial - for internal design teams for instance to be able to upload various designs seamlessly.
* Offer better management of team accounts - making it easier to automatically share all surveys regardless of the individual account (that is under a group company account).
* Offer complex questions are a bit of a challenge to format in the software

**LimeSurvey**

**Advantages:**

* Surveys can be made more accurately. ● Free tier without a time limit.
* Multiple surveys can be created at one time.
* Condition questions can be given easily.

For example :

Q1) how old are you?

Q2 ) do you consume alcohol?

So here if the answer for the Q1 is >= 21 then the Q2 appears ,if not we can hide that question until the condition is met. And if the answer is <21 Q2 doesn't appear.

* We can do basically any order of questions or flow, re-organise how it sends out, when, when it closes, who can answer.It has many options to make the survey completely customised.
* we can create the forms and send them automatically by email, the data will be safe and we will be able to observe the results clearly to make correct decisions based on these.

**Disadvantages:**

* Difficult to use; one cannot use without computer knowledge.
* Free tier is limited to only 25 responses.
* Must pay to use LimeSurvey PRO to leverage hosting services, otherwise you need a server to install the software yourself.
* Creating surveys is time consuming.
* Conflicts requests during testing.
* Has no reporting or analytic capabilities.
* Complicated. It's easily an order of magnitude more complex than the likes of SurveyMonkey.
* The administrative part: getting a view of the questions for a certain period of time can be a bit tricky and exporting them to excel takes a lot of actions
* Only administrators can create and activate surveys. When we create an administrator, we can define the rights in such a way that he can create surveys and sees only his own surveys. But of course he must have the knowledge how to create and administer surveys
* A self-signup for the limesurvey backend is not provided. we'll have to write a self-signup routine on our own, however this is quite an easy task if we have access to the database.

**Qualtrics**

Advantages:

* Ease of use in creating surveys, data collection, data analysis, customizable question types, exporting data and survey distribution.
* User Friendly UX.
* The analytics and reporting that one can use from survey results is extensive, there are so many options.
* Compared to other survey products, Qualtrics offers more flexibility and options to create questions so we can build our surveys exactly like we want them.
* A passionate and friendly community of users always willing to help.
* Multiple item structures (e.g. multiple choice, likert, sliding scale, etc.)
* There are both personal and shared libraries for individual users and business teams.
* One can save results to a wide variety of formats (CSV, SPSS, and more) for further data management

**Disadvantages:**

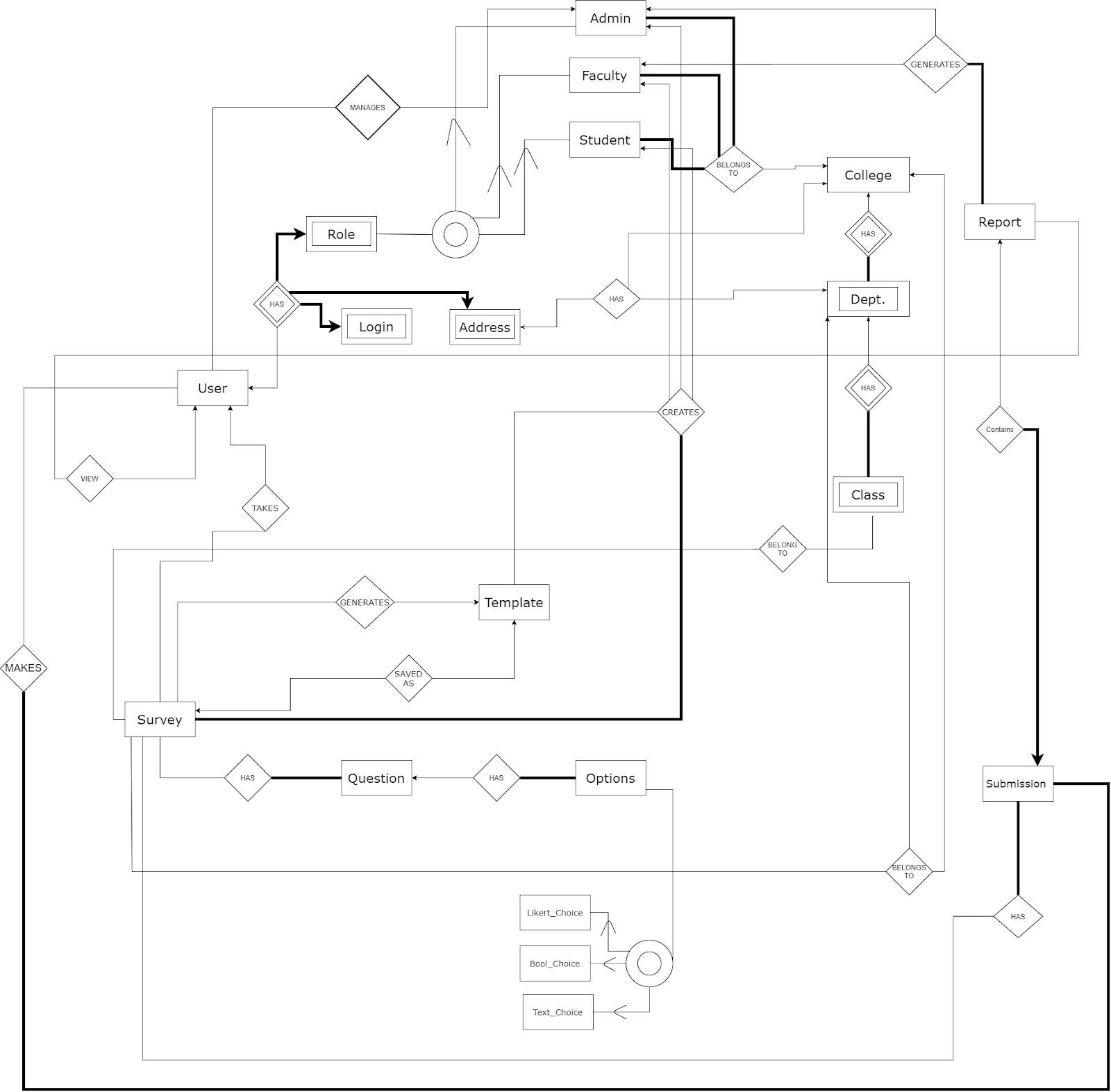
* Free trial available with a time limit of 30 days.
* Limitations on number of questions created on free trial of survey.
* Survey logic (need more options).
* It can be hard to interpret readability results.
* The mobile view (while editing from desktop) doesn't really look like the actual mobile view of a survey.
* The amount of options to build questions can be overwhelming if you just want to use the product to create simple surveys.
* Need better options for exporting the reporting and analytics into formats that can be shared easily.
* XM directory needs more flexibility in how you manage ,how often to contact panelists.
* XM directory could have more information on the contact level - When statuses were changed or added and the source.
* It should be easier to get a 360 view of an individual respondent.
* Surveys could be more organized.
* Ability to make reports (scales on graphs) more customizable for bipolar scales.
* When reverse coding items, the default settings keep overriding what one person tries to enter, which is time-consuming, and also difficulties in assigning the values that are needed.

# 3. Business Rules/Constraints

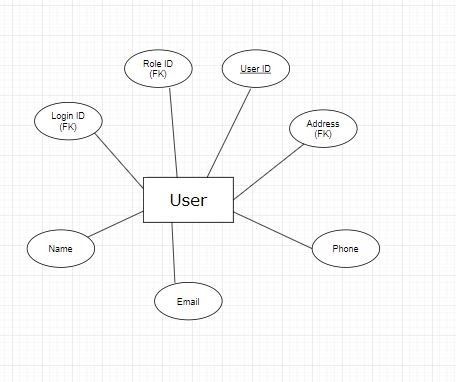
1. Allow a survey creator to associate more than one response type with a question but allow for limiting the response types to a specific group.
2. Respondents can only respond to the survey on getting an invitation link by creators.
3. Limitation for number of questions - creator can publish up to 10 questions for basic plan.
4. Responses to questions permitted are one of three types: likert scale, boolean (true or false), and free text. Likert scale choices are: Strongly agree, agree, neutral, disagree, strongly disagree.
5. Users can create custom options for response to a question.
6. No limitation on the number of questions for creators on premium plans for a creator.
7. Respondents cannot view responses of other participants/respondents.
8. One user can create many surveys.
9. Admin can manage many users.
10. Admin can generate as many reports.
11. Respondents can keep themselves as anonymous while giving response to a survey.
12. Creators can schedule a survey beforehand.
13. Many surveys belong to one college.
14. Many surveys belong to one department.
15. Creators can set up a recurring alert for sending the surveys with a fixed time interval.
16. One survey can be of multiple types/categories like template, previous template used and a new survey.
17. Participants who do not respond within a defined time range should be sent a reminder email by the system. They can still submit an anonymous response.
18. Creators can allow or prohibit participants from viewing a basic report.
19. Creators can generate detailed reports of one or more surveys.
20. Survey must have a start date but does not need to have an expiry date.

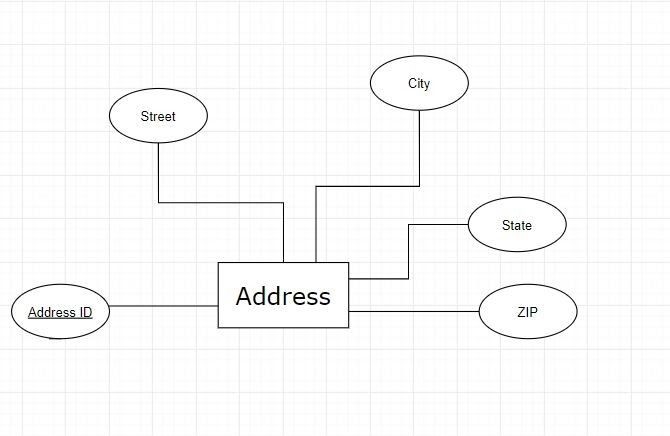
**4. Main E-R Diagram**

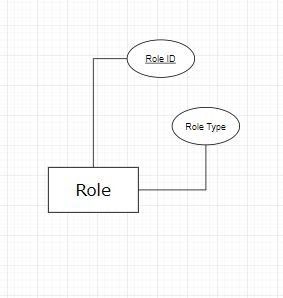
Also attached as a PNG in the submission ZIP for readability.

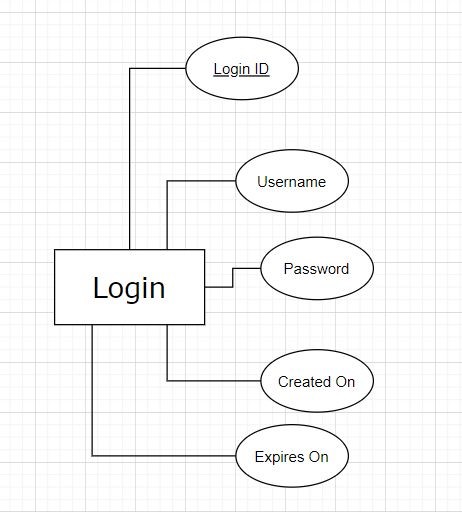


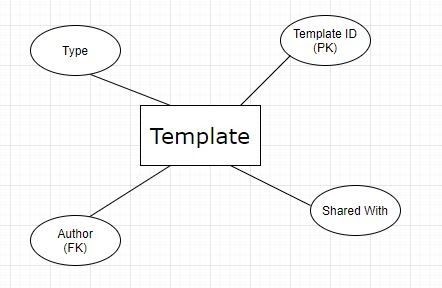
# 5. E-R Design of Each Entity

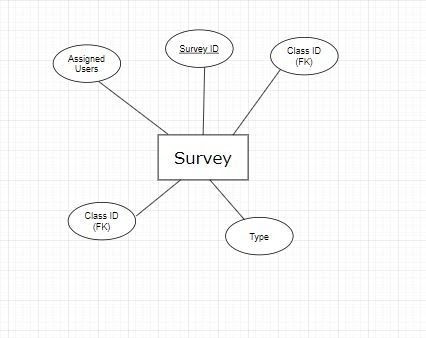


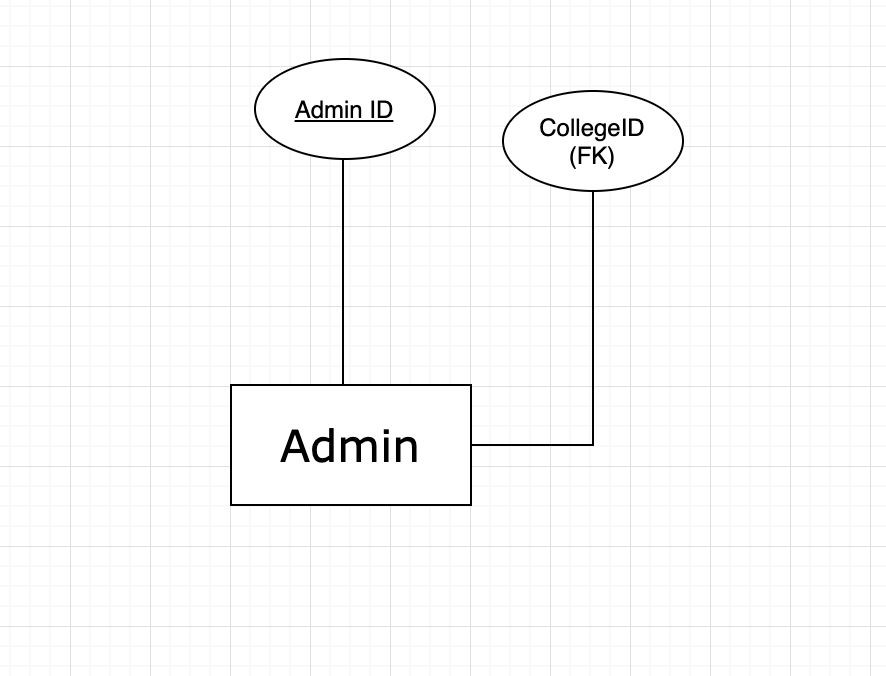


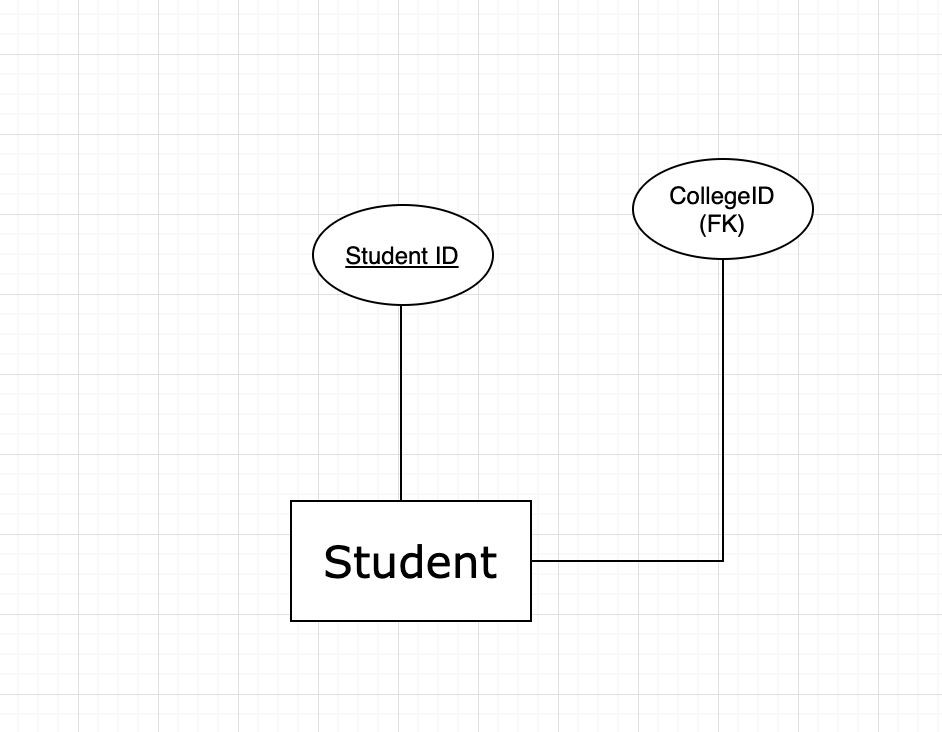


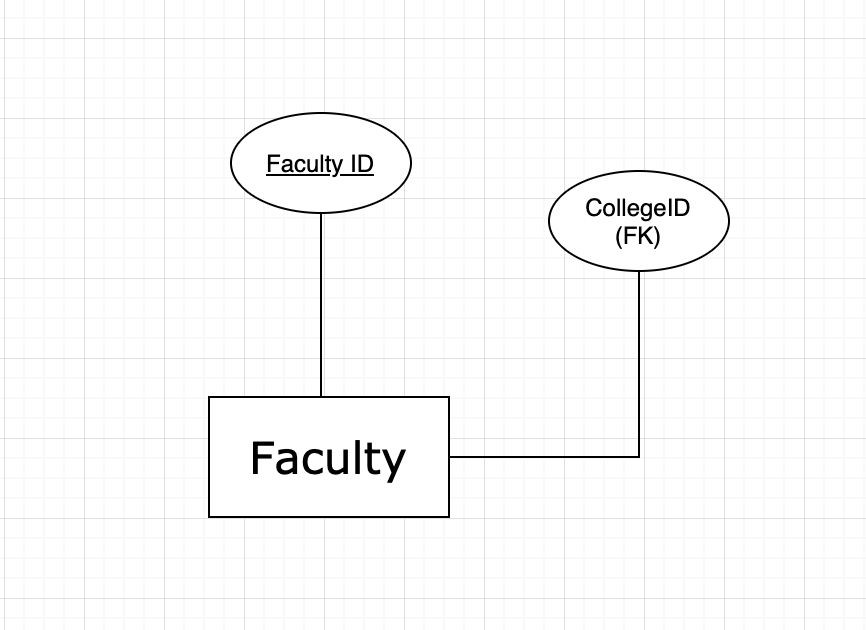


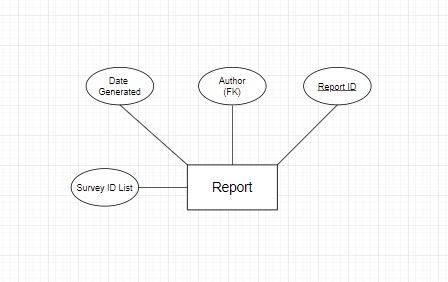


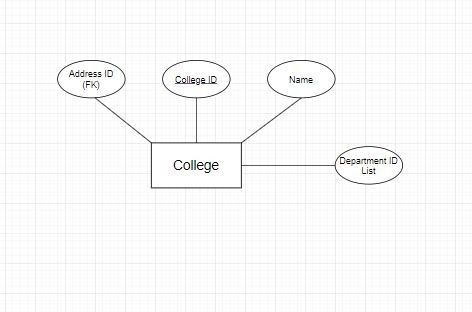


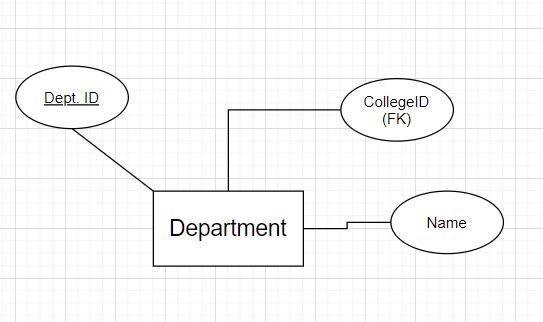


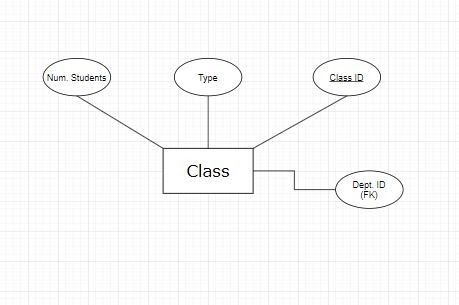


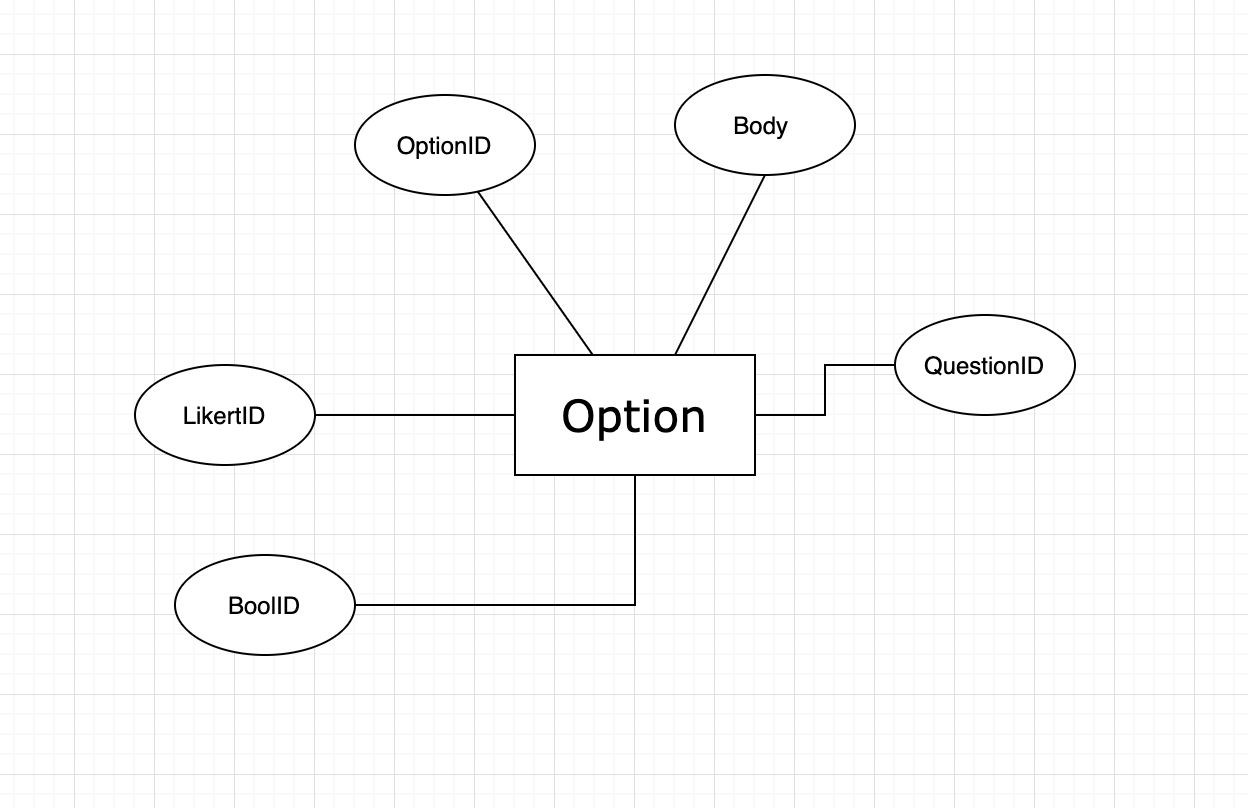


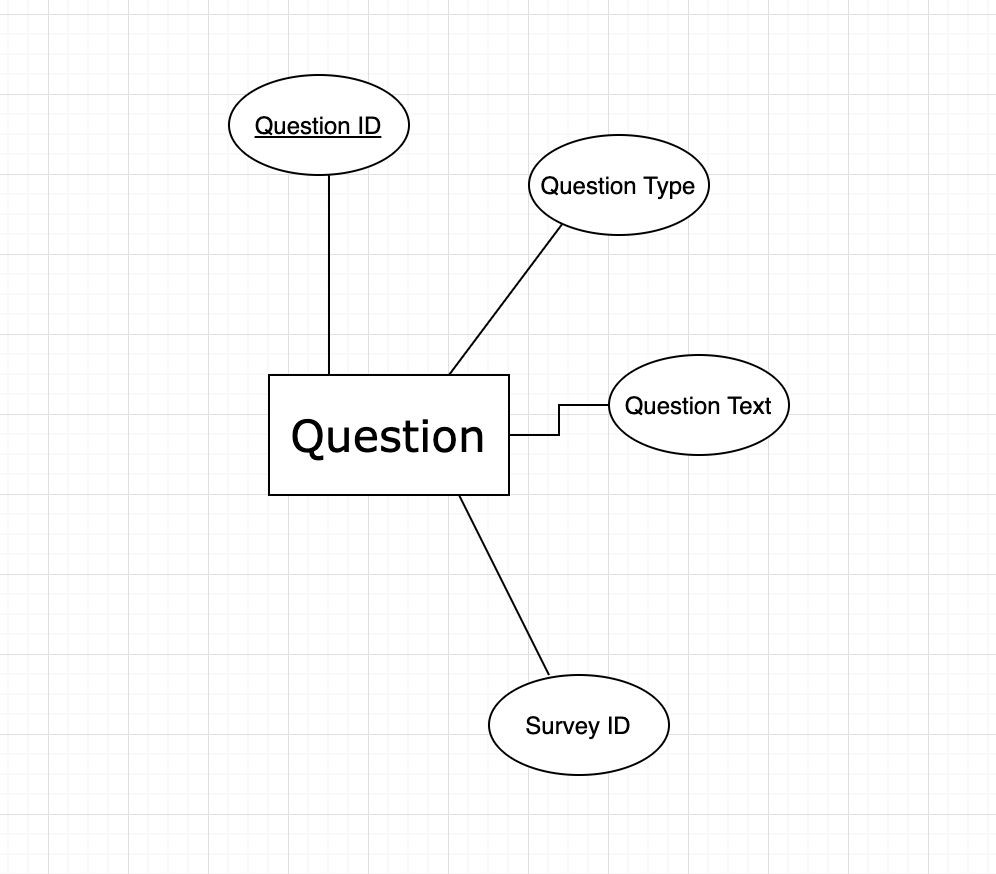


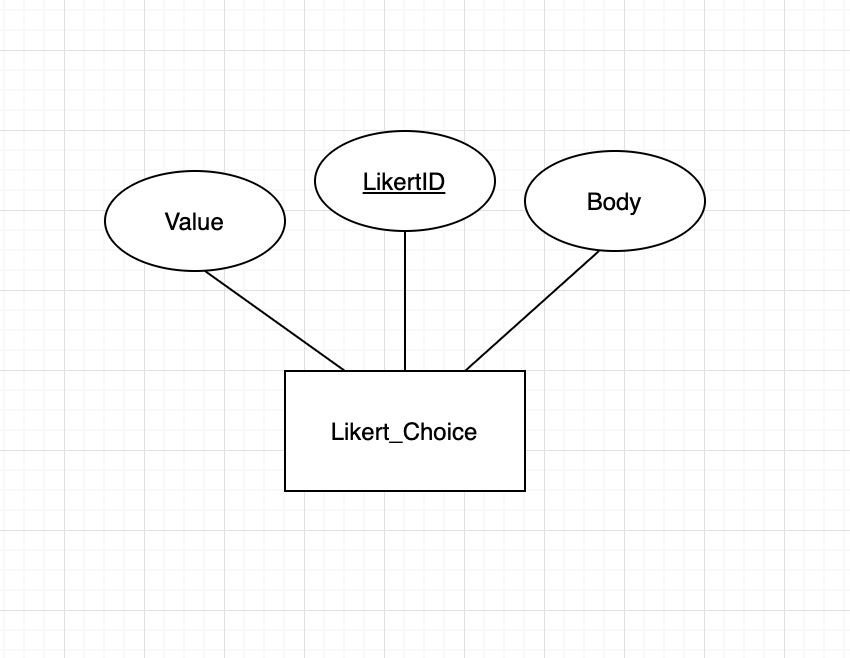


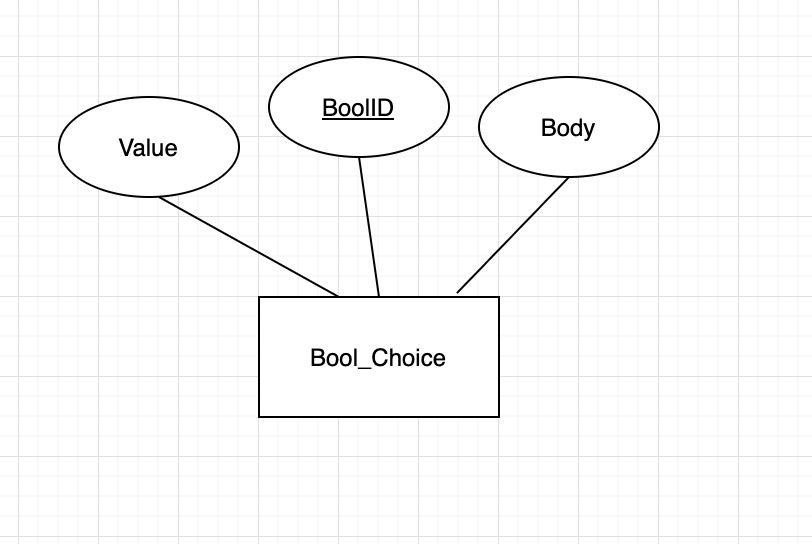


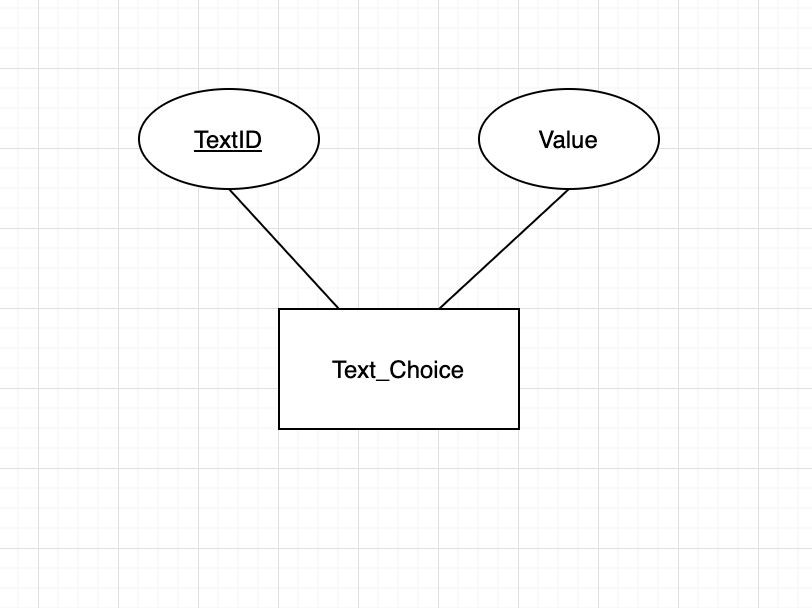


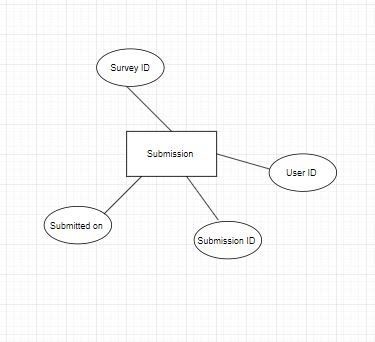






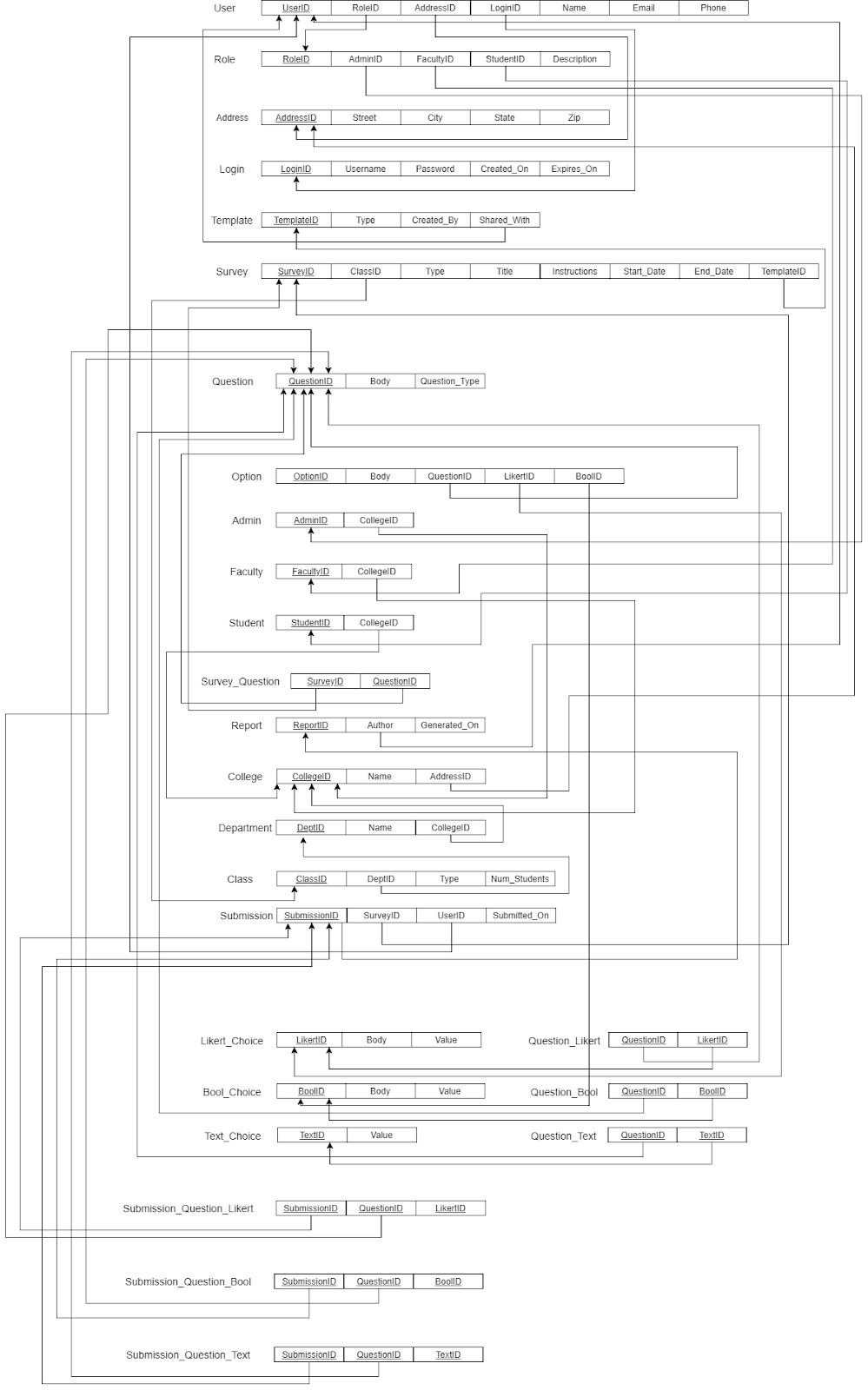






# 6. Relational Schema Diagram

Export of the Relational Schema Diagram is included in the submission ZIP for readability.



## 6.1 Relational Schema Definition and Domain of all attributes.

1. User(​**UserID**​, ​*RoleID*​, ​*AddressID*​, ​*LoginID*​, Name, Email, Phone)
2. Role(​**RoleID**​, AdminID, ​*FacultyID*​, ​*StudentID*​, Description)
3. Address(​**AddressID**​, Street, City, State, ZIP)
4. Login(​**LoginID**, Username, Password, Created\_On, Expires\_On)​
5. Template(​**TemplateID**​, Type, ​*Created\_By*​, Shared\_With)
6. Survey(​**SurveyID**, Title, ​ *ClassID*​ ​, Type, Title,Instructions, Start\_Date, End\_Date, *TemplateID*​)
7. Question(​**QuestionID**​, Body, Question\_Type)
8. Option(​**OptionID**​, Body, QuestionID,LikertID,BoolID)
9. Admin(​**AdminID,** ​CollegeID)

10)Faculty(​**FacultyID,** ​CollegeID)

11)Student(​**StudentID,**​CollegeID)

12)Survey\_Question(​**SurveyID**​, **QuestionID**​ ​) 13)Report(​**ReportID**, ​ *Author*​ ​, Generated\_On)

14)College(​**CollegeID**​, Name, ​*AddressID*​)

15)Department(**DepartmentID**​ , Name, ​ *CollegeID*​ ​)

16)Class(​**ClassID**​, ​*DeptID*​, Type, Num\_Students)

17)Submission(​**SubmissionID,**​*surveyID,UserID,*​Submitted\_On)

18)Likert\_Choice(​**LikertID**, Body, Value)​

19)Bool\_Choice(​**BoolID**​, Body, Value)

20)Text\_Choice(​**TextID**​,Value)

21)Question\_Likert(**QuestionID**​ , ​ ​**LikertID**)​

22)Question\_Bool(​**QuestionID**​, ​**BoolID**​)

23)Question\_Text(​**QuestionID**​, ​**TextID**​)

24)Submission\_Question\_Likert(​**SubmissionID**, ​ ​**QuestionID**​, LikertValue)

25)Submission\_Question\_Bool(​**SubmissionID**​, ​**QuestionID**​, BoolValue) 26)Submission\_Question\_Text(​**SubmissionID**​, ​**QuestionID**​, TextValue)

# 7. Domains of all attributes

1. **User Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| UserID | NUMBER(15) | Primary key. |
| LoginID | NUMBER(15) | Foreign key establishing relationship with login entity. |
| RoleID | NUMBER(15) | Foreign key for role, establishing relationship with role entity. |
| AddressID | NUMBER(15) | Foreign key establishing relationship with address entity. |
| Email | VARCHAR2(100) | User’s email address. |
| Phone | VARCHAR2(10) | 10 digit US based phone number |

1. **Role Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| RoleID | NUMBER(15) | Primary key. |
| AdminID | NUMBER(15) | Foreign key of admin. |
| FacultyID | NUMBER(15) | Foreign key of faculty. |
| StudentID | NUMBER(15) | Foreign key of students. |
| Description | VARCHAR2(20) | Will tell the type of the user. |

1. **Address Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| AddressID | NUMBER(15) | Primary key |
| Street | VARCHAR2(100) | Holds the street number and name |
| City | VARCHAR2(100) | City as string |
| State | VARCHAR2(2) | 2 letter abbreviated state code |
| ZIP | NUMBER(5) | 5 digit postal ZIP code |

1. **Login Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| LoginID | NUMBER(15) | Primary key |
| Username | VARCHAR2(100) | User’s chosen username. May decide to make this the user’s email. |
| Password | VARCHAR2(255) | User’s password. |
| Created\_On | DATE | When was the account created |
| Expires\_On | DATE | When does the account expire |

1. **Template Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| TemplateID | NUMBER(15) | Foreign key for survey. |
| Type | VARCHAR2(100) | What type of template will this be, to give the user a general idea. For example, a “Course Survey” or “General” type. |
| Created\_By | NUMBER(15) | Foreign key for UserID in User table. |
| Shared\_With | VARCHAR2(1000) | A string holding a comma separated list of UserIDs with whom the template has been shared. |

1. **Survey Types**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| SurveyID | NUMBER(15) | Primary key |
| ClassID | NUMBER(15) | Foreign key establishing relationship with a class if necessary (e.g., survey is for a course evaluation). |
| Type | VARCHAR2(255) | The type of survey being conducted. Very similar to the Template’s type field. Will describe whether the survey is general or for a course evaluation, for example. |
| Title | VARCHAR2(255) | Survey’s main heading |
| Instructions | VARCHAR(2000) | Instructions for the user |
|  |  | regarding how to complete the survey. This is important as the questions have to employ a particular phrasing if using a Likert or Boolean option for instance |
| Start\_Date | DATE | When does the survey  start |
| End\_Date | DATE | When does the survey close |
| TemplateID | NUMBER(15) | Foreign key establishing relationship with the template entity, provided the survey was generated using an existing template. |

1. **Questions Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| QuestionID | NUMBER(15) | Will contain the ID of the question. |
| Body | VARCHAR(255) | The text body of the question itself. |
| Question\_Type | VARCHAR2(7) | Can be one of three types: Likert, Boolean, or Text. |

1. **Option Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| OptionID | NUMBER(15) | Primary contain the OptionID |
| Body | VARCHAR2(200) | The text body of the question itself. Example: “Did you vote in 2016?” |
| Question ID | NUMBER(15) | This is a foreign key that will establish a relationship with the Question entity/table |

1. **Admin Type:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| AdminID | NUMBER(15) | Primary key to identify the admin |
| CollegeID | NUMBER(15) | This a foreign key that will establish a relationship with the College entity/table. |

1. **Faculty Type:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| FacultyID | NUMBER(15) | Primary key to identify the faculty |
| CollegeID | NUMBER(15) | This a foreign key that will establish a relationship with the College entity/table. |

1. **Student Type:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| StudentID | NUMBER(15) | Primary key to identify the student |
| CollegeID | NUMBER(15) | This a foreign key that will establish a relationship with the College entity/table. |

1. **Report Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| ReportID | NUMBER(15) | Primary key |
| Author | VARCHAR2(30) | Generally contains the details about the user. |
| Generated\_on | DATE | When was the report created. |

1. **College Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| CollegeID | NUMBER(15) | ID number might contain not more than 20 digits |
| Name | VARCHAR2(100) | Name of the college. Ex. - College of Arts and Sciences. |
| AddressID | NUMBER(15) | Foreign key establishing relation with the address entity, where the college’s address will be stored (if provided). |

1. **Department Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| DeptID | NUMBER(10) | ID number might contain not more than 10 digits |
| Name | VARCHAR2(30) | Name of the Department  Example: Computer  Science |
| CollegeID | NUMBER(20) | ID number contain not more than 20 digits |

1. **Class Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| ClassID | NUMBER(15) | Primary key for class identification |
| DeptID | NUMBER(15) | Foreign key establishing relationship with the department table, so we can tell to which department does the course belong |
| Type | VARCHAR2(100) | Type of class - graduate or undergraduate. |
| Num\_Students | VARCHAR2(100) | Total number of students in the class |

1. **Submission Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| SubmissionID | NUMBER(15) | Submission ID not more then 10 digits |
| SurveyID | NUMBER(15) | Survey ID not more then 10 digits |
| UserID | NUMBER(15) | UserID not more then 10  digits |
| SubmittedOn | DATE | When was the submission created |

1. **Likert\_Choice Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| LikertID | NUMBER(15) | Primary key |
| Body | VARCHAR2(17) | Will hold the static information an option contains, for display on the UI. For example, “Strongly Agree” as a string value. Only 17 characters are needed, that is the longest  count in “Strongly  Disagree” |
| Value | NUMBER(1) | Value can be 1 through 5 and corresponds to the ranking like so: 1. Strongly Agree   1. Agree 2. Neutral 3. Disagree 4. Strongly Disagree |

1. **Bool\_Choice Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| BoolID | NUMBER(15) | Primary key |
| Body | VARCHAR2(3) | Contains the string literals for Yes and No. Longest is yes at 3 characters |
| Value | CHAR(1) | Oracle RDBMS does not support a Boolean datatype. We create a table with a column of datatype CHAR(1) and  store either “Y” or “N” in that column to indicate TRUE or FALSE. |

1. **Text\_Choice Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| TextID | NUMBER(15) | Primary key |
| Body | VARCHAR2(50) | Contains the string literals of the user’s custom defined option for a  question |

**20.Question\_Likert Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| QuestionID | NUMBER(15) | Primary key of the question. |
| LikertID | NUMBER(15) | Primary key of the likert |
|  |  | option. So, suppose the question with ID #1 is of type Likert. And Likert has 5 options.  It will look like this:  1 - 1  1 - 2  1 - 3  1 - 4  1 - 5    It is a bit verbose, yes, but allows for object reuse. |

**21.Question\_Bool Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| QuestionID | NUMBER(15) | Primary key of the question |
| BoolID | NUMBER(15) | Primary key of the Bool options, allowing for reuse in many questions.  So, suppose the question with ID #5 is of type Bool.  And Bool has 2 options.  It will look like this:  5 - 1  5 - 2 |

**22.Question\_Text Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| QuestionID | NUMBER(15) | Primary key of the question |
| TextID | NUMBER(15) | Primary key of the Text options, allowing for reuse in many questions. Associates a question with a user’s custom defined text option. |

**23.Submission\_Question\_Likert Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| SubmissionID | NUMBER(15) | Primary key of the submission entity, creating implicit relationship with survey, question, option, and user |
| QuestionID | NUMBER(15) | Primary key of question entity |
| LikertValue | NUMBER(1) | Value of the user’s actual response. It is the aggregation of the three tables #21, #22, and #22 that comprise a user’s submission - we look in all 3 tables for a matching SubmissionID and then get the corresponding values for each question. Here we store the user’s response for a question of type Likert. |

**24.Submission\_Question\_Bool Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| SubmissionID | NUMBER(15) | Primary key of the submission entity, creating implicit relationship with survey, question, option, and user |
| QuestionID | NUMBER(15) | Primary key of question entity. It should be noted that there is no explicit entity for Text, we determine it if both the LikertID and the BoolID in the Option entity are null. |
| BoolValue | CHAR(1) | Value of the user’s actual response. It is the aggregation of the three tables #21, #22, and #22 that comprise a user’s submission - we look in all 3 tables for a matching SubmissionID and then get the corresponding values for each question. Here we store the user’s response for a question of type Bool. |

**25.Submission\_Question\_Text Types:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| SubmissionID | NUMBER(15) | Primary key of the submission entity, creating implicit relationship with survey, question, option, and user |
| QuestionID | NUMBER(5) | Primary key of question entity |
| TextID | VARCHAR2(100) | Value of the user’s actual response. It is the aggregation of the three tables #21, #22, and #22 that comprise a user’s submission - we look in all 3 tables for a matching  SubmissionID and then get the corresponding values for each question. Here we store the user’s response for a question of type free text. |

# 8. Design Decision and Domain of all Attributes (For E-R Diagram)

Many of the descriptions here are shortened for brevity, as they are already explained in detail in the previous section.

|  |  |
| --- | --- |
| Entity: User | |
| Primary Key Attribute: User ID | |
| Attributes | Description |
| RoleID | Foreign key establishing relationship with the role entity |
| AddressID | Foreign key establishing relationship with address entity |
| LoginID | Foreign key establishing relationship with login entity |
| First Name | User’s first name as a string. Required |
| Last Name | User’s last name as a string. Required |
| Email | User’s email address as a string. Required |
| Phone | User’s phone number as a string. Optional |
| * User can either be Admin,Faculty,Student or Participants. * If a user is Admin. Admin can manage all other user types, can generate a Survey,can take a survey and can view the Report. * If a user is a faculty, it can generate a survey and can also view the report. * If a user is a student,it can take a survey * If a user is simply a participant, can just take a survey. * User HAS Login [One-to-One] * User HAS Role [One-to-One] * User TAKES Survey [One-to-Many] * User VIEWS Report [One-to-Many] | |

|  |  |
| --- | --- |
| Entity: Role | |
| Primary Key Attribute: RoleID | |
| Attributes | Description |
| AdminID | Foreign key establishing relationship with admin entity. |
| FacultyID | Foreign key establishing relationship with faculty entity. |
| StudentID | Foreign key establishing relationship with student entity. |
| Description | It represents the description of type of role. |
| * Role consist of the type of the user. * User can have only one role at a time.Example: User can be an Admin or a faculty or a student or can be a participant. * Role is in **Total Participation**​ * Role can be one of Admin, Faculty or Student * If all foreign keys are null, then role is assumed to be general participant | |

|  |  |
| --- | --- |
| Entity: Address | |
| Primary Key Attribute: User ID | |
| Attributes | Description |
| AddressID | Primary key |
| Street | Street number and name as a string |
| City | Name as a string |
| State | 2 letter abbreviated state code |
| * User address is stored in this field * AddressID will denote whether the address is of a user,department,college. * Street will contain the name of the string its type will be VARCHAR2 ● City will have the name of the city of VARCHAR2 datatype. * State will store the 2 letter abbreviated state code * College, Department, or User can all have Addresses | |

|  |  |
| --- | --- |
| Entity: Login | |
| Primary Key Attribute: LoginID | |
| Attributes | Description |
| LoginID | Determine the user ID |
| Username | Give information about username |
| Password | Password details of the user |
| Created on | Date when user created his account. |
| Expires on | Date when the login will expire. |
| * LoginID will determine the ID of the user of the datatype NUMBER * Username field will include the username kept by the user datatype STRING ● Password field will have a password of character data type. * User HAS Login [One-to-One] * Login can expire and must be renewed on or before the expiry date | |

|  |  |
| --- | --- |
| Entity: Template | |
| Primary Key Attribute: User ID | |
| Attributes | Description |
| Template ID | Determine the template ID |
| Type | Determine the type. For example can be class eval survey or general. |
| Author | Foreign key |
| Shared with | Other users who have access to it. |
| * Template ID can Generate the Survey[One-to-Many] * Template has a Author[One-to-One] * Template can be shared with different user [One-to-Many]. | |

|  |  |
| --- | --- |
| Entity: Survey | |
| Primary Key Attribute: SurveyID | |
| Attributes | Description |
| ClassID | ClassID is a foreign key which establishes a relationship with the Class entity. |
| Type | Type indicates different types of survey like research survey,public survey, market survey etc. |
| Title | Title stores the survey title. |
| Instruction | Instruction provides detailed information about how the survey needs to be done and how to give response to different survey questions. |
| Start\_Date | Start\_Date indicates the date when the survey starts |
| End\_Date | End\_Date indicates the date when the survey completes. |
| TemplateID | TemplateID is a foreign key which establishes a relationship with the Template entity. |
| * Survey Has Question [Many to Many]. * One Template can generate many Survey [One-to-many] * User takes a survey [One-to-many] * Survey may belong to particular class [One-to-Many] * Survey may be saved as a Template [One-to-One] | |

|  |  |
| --- | --- |
| Entity: Question | |
| Primary Key Attribute: QuestionID | |
| Attributes | Description |
| Question ID | ID of the Question |
| Survey ID | ID of the Survey |
| Question Type | Determine the type of the Question |
| Question Text | Contains the text of text Itself. |
| * Question has ID. [One-to-One] * Question entity is in **Total participation**​ as there will be no question if the survey​ does not exist. * Question will have Survey ID[One-to-One] | |

|  |  |
| --- | --- |
| Entity: Option | |
| Primary Key Attribute: OptionID | |
| Attributes | Description |
| Body | Body is the survey question description |
| QuestionID | QuestionID is a foreign key which establishes a relationship with the Question entity. |
| LikertID | LikertID is a foreign key which establishes a relation to Likert\_Choice entity.. |
| BOOLID | BoolID is a foreign key which establishes a relation to Bool\_Choice entity. |
| ● Option can be of Likert\_Choice or Bool\_Choice, if neither then it is free text | |

|  |  |
| --- | --- |
| Entity: Likert\_Choice |  |
| Primary Key Attribute: LikertID |  |
| Attributes | Description |
| LikertID | Primary key |
| Body | Text body of the Likert option as a predefined value |
| Value | Numeric representation of the Likert option |

|  |  |
| --- | --- |
| Entity: Bool\_Choice |  |
| Primary Key Attribute: LikertID |  |
| Attributes | Description |
| BoolID | Primary key |
| Body | Text body of the Likert option as a predefined value |
| Value | Numeric representation of the Likert option |

|  |  |
| --- | --- |
| Entity: Text\_Choice | |
| Primary Key Attribute: TextID | |
| Attributes | Description |
| Body | Text body of the user defined option |
| ● The intent of this table is to enable users to create their own types of options in a custom way | |

|  |  |
| --- | --- |
| Entity: Admin | |
| Primary Key Attribute: AdminID | |
| Attributes | Description |
| CollegeID | CollegeID is a foreign key which connects to the primary key CollegeID in College entity. A college may have many departments. |
| * Admin ​**Manages**​ User.[One to Many]. * All Admin is associated with a user (Total Participation) but not all User associated with Admin. * Admin **Generates**​ Report. [One to Many].​ ● Admin ​**Belongs**​ to College.[Many to One] ● Admin ​**Creates**​ Survey. [One to Many]. | |

|  |  |
| --- | --- |
| Entity: Faculty | |
| Primary Key Attribute: FacultyID | |
| Attributes | Description |
| CollegeID | CollegeID is a foreign key which connects to the primary key CollegeID in College entity. A college may have many departments. |
| * Faculty ​**Creates**​ Survey.[One to Many] * Faculty ​**Belongs** ​to College.[Many to One] * Faculty ​**Generates**​ Report.[One to Many] | |

|  |  |  |
| --- | --- | --- |
| Entity: Student | |  |
| Primary Key Attribute: StudentID | |  |
| Attributes | | Description |
| CollegeID | | CollegeID is a foreign key which connects to the primary key CollegeID in College entity. A college may have many departments. |
| ●  ● | Student ​**Creates**​ Survey.[One to Many].  Student ​**Belongs**​ to College.[Many to One]. | |

|  |  |
| --- | --- |
| Entity: Report | |
| Primary Key Attribute: ReportID | |
| Attributes | Description |
| Author | Author is a foreign key which establishes a relationship with the user. |
| Date Generated\_On | Generated\_on depicts the date and establishes a connection with the user |
| * Report ​**Has** ​Submissions.[One to Many] * Submissions will contain user’s answers to survey questions * An Admin ​**Generates** ​Report [One-to-Many] * A Faculty ​**Generates** ​Report [One-to-Many] | |

|  |  |
| --- | --- |
| Entity: College | |
| Primary Key Attribute: CollegeID | |
| Attributes | Description |
| Name | Name is an attribute which presents the name of the college. |
| AddressID | AddressID is a foreign key which connects to the primary key AddressID in College entity. Address is the location of the college. |
| * College ​**Has** ​departments.[one to Many].   Here Department is a weak entity it is a ​**Total Participation.** ​Every department is associated with a college.     * College ​**Has**​ Address.[one to one]. * College ​**Belong**​ to Survey.[one to many] | |

|  |  |
| --- | --- |
| Entity: Department |  |
| Primary Key Attribute:DeptID |  |
| Attributes | Description |
| Name | Name is an attribute which represents different types of departments such as computer science ,mechanical, electrical and etc. |
| CollegeID | CollegeID is a foreign key which connects to the primary key CollegeID in College entity. A college may have many departments. |

|  |  |
| --- | --- |
| Entity: Class | |
| Primary Key Attribute: ClassID | |
| Attributes | Description |
| DeptID | DeptID is a foriegn key connected to the primary key DeptID in the entity called department. A department may have many classes. |
| Type | Type is an attribute which may contain different types of classes such as graduate and undergraduate or even high school. |
| Num\_Students | Num\_students is an attribute which indicates the total number of students in a class. |
| * Class ​**Has**​ students.[one to many] * Here Class is a weak entity it is a ​**Total Participation.** ​Every Class is associated with the department. * Class ​**Has** ​Types[one to many]. | |

|  |  |
| --- | --- |
| Entity: Submission | |
| Primary Key Attribute: SubmissionID | |
| Attributes | Description |
| SurveyID | Foreign key establishing relationship with survey entity, so we know which survey the submission is for. |
| UserID | Foreign key establishing relationship with user entity, so we know who made the submission |
| Submitted\_On | Date on which the submission was made |
| * User can submit survey response [many-to-many] * Survey has a submission [many-to-many] * Report contains a submission [many-to-many] * Submission entity is essentially a lookup table, allowing us to create a join query and get all the user’s responses. The relevant tables that contain the user’s responses to the survey question are outlined in section 8.1 question 1 about resolving n:m relationships. * A submission must be associated with a user and a survey | |

## 8.1 Description for mapping E-R diagram to Relational Schema Diagram.

1. How you handled n:m relationships?

* + Survey\_Question allows us to decompose the many-to-many relationship between survey and question. We can also reuse questions.
  + Question\_Likert: associates the likert choices with a question and allows for reuse among many questions. So questions 1 and 2 can both leverage these as options a user can choose.
  + Question\_BoolID: associates the boolean choices with a question and allows for reuse among many questions. So questions 1 and 2 can both leverage these as options a user can choose.
  + Question\_TextID: associates the custom choices with a question and allows for reuse among many questions. So questions 1 and 2 can both leverage these as options a user can choose.
  + User submission of a survey:

○ Submission\_Question\_Likert: This table will store user responses from a submitted survey if the question is of type Likert. It will contain the user’s selection.

○ Submission\_Question\_Bool: This table will store user responses from a submitted survey if the question is of type Bool. It will contain the user’s selection.

○ Submission\_Question\_Text: This table will store user responses from a submitted survey if the question is of type text (free text) or of type text as in a survey creator’s custom options. An example would be “What chocolate do you like?” and then the options can vary to any type of defined chocolate. The user’s selection can be found here.

1. How you handled ternary relationships?

We do not have ternary relationships in our project.

1. How you handled recursive relationships?

We do not have recursive relationships in our project.

1. How do you map multi valued attributes to the Relational Model?

We do not have multi valued attributes in our model.

1. How you handled inheritance?

We do not have inheritance in our project. One alternative option to handling custom responses using inheritance may be to add a Custom entity, inheriting from Option entity. It would contain OptionID, QuestionID, and Value. User can then enter custom options, for example in the question “Which chocolate do you like?” the user can specify whatever types of chocolate he/or she wants to ask about.

1. Informally state any constraints that you can represent in ER design, but cannot be mapped to relational design.

● An administrator manages users. This can be mapped on the ER diagram but not on the relational design because There does not need to be an explicit relationship via private-foreign key relation between the two. It would be at the application layer, where an admin can have access to make changes a standard user cannot.

# 9. Problems faced during the design of E-R Diagram and its Mapping

* For E-R diagrams, defining Participation and Cardinality was a time consuming task. ● We also focused on having a robust model that can support multiple answer types. We wanted to provide users with predefined choices but also allow them to define their own options.
* For Relational Schema Diagram, dealing with three subclasses of Client Entity was challenging as it includes the concepts of overlapping and disjoint.
* Establishing the data types for the entity attributes was challenging. The sample report indicated String, Boolean, and Int but Oracle DB does not have those types. We looked at Oracle documentation and used the actual data types that are supported.
* Decomposing many-to-many relationships with additional tables created some additional challenges because of the ambiguity of which tables and which primary keys needed to be included.

# 10. Not Included

The following are the few things we could not include in our E-R diagram.

* How the users creates an account
* How the user is sent an email notification that they are invited to participate in a survey
* How the user is sent a reminder email if the survey is about to expire and they have not yet submitted a response

# 11. Group Log

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Time | Participants | Activity |
| 9/12/2020 | 11:00  AM | All members | First project meeting. Discussed overall assignment. Assigned initial work items for document creation and analysis of the two survey tools. Boris to create a brainstorm document. |
| 9/12/2020 | - | Yash Patel | Updated both documents - ER entities brainstorm and comparative analysis. |

|  |  |  |  |
| --- | --- | --- | --- |
| 9/15/2020 | - | Boris | Updated both documents - ER entities brainstorm and comparative analysis. |
| 9/15/2020 | 6:40  PM | All members | Second project meeting. Went over the list of entities and attributes. Discussed pros and cons of SurveyMonkey and LimeSurvey. Updated docs and decided on a virtual daily stand up conducted via Teams messages. |
| 9/15/2020 | - | Nicky | Updated Comparative Analysis document. Updated business constraints. |
| 9/17/2020 | - | Boris | Created Draw.io ER diagram document and shared with team. Created a SurveyMonkey account and utilized the software. Updated ER and Comparative Analysis documents. |
| 9/17/2020 | - | Yash | Working on Business Constraint |
| 9/20/2020 | - | Yash | Worked on ER diagram, placing entities and attributes |
| 9/21/2020 | - | Nicky | Worked on analyzing Qualtrics Survey tool |
| 9/21/2020 | - | Boris | Updated ER diagram, focusing on relationships and attributes. Scheduled Teams meeting for 9/22 at 6:20 PM. |
| 9/21/2020 | 6:20  PM | All members | Third team meeting, discussed ER diagram and comparative analysis of the three tools. Assigned deliverables and split team in two. Boris and Yash on ER diagram, Nicky and Jaswanth on comparative analysis. |
| 9/22/2020 | 6:45  PM | Boris, Yash,  Nicky | Fourth team meeting, went over ER diagram in detail and added detail. Discussed report, compiled questions for instructor, and agreed on a timetable for delivery of ER diagram COB 9/24, after which we will start on the report and add requirements with justification for design decisions. |
| 9/25/2020 | - | Yash | Worked on report |
| 9/25/2020 | 7:45  PM | All members | Fifth team meeting, went over the ER diagram, decided to remove all attributes to focus on clear relationships. Began work on a relational model and created an outline of the report. Divided up E-R design of each entity between all members. Boris to finalize ER diagram and begin on relational model. Yash to update the report with details. | |
| 9/26/2020 | - | Yash | Worked on report, ER Diagram, ER relation diagram | |
| 9/28/20202 | 6:30  PM | Yash, Boris | Discussed ER diagram, disjoint sets, inheritance, and simplifying the diagram so it is readable and adheres to convention in class notes. | |
| 9/28/2020 | - | Boris | Updated ER diagram with weak entities, multiplicity, and updated individual entity diagrams. | |
| 9/29/2020 | 6:20  PM | Nicky, Yash,  Boris | Went through the ER diagram and finalized items and design. Went through Relational Schema Diagram and began connecting entities according to their PK-FK relationships. | |
| 9/30/2020 | 9:00  PM | All members | 6th team meeting. Went over relational schema and decomposing many-to-many relationships. Small blocker on handling user response submissions for a survey, will need to work on decomposing relationships. Assigned items to each member. | |
| 10/1/2020 | 9:20  PM | All members | 7th team meeting. Reviewed all diagrams and worked on the project report across all sections. All members working on sections individually throughout the day. | |
| 10/2/2020 | 6:00  PM | All members | 8th team meeting. Work session to finalize diagrams and report and package the project for submission. All members working on sections individually throughout the day. | |

LAST PAGE - NOTHING FOLLOWS