CPSC 304 Project Cover Page

Milestone #: 2

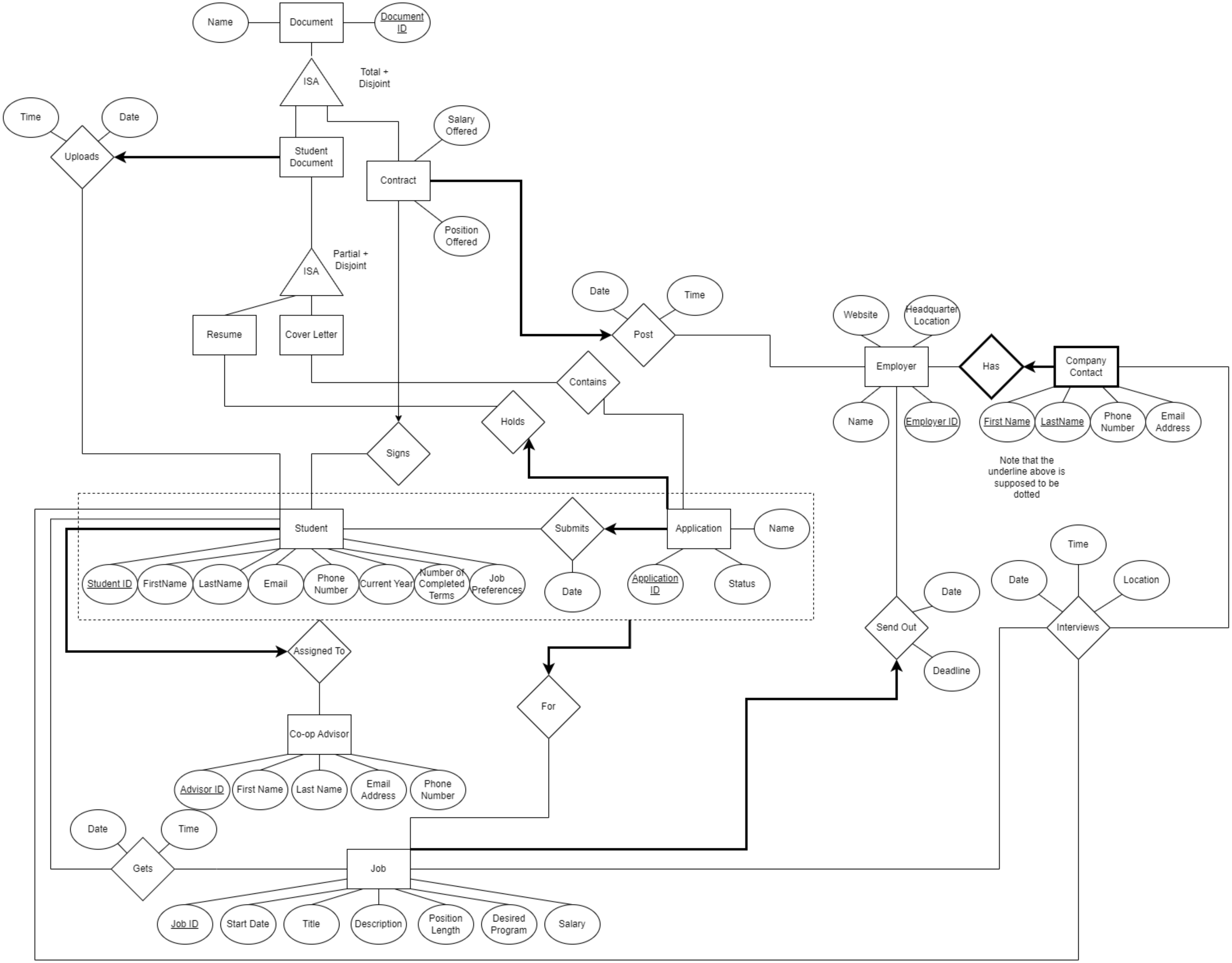
Date: July 28th, 2023

Group Number: 36

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Cole Rowell	52077435	chr2002	colerowell11@gmail.com
Nicholas Fong	78575271	nfong01	nicholasfong1120@gmail.com
Anikait Kapur	83243055	r0u2d	anikaitkapur@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia



Note that the underline above is supposed to be dotted

3. **Relational Model**

Co-opAdvisor (AdvisorID, FirstName, LastName, EmailAddress, PhoneNumber)
PK: {Advisor ID}
CKs: {Advisor ID}, {FirstName, LastName, EmailAddress, PhoneNumber}

Student (Student ID, **AdvisorID**, FirstName, LastName, Email, PhoneNumber, CurrentYear, NumberofCompletedTerms, JobPreferences)
PK: {Student ID}
CKs: {Student ID}, {FirstName, LastName, Email, PhoneNumber, CurrentYear}

StudentDocument (DocumentID, **StudentID**, DocumentName, UploadDate, UploadTime)
PK: {Document ID}
CKs:

Resume (**DocumentID**)
PK: {Document ID}
CKs: {Document ID}

Cover Letter (**DocumentID**)
PK: {Document ID}
CKs: {Document ID}

CompacyContact (FirstName, LastName, **EmployerID**, EmailAdress, PhoneNumber)
PK: {FirstName, LastName, EmployerID}
CKs: {FirstName, LastName, EmployerID}

JobContract (DocumentID, **EmployerID**, StudentID, DatePosted, SalaryOffered, PositionOffered)
PK: {Document ID}
CKs: {Document ID}

JobApplication (ApplicationID, **StudentID**, **ResumeDocumentID**, **JobID**, ApplicationDate, ApplicationName, JobApplicationStatus)
PK: {Document ID}
CKs: {Document ID}

Job (JobID, **EmployerID**, StartDate, Title, JobDescription, PositionLength, DesiredProgram, Salary)
PK: {JobID}
CKs: {JobID}

Employer (EmployerID, EmployerName, Website, HeadQuarterLocation)

PK: {EmployerID}
CKs: {EmployerID}

AppContainsCoverLetter (**ApplicationID**, **DocumentID**)
PK: {ApplicationID, DocumentID}
CKs: {ApplicationID, DocumentID}

StudentGetsJob (**StudentID**, **JobID**, AcceptanceDate, AcceptanceTime)
PK: {StudentID, JobID}
CKs: {StudentID, JobID}

StudentInterviewJobwCompanyContact (**StudentID**, **JobID**, **CompanyContactFirstName**, **CompanyContactLastName**, **EmployerID**, InterviewDate, InterviewLocation, InterviewTime)
PK: {StudentID, JobID}
CKs: {StudentID, JobID}

4. **Functional Dependencies (FD's)**

Student:
FN, LN, EA, PN, CY -> SID
SID -> FN, LN, EA, PN, CY, NCT, JP, AID
EA -> PN

Job:

T, D, PL, SD -> JID
JID -> SD, T, D, PL, DP, S, EID

Co-op Advisor:

EA -> PN
FL, LN, EA, PN -> AID
AID -> FN, LN EA, PN

Employer:

HQL, W -> EN
EID -> EN, W, HQL
EN -> W

CompanyContact:

EA -> PN
FN, LN, EID -> PN, EA

EA, PN, EID -> FN, LN

StudentDocument:

UD, DN, UT -> SID
DID -> UD, DN, SID, UT

JobApplication:

AD, AN, JAS -> AID
SID, JID -> AID
AID -> SID, DID, JID, AD, AN, JAS

JobContract:

DP, TP, SO, PO -> EID, SID
DID -> DP, SO, PO, EID, SID

StudentGetsJob

SID, AD, AT -> JID
JID, AD -> AT
SID, JID -> AD, AT

StudentInterviewJobwCompanyContact:

SID, IL, ID, IT -> JID
SID, JID, CCFN, CCLN, EID -> IL, ID, IT

5. **Normalization**

6. **SQL DDL**

```
CREATE TABLE CoopAdvisor{
   AdvisorID INTEGER,
   FirstName CHAR(40),
   LastName CHAR(40)
   EmailAddress CHAR(40),
   PhoneNumber CHAR(30),
   PRIMARY KEY (AdvisorID)
};

CREATE TABLE Student {
   StudentID INTEGER,
   AdvisorID INTEGER NOT NULL,
   FirstName CHAR(40),
   LastName CHAR(40),
   Email CHAR(40),
   PhoneNumber CHAR(20),
   CurrentYear INTEGER,
   NumberofCompletedTerms INTEGER,
   JobPreferences CHAR(100),
   PRIMARY KEY (StudentID),
   FOREIGN KEY (AdvisorID) REFERENCES CoopAdvisor,
};


CREATE TABLE StudentDocument {
   DocumentID INTEGER,
   DocumentName Char(30),
   UploadDate Date,
   UploadTime Time,
   StudentID Char(20) NOT NULL,
   PRIMARY KEY(DocumentID),
   FOREIGN KEY (StudentID) REFERENCES Student,
};

CREATE TABLE Resumé {
   DocumentID INTEGER,
   PRIMARY KEY(DocumentID),
   FOREIGN KEY (DocumentID) REFERENCES StudentDocument,
};

CREATE TABLE CoverLetter {
   DocumentID INTEGER,
   PRIMARY KEY(DocumentID),
   FOREIGN KEY (DocumentID) REFERENCES StudentDocument,
};

CREATE TABLE JobContract {
   DocumentID INTEGER,
   StudentID CHAR(20),
   DatePosted Date,
   TimePosted Time,
   SalaryOffered INTEGER,
   PositionOffered Char(20),
   EmployerID CHAR(20) NOT NULL,
   PRIMARY KEY(DocumentID),
   FOREIGN KEY (EmployerID) REFERENCES Employer,
};


CREATE TABLE JobApplication {
   ApplicationID INTEGER,
   ApplicationName Char(20),
   StudentID Char(20) NOT NULL,
   ResuméDocumentID Char(20) NOT NULL,
   JobID Char(20) NOT NULL,
   ApplicationDate Date,
   JobApplicationStatus CHAR(20) SET DEFAULT('Pending'),
   PRIMARY KEY (ApplicationID),
   FOREIGN KEY (StudentID) REFERENCES Student,
   FOREIGN KEY (ResuméDocumentID) REFERENCES Resumé(DocumentID),
   FOREIGN KEY (JobID) REFERENCES Job,
};


CREATE TABLE Job {
   JobID INTEGER,
   EmployerID INTEGER NOT NULL,
   StartDate Date,
   Title CHAR(50),
   JobDescription CHAR(400),
   PositionLength CHAR(50),
   DesiredProgram CHAR(50),
   Salary CHAR(50),
   PRIMARY KEY (JobID),
   FOREIGN KEY (EmployerID) REFERENCES Employer,
};

CREATE TABLE Employer {
   EmployerID INTEGER,
   EmployerName CHAR(50),
   Website Char(50),
   HeadQuarterLocation Char(20),
   PRIMARY KEY (EmployerID)

};

CREATE TABLE CompanyContact {
   FirstName CHAR(40),
   LastName CHAR(40),
   PhoneNumber CHAR(40),
   EmailAddress CHAR(40),
   EmployerID INTEGER,

   PRIMARY KEY(CompanyContactName,EmployerID)
   FOREIGN KEY(EmployerID) REFERENCES Employer ON DELETE CASCADE,
};

CREATE TABLE AppContainsCoverLetter {
   ApplicationID INTEGER,
   DocumentID INTEGER,

   PRIMARY KEY (ApplicationID, DocumentID),
   FOREIGN KEY (ApplicationID) REFERENCES JobApplication,
   FOREIGN KEY (DocumentID) REFERENCES CoverLetter,

}


CREATE TABLE StudentGetsJob {
   StudentID INTEGER,
   JobID INTEGER,
   AcceptanceDate Date,
   AcceptanceTime Time,

   PRIMARY KEY (StudentID, JobID)
   FOREIGN KEY (StudentID) REFERENCES Student,
   FOREIGN KEY (JobID) REFERENCES Job,
}

CREATE TABLE StudentInterviewJobwCompanyContact {
   StudentID INTEGER,
   JobID INTEGER,
   CompanyContactFirstName Char(40),
   CompanyContactLastName Char(40)
   EmployerID INTEGER,
   InterviewTime Time,
   InterviewDate Date,
   InterviewLocation CHAR(50),

   PRIMARY KEY (StudentID, JobID, CompanyContactFirstName, CompanyContactLastName,
EmployerID),
   FOREIGN KEY (JobID) REFERENCES Job,
   FOREIGN KEY (StudentID) REFERENCES Student,
   FOREIGN KEY (CompanyContactFirstName, CompanyContactLastName, EmployerID)
REFERENCES CompanyContact(FirstName, LastName, EmployerID),
}
```

7. **SQL insertion**

Task Breakdown:

Nicholas - Create react components and set up GUI, help with implementing queries and api endpoints

Cole - Create project description PDF and README file, work on implementing queries options

Anikait - Work on implementing query options

Time-Line:

August 2nd:

GUI complete, finished creating react components such as places to input “queries”, and output components that can be used to map data through GET requests to the backend server.

Oracle Database sql script created to instantiate tables, utilizing the insertion statements from Milestone 2 we will also use the sql script to populate the tables as well as adding 5 more for each table to allow for meaningful queries

Aug 4th:

Finish project description PDF

Finish implementing INSERT, DELETE, UPDATE, Selection, and Projection query operations.

Aug 6th:

Finish implementing Aggregtion with Group By, Aggregation with Having, Nested Aggregation with Group By, and Division

Aug 7th:

Connect the front-end GUI to the back-end server and create functions that will populate the data into the previously made front-end output components

Aug 8th:

Fix any potential bugs, and any additional components or queries that are needed