# Summary

The foundation, TechForAll, solicits technology donations for students across the city. The devices are distributed by criteria determined by the foundation’s board of directors. The CompU database tracks the requests made by students to receive a device from the charity and the inventory of devices waiting to be distributed.

TechForAll’s mission is to bridge the digital divide and promote equal opportunities in education, TechForAll provides laptops and tablets to students in need, taking into account both their school district and disability.

To achieve their goal, TechForAll partnered closely with local schools and education authorities in the city. They established strong connections with school administrators, teachers, and counselors, who would identify students lacking access to technology due to financial constraints or disabilities.

Through their collaboration, TechForAll has devised a comprehensive process to evaluate each student's needs. They took into consideration the specific school district the student belonged to, ensuring that students from underserved areas were given priority. They also factored in disabilities or special needs, recognizing the importance of providing tailored support to those facing additional challenges.

TechForAll works diligently to gather resources, securing donations from individuals, local businesses, and corporate sponsors who share their vision.

# Goal

The goal of the database is to manage the requests, distribution, and inventory of gifted devices.

# Stakeholders

* Students
* Donors
* TechForAll Foundation

# Business Rules

* A student can request one or more devices
* A district can have more than one school
* A donor can donate more than one device to the inventory
* A district can only have one ranking
* A student can only have one disability ranking
* A district can only have one district rank
* A student can only have one grade rank

# Glossary

*Inventory:* The number of devices available for student requests.

*Students:* The requestors and recipients of devices.

*Requests:* Submitted by students to the foundation to be reviewed by CompU and then approved or denied based on the foundation's criteria.

*Device types:* The brand of laptops or tablets that are donated and distrusted by the foundation.

*Districts:* A special-purpose district that operates local public primary and secondary schools.

*Schools:* An institution for educating students.

*Donors:* The names of people who donated devices to TechForAll.

*District ranks:* Order of districts based on criteria defined by the charity’s board of directors. Ranked from 1-5, 1 beginning the ‘best.’

*Disability ranks:* Order of disabilities based on criteria defined by the charity’s board of directors. Ranked from 1-6. 6 is the most severe.

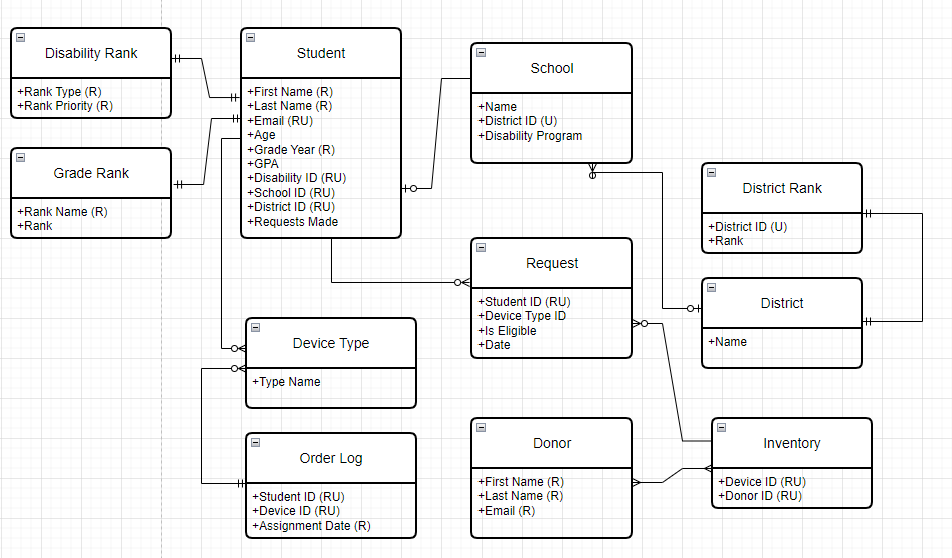
*Grade ranks*: The order of grades from 8th grade to 12th grade and their rank, 8th being 1 and 12th being 5.

*Order logs:* The transactions of devices approved for students.

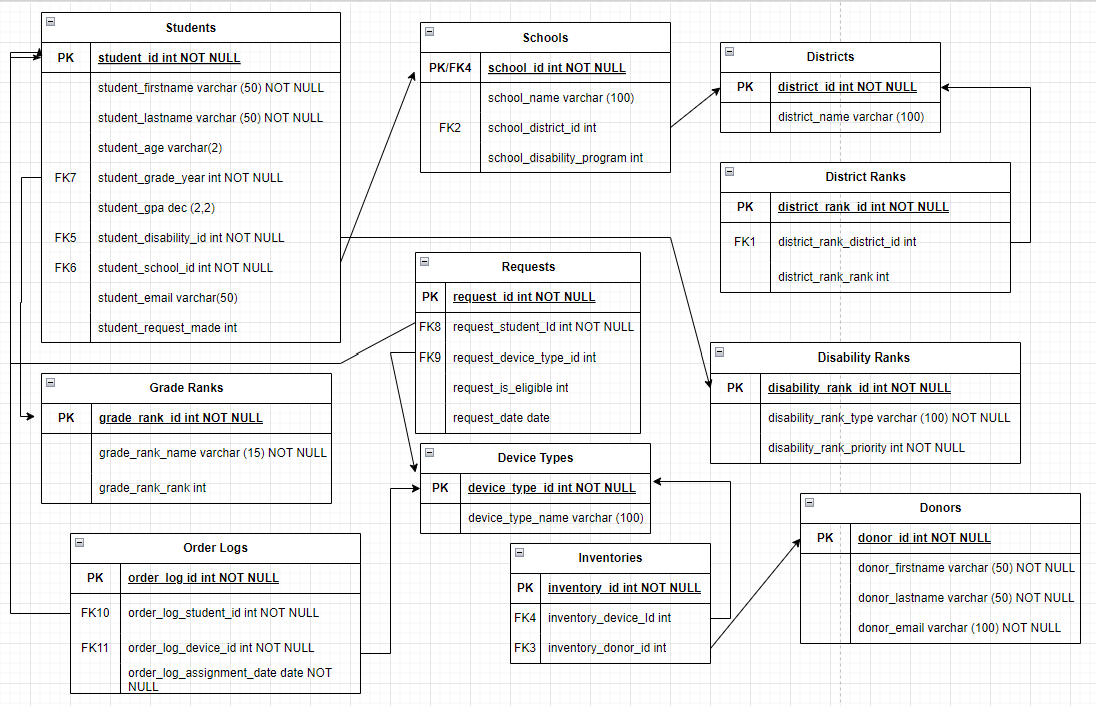
# Data Questions

1. How can we notify a donor once their donated laptops have been given to a student?
2. How many students from each district have requested devices?
3. Who are our top donors?
4. How many students from each district have been approved for device allocation?
5. How many laptops and tablets are currently available in the inventory?
6. How will students be measured against GPA, disability, grade, and district?

# Conceptual Model



# Logical Model



# Physical Database

-- down script

if not exists( select \* from sys.databases where name = 'compu')

create database compu

GO

use compu

GO

--drop procedures

IF OBJECT\_ID('EvaluateStudentRequests') IS NOT NULL

DROP PROCEDURE EvaluateStudentRequests

-- Drop the procedure if it already exists

IF OBJECT\_ID('AssignDeviceToStudent', 'P') IS NOT NULL

DROP PROCEDURE AssignDeviceToStudent;

GO

-- drop views

IF OBJECT\_ID('EvaluateStudentRequestView') IS NOT NULL

DROP VIEW EvaluateStudentRequestView

-- Drop tables in reverse order of creation

DROP TABLE IF EXISTS order\_logs;

DROP TABLE IF EXISTS requests;

DROP TABLE IF EXISTS students;

DROP TABLE IF EXISTS grade\_ranks;

DROP TABLE IF EXISTS inventorys;

DROP TABLE IF EXISTS device\_types;

DROP TABLE IF EXISTS donors;

DROP TABLE IF EXISTS schools;

DROP TABLE if exists disability\_ranks;

DROP table if exists district\_ranks;

DROP TABLE IF EXISTS districts;

-- up script

-- Create table for districts

CREATE TABLE districts (

district\_id INT NOT NULL identity,

district\_name VARCHAR(100)

CONSTRAINT pk\_districts\_district\_id primary key (district\_id)

);

-- Insert records for districts

INSERT INTO districts (district\_name) VALUES ( 'North School District');

INSERT INTO districts (district\_name) VALUES ( 'West School District');

INSERT INTO districts (district\_name) VALUES ('Town Square District');

INSERT INTO districts (district\_name) VALUES ('Uptown District');

INSERT INTO districts (district\_name) VALUES ('East Town District');

-- create table for district rank  
CREATE TABLE district\_ranks (

district\_rank\_id INT NOT NULL identity,

district\_rank\_district\_id INT,

district\_rank\_rank INT,

CONSTRAINT pk\_district\_ranks\_district\_rank\_id PRIMARY KEY (district\_rank\_id),

FOREIGN KEY (district\_rank\_district\_id) REFERENCES districts(district\_id)

);

-- insert into district ranks

INSERT INTO district\_ranks (district\_rank\_district\_id, district\_rank\_rank) VALUES ( 1, 5);

INSERT INTO district\_ranks (district\_rank\_district\_id, district\_rank\_rank) VALUES (2, 3);

INSERT INTO district\_ranks (district\_rank\_district\_id, district\_rank\_rank) VALUES (3, 2);

INSERT INTO district\_ranks (district\_rank\_district\_id, district\_rank\_rank) VALUES ( 4, 4);

INSERT INTO district\_ranks (district\_rank\_district\_id, district\_rank\_rank) VALUES (5, 1);

-- create table for disability priority

create table disability\_ranks(

disability\_rank\_id INT NOT NULL identity,

disability\_rank\_type VARCHAR(100) NOT NULL,

disability\_rank\_priority INT NOT NULL

CONSTRAINT pk\_disability\_ranks\_disability\_rank\_id PRIMARY KEY (disability\_rank\_id)

);

-- Insert records into disability\_ranks

INSERT INTO disability\_ranks (disability\_rank\_type, disability\_rank\_priority) VALUES ('Multiple Disabilities', 5);

INSERT INTO disability\_ranks (disability\_rank\_type, disability\_rank\_priority) VALUES ('Learning Disability', 4);

INSERT INTO disability\_ranks (disability\_rank\_type, disability\_rank\_priority) VALUES ( 'Hearing Disability', 3);

INSERT INTO disability\_ranks (disability\_rank\_type, disability\_rank\_priority) VALUES ( 'Visual Disability', 2);

INSERT INTO disability\_ranks (disability\_rank\_type, disability\_rank\_priority) VALUES ( 'Physical Disability', 1);

INSERT INTO disability\_ranks (disability\_rank\_type, disability\_rank\_priority) VALUES ( 'None', 0);

-- Create table for schools

CREATE TABLE schools (

school\_id INT NOT NULL identity,

school\_name VARCHAR(100),

school\_district\_id INT,

school\_disabilities\_program TINYINT DEFAULT 0,

CONSTRAINT pk\_schools\_school\_id PRIMARY KEY (school\_id),

CONSTRAINT fk\_schools\_district

FOREIGN KEY (school\_district\_id)

REFERENCES districts(district\_id)

);

-- Inserting records for District 1

INSERT INTO schools (school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ('North Middle School', 1, 0);

INSERT INTO schools (school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ('North High School', 1, 1);

-- Inserting records for District 2

INSERT INTO schools (school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ('West High School', 2, 0);

INSERT INTO schools ( school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ('West Middle School', 2, 1);

INSERT INTO schools (school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ('Tom Barady High School', 2, 0);

-- Inserting records for District 3

INSERT INTO schools (school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ('Town Middle School', 3, 1);

INSERT INTO schools ( school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ('Town High School', 3, 0);

-- Inserting records for District 4

INSERT INTO schools ( school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ( 'Uptown Middle School', 4, 1);

INSERT INTO schools ( school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ( 'Uptown High School', 4, 0);

-- Inserting records for District 5

INSERT INTO schools ( school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ( 'East Middle School', 5, 0);

INSERT INTO schools ( school\_name, school\_district\_id, school\_disabilities\_program)

VALUES ( 'East High School', 5, 1);

-- Create table for donors

CREATE TABLE donors (

donor\_id INT NOT NULL identity,

donor\_firstname VARCHAR(50) NOT NULL,

donor\_lastname VARCHAR(50) NOT NULL,

donor\_email VARCHAR(100) NOT NULL

CONSTRAINT pk\_donors\_donor\_id PRIMARY KEY (donor\_id)

);

-- Insert records for donors

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'John', 'Smith', '[john.smith@example.com](mailto:john.smith@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'Jane', 'Doe', '[jane.doe@example.com](mailto:jane.doe@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'Michael', 'Johnson', '[michael.johnson@example.com](mailto:michael.johnson@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'Emily', 'Brown', '[emily.brown@example.com](mailto:emily.brown@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'David', 'Miller', '[david.miller@example.com](mailto:david.miller@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'Olivia', 'Taylor', '[olivia.taylor@example.com](mailto:olivia.taylor@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'Daniel', 'Anderson', '[daniel.anderson@example.com](mailto:daniel.anderson@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'Sophia', 'Clark', '[sophia.clark@example.com](mailto:sophia.clark@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'Matthew', 'Walker', '[matthew.walker@example.com](mailto:matthew.walker@example.com)');

INSERT INTO donors ( donor\_firstname, donor\_lastname, donor\_email) VALUES ( 'Ava', 'Harris', '[ava.harris@example.com](mailto:ava.harris@example.com)');

-- Create table for devices

CREATE TABLE device\_types (

device\_type\_id INT NOT NULL identity,

device\_type\_name VARCHAR(100)

CONSTRAINT pk\_device\_types\_device\_type\_id primary key (device\_type\_id)

);

-- Insert records into device\_types

INSERT INTO device\_types ( device\_type\_name) VALUES ('iPad');

INSERT INTO device\_types (device\_type\_name) VALUES ('Samsung Tablet');

INSERT INTO device\_types (device\_type\_name) VALUES ('Macbook');

INSERT INTO device\_types (device\_type\_name) VALUES ('Chromebook');

-- Create table for inventory

CREATE TABLE inventorys (

inventory\_id INT NOT NULL identity,

inventory\_device\_id INT,

inventory\_donor\_id INT,

CONSTRAINT pk\_inventorys\_inventory\_id primary key (inventory\_id),

CONSTRAINT fk\_inventorys\_device\_id

FOREIGN KEY (inventory\_device\_id)

REFERENCES device\_types(device\_type\_id),

constraint fk\_inventorys\_donor\_id

FOREIGN KEY (inventory\_donor\_id)

REFERENCES donors(donor\_id)

);

-- Insert records into inventorys

-- Inserting records for inventory\_id 1-10

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 1, 1);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 2, 2);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 3);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 4, 4);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 5);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 1, 6);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 1, 7);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 8);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 4, 9);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 10);

-- Inserting records for inventory\_id 11-20

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 1, 1);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 2, 1);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 1);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 4, 4);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 1, 6);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 1, 6);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 2, 7);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 8);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 4, 6);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 2, 10);

-- Inserting records for inventory\_id 21-30

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 1, 1);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 2, 2);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 7);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 4, 7);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 4, 5);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 1, 1);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 2, 1);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 8);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 4, 10);

INSERT INTO inventorys ( inventory\_device\_id, inventory\_donor\_id)

VALUES ( 3, 10);

--create table for grade\_ranks

create table grade\_ranks(

grade\_rank\_id INT NOT NULL ,

grade\_rank\_name VARCHAR(15) NOT NULL,

grade\_rank\_rank INT NULL

CONSTRAINT pk\_grade\_ranks\_grade\_rank\_id PRIMARY KEY (grade\_rank\_id)

);

--insert values for grade\_ranks

INSERT INTO grade\_ranks (grade\_rank\_id, grade\_rank\_name, grade\_rank\_rank)

VALUES

(8, '8th Grade', 1),

(9, '9th Grade', 2),

(10, '10th Grade', 3),

(11, '11th Grade', 4),

(12, '12th Grade', 5);

-- Create table for students

CREATE TABLE students (

student\_id INT NOT NULL identity,

student\_firstname VARCHAR(50) NOT NULL,

student\_lastname VARCHAR(50) NOT NULL,

student\_email VARCHAR(50) NOT NULL UNIQUE,

student\_age INT NULL,

student\_grade\_year INT NOT NULL,

student\_gpa DECIMAL (3,2) NULL,

student\_disability\_id INT not null,

student\_school\_id INT NOT NULL,

student\_requests\_made INT NULL

CONSTRAINT pk\_students\_student\_id PRIMARY key (student\_id),

CONSTRAINT fk\_students\_disability\_id

FOREIGN KEY (student\_disability\_id)

REFERENCES disability\_ranks(disability\_rank\_id),

CONSTRAINT fk\_students\_school

FOREIGN KEY (student\_school\_id)

REFERENCES schools(school\_id),

constraint fk\_students\_student\_grade\_year

FOREIGN key (student\_grade\_year)

REFERENCES grade\_ranks(grade\_rank\_id)

);

ALTER TABLE students

ALTER COLUMN student\_gpa DECIMAL(3, 2);

ALTER TABLE students

DROP COLUMN student\_requests\_made;

-- Insert records for students

INSERT INTO students ( student\_firstname, student\_lastname, student\_email, student\_age, student\_grade\_year, student\_gpa, student\_disability\_id, student\_school\_id)

VALUES

('John', 'Smith', '[john.smith@example.edu](mailto:john.smith@example.edu)', 16, 10, 3.75, 1, 1),

('Emily', 'Johnson', '[emily.johnson@funnyemail.edu](mailto:emily.johnson@funnyemail.edu)', 15, 9, 3.88, 6, 1),

( 'Michael', 'Williams', '[michael.williams@lolmail.edu](mailto:michael.williams@lolmail.edu)', 17, 11, 3.60, 4, 1),

( 'Olivia', 'Brown', '[olivia.brown@hilariousmail.edu](mailto:olivia.brown@hilariousmail.edu)', 14, 8, 3.92, 5, 2),

( 'James', 'Jones', '[james.jones@crazymail.edu](mailto:james.jones@crazymail.edu)', 15, 9, 2.80, 1, 2),

( 'Sophia', 'Davis', '[sophia.davis@randommail.edu](mailto:sophia.davis@randommail.edu)', 16, 10, 3.65, 2, 3),

( 'Daniel', 'Miller', '[daniel.miller@sillymail.edu](mailto:daniel.miller@sillymail.edu)', 15, 9, 3.95, 1, 4),

( 'Ava', 'Wilson', '[ava.wilson@weirdmail.edu](mailto:ava.wilson@weirdmail.edu)', 17, 11, 3.72, 5, 4),

( 'Liam', 'Taylor', '[liam.taylor@goofymail.edu](mailto:liam.taylor@goofymail.edu)', 14, 8, 2.81, 1, 5),

( 'Emma', 'Anderson', '[emma.anderson@oddmail.edu](mailto:emma.anderson@oddmail.edu)', 15, 9, 3.88, 2, 5),

( 'Benjamin', 'Martinez', '[benjamin.martinez@crazymail.edu](mailto:benjamin.martinez@crazymail.edu)', 16, 10, 3.70, 4, 6),

( 'Mia', 'Hernandez', '[mia.hernandez@funnyemail.edu](mailto:mia.hernandez@funnyemail.edu)', 14, 8, 3.91, 5, 6),

( 'Elijah', 'Lopez', '[elijah.lopez@lolmail.edu](mailto:elijah.lopez@lolmail.edu)', 15, 9, 2.76, 5, 6),

( 'Charlotte', 'Garcia', '[charlotte.garcia@hilariousmail.edu](mailto:charlotte.garcia@hilariousmail.edu)', 17, 11, 3.84, 2, 7),

( 'William', 'Smith', '[william.smith@crazymail.edu](mailto:william.smith@crazymail.edu)', 14, 8, 3.90, 6, 7),

( 'Amelia', 'Johnson', '[amelia.johnson@randommail.edu](mailto:amelia.johnson@randommail.edu)', 15, 9, 3.78, 2, 7),

( 'Alexander', 'Williams', '[alexander.williams@weirdmail.edu](mailto:alexander.williams@weirdmail.edu)', 16, 10, 3.68, 1, 8),

( 'Harper', 'Brown', '[harper.brown@hilariousmail.edu](mailto:harper.brown@hilariousmail.edu)', 14, 8, 3.89, 2, 8),

( 'Daniel', 'Jones', '[daniel.jones@crazymail.edu](mailto:daniel.jones@crazymail.edu)', 15, 9, 2.83, 6, 9),

('Evelyn', 'Davis', '[evelyn.davis@randommail.edu](mailto:evelyn.davis@randommail.edu)', 17, 11, 3.79, 2, 9),

( 'Jacob', 'Miller', '[jacob.miller@sillymail.edu](mailto:jacob.miller@sillymail.edu)', 15, 9, 3.82, 6, 9),

( 'Abigail', 'Wilson', '[abigail.wilson@weirdmail.edu](mailto:abigail.wilson@weirdmail.edu)', 16, 12, 2.73, 3, 10),

( 'Noah', 'Taylor', '[noah.taylor@goofymail.edu](mailto:noah.taylor@goofymail.edu)', 14, 8, 3.93, 6, 10),

( 'Sofia', 'Anderson', '[sofia.anderson@oddmail.edu](mailto:sofia.anderson@oddmail.edu)', 15, 9, 3.87, 2, 11),

( 'Logan', 'Martinez', '[logan.martinez@crazymail.edu](mailto:logan.martinez@crazymail.edu)', 17, 12, 2.77, 5, 11);

-- Create table for requests

CREATE TABLE requests (

request\_id INT NOT NULL identity,

request\_student\_id INT,

request\_device\_type\_id INT,

request\_date DATE,

request\_is\_eligible TINYINT,

CONSTRAINT pk\_requests\_request\_id PRIMARY key (request\_id),

CONSTRAINT fk\_requests\_student

FOREIGN KEY (request\_student\_id)

REFERENCES students(student\_id),

CONSTRAINT fk\_requests\_device

FOREIGN KEY (request\_device\_type\_id)

REFERENCES device\_types(device\_type\_id)

);

-- insert into requests

INSERT INTO requests (request\_student\_id, request\_device\_type\_id, request\_date, request\_is\_eligible)

VALUES

(1, 1, '2022-08-15',1),

(2, 2, '2022-07-04',0),

(3, 3, '2022-08-17',1),

(4, 4, '2022-09-19',1),

(5, 1, '2022-10-25', 0),

(6, 2, '2022-08-15',1),

(7, 2, '2022-08-20',0),

(8, 4, '2022-07-12',1);

create table order\_logs (

order\_log\_id int NOT NULL identity,

order\_log\_student\_id int NOT NULL,

order\_log\_device\_id int NOT NULL,

order\_log\_assignment\_date DATE NOT NULL,

CONSTRAINT pk\_order\_logs\_order\_log\_id primary key (order\_log\_id),

CONSTRAINT fk\_order\_logs\_student\_id

FOREIGN key (order\_log\_student\_id)

REFERENCES students(student\_id),

CONSTRAINT fk\_order\_logs\_order\_log\_device\_id

FOREIGN key (order\_log\_device\_id)

REFERENCES device\_types(device\_type\_id)

)

-- now lets create a procedure to insert a new request and see if a student who is making a request is already in the database

use compu

GO

DROP PROCEDURE IF EXISTS dbo.InsertRequest;

GO

CREATE PROCEDURE dbo.InsertRequest

@firstName varchar(50),

@lastName varchar(50),

@email varchar(50),

@grade INT,

@gpa DECIMAL(3,2),

@disability varchar(100),

@school varchar(100),

@request\_date DATE,

@deviceType VARCHAR(100)

AS

BEGIN

-- checking whether or not the student is a new student in the database

If Not Exists (Select 1 from students where student\_email = @email)

BEGIN

--inserting new student

Insert into students(student\_firstname, student\_lastname, student\_email, student\_grade\_year, student\_gpa, student\_disability\_id, student\_school\_id)

Values (@firstName, @lastName, @email, @grade, @gpa, (select disability\_rank\_id from disability\_ranks where disability\_rank\_type = @disability),

(select school\_id from schools where school\_name = @school)

);

END

--insert the request

INSERT INTO requests (request\_student\_id, request\_device\_type\_id, request\_date)

VALUES ((select student\_id from students where student\_email=@email), (select device\_type\_id from device\_types where device\_type\_name =@deviceType), @request\_date );

END

GO

declare @studentfirstname varchar(50), @studentlastname varchar(50), @email varchar(50), @grade INT, @gpa decimal (3,2), @disability varchar(100), @school varchar(100),

@request\_date DATE, @devicetype VARCHAR(100)

SET @studentfirstname = 'Wade'

set @studentlastname = 'Wilson'

set @email = '[wadewilly@deadpool.edu](mailto:wadewilly@deadpool.edu)'

set @grade = 10

set @gpa = 3.82

set @disability = 'Learning Disability'

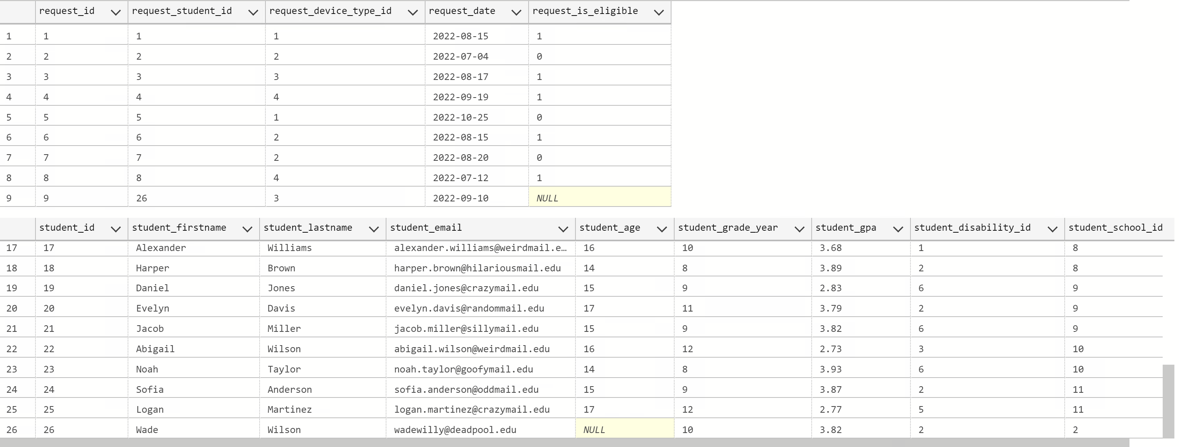
set @school = 'North High School'

set @request\_date = '2022-09-10'

set @devicetype = 'Macbook'

EXEC dbo.InsertRequest @studentfirstname, @studentlastname, @email, @grade, @gpa, @disability, @school, @request\_date, @devicetype

select \* from requests

select \* from students  


-- we can see that a student has been added to the table, students, and a new request as well

-- now lets create a view for us to decide if they get a laptop, enter in a student to get a composite score back and see how high that composite score is, anything in the 3 or above is a great score.

GO

CREATE VIEW EvaluateStudentRequestView

AS

SELECT

s.student\_id,

s.student\_firstname + ' ' + s.student\_lastname as Student\_name,

s.student\_gpa,

dr.disability\_rank\_type,

g.grade\_rank\_name,

r.request\_date,

sc.school\_name,

dis.district\_name,

(

(s.student\_gpa / 4.0) \* 0.3 +

dr.disability\_rank\_priority \* 0.3 +

g.grade\_rank\_rank \* 0.2 + 1`

disr.district\_rank\_rank \* 0.2

) AS composite\_score,

RANK() OVER (ORDER BY

(s.student\_gpa / 4.0) \* 0.3 +

dr.disability\_rank\_priority \* 0.3 +

g.grade\_rank\_rank \* 0.2 +

disr.district\_rank\_rank \* 0.2 DESC

) AS student\_rank

FROM

requests r

left join students as s on r.request\_student\_id = s.student\_id

left join disability\_ranks as dr on s.student\_disability\_id = dr.disability\_rank\_id

left join schools as sc on s.student\_school\_id = sc.school\_id

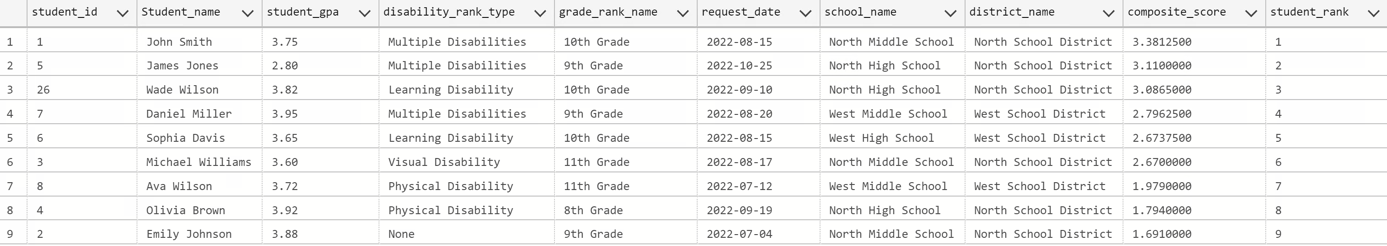
left join grade\_ranks as g on g.grade\_rank\_id = s.student\_grade\_year

left join districts as dis on sc.school\_district\_id = dis.district\_id

left join district\_ranks as disr on disr.district\_rank\_district\_id = dis.district\_id

GO

select \* from EvaluateStudentRequestView



-- Create Procedure for AssignDevicetoStudent if they are evaluated and accepted

GO

CREATE PROCEDURE AssignDeviceToStudent (@student\_email VARCHAR(50), @device\_type\_name VARCHAR(100))

AS

BEGIN

-- Check if the student exists in the 'students' table

IF NOT EXISTS (SELECT 1 FROM students WHERE student\_email = @student\_email)

BEGIN

RAISERROR ('Student does not exist.',12,1)

RETURN

END

-- check if requested device type is available in inventory

IF NOT Exists (Select 1 from inventorys left join device\_types on device\_type\_id = inventory\_device\_id where device\_type\_name = @device\_type\_name)

BEGIN

RAISERROR ( 'Device type is not available inventory.', 16, 1)

RETURN

END

-- Check if there is an available device of the requested type

DECLARE @deviceId INT

SELECT TOP 1 @deviceId = inventory\_device\_id

FROM inventorys left join device\_types on device\_type\_id = inventory\_device\_id

WHERE device\_type\_name = @device\_type\_name

IF @deviceId IS NULL

BEGIN

RAISERROR('No available device of the requested type.', 12, 1)

RETURN

END

-- Assign the device to the student

delete from inventorys

WHERE inventory\_device\_id = @deviceId

-- Delete the request

DELETE FROM requests

WHERE request\_student\_id = (SELECT student\_id FROM students WHERE student\_email = @student\_Email)

AND request\_device\_type\_id = (SELECT device\_type\_id FROM device\_types WHERE device\_type\_name = @device\_type\_name)

-- Store the assignment information in a separate table (e.g., assignments\_history)

INSERT INTO order\_logs (order\_log\_student\_id, order\_log\_device\_id, order\_log\_assignment\_date)

VALUES ((SELECT student\_id FROM students WHERE student\_email = @student\_Email), @deviceId, GETDATE())

-- Return success message

SELECT 'Device assigned successfully.' AS Message

END

-- Test Procedure AssignDevicetoStudent

GO

declare @student\_email VARCHAR(50) , @device\_type\_name Varchar(50)

set @student\_email = '[emily.johnson@funnyemail.edu](mailto:emily.johnson@funnyemail.edu)'

set @device\_type\_name = 'Macbook'

EXEC AssignDeviceToStudent @student\_email , @device\_type\_name

select \* from order\_logs

-- Data Question 1: How can we notify a donor once their donated laptops have been given to a student?

IF OBJECT\_ID('NotifyDonor', 'P') IS NOT NULL

DROP PROCEDURE NotifyDonor;

GO

CREATE PROCEDURE NotifyDonor (@donor\_id INT)

AS

BEGIN

-- Check if the donor exists in the 'donors' table

IF EXISTS (SELECT 1 FROM donors WHERE donor\_id = @donor\_id)

BEGIN

-- Get the donor's email

DECLARE @donor\_email VARCHAR(100);

SELECT @donor\_email = donor\_email FROM donors WHERE donor\_id = @donor\_id;

-- Get the student's full name

DECLARE @student\_fullname VARCHAR(100);

SELECT @student\_fullname = CONCAT(student\_firstname, ' ', student\_lastname)

FROM students WHERE student\_id = @student\_fullname;

-- Compose the email message

DECLARE @message VARCHAR(MAX);

SET @message = CONCAT(

'Dear Donor,',

CHAR(50), CHAR(10), CHAR(13), CHAR(10),

'We want to inform you that your donated device has been assigned to a student.',

CHAR(50), CHAR(10), CHAR(13), CHAR(10),

'Student Name: ', @student\_fullname,

CHAR(50), CHAR(10), CHAR(13), CHAR(10),

'Thank you for your generous donation!',

CHAR(50), CHAR(10), CHAR(13), CHAR(10),

'Best regards,',

CHAR(50), CHAR(10),

'TechForAll'

);

-- Send the email to the donor

EXEC send\_dbmail

@recipients = @donor\_email,

@subject = 'Device Donation Notification',

@body = @message;

PRINT 'Donor notification sent successfully.';

END

ELSE

BEGIN

PRINT 'Donor does not exist.';

END

END

-- Data Question 2: How many students from each district have requested a device?

SELECT d.district\_name , COUNT(r.request\_student\_id) AS request\_count

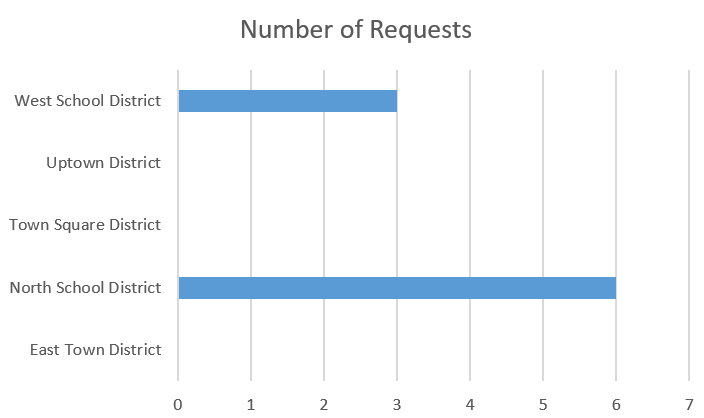
FROM districts d

LEFT JOIN schools s ON d.district\_id = s.school\_district\_id

LEFT JOIN students st ON s.school\_id = st.student\_school\_id

LEFT JOIN requests r ON st.student\_id = r.request\_student\_id

GROUP BY d.district\_name;



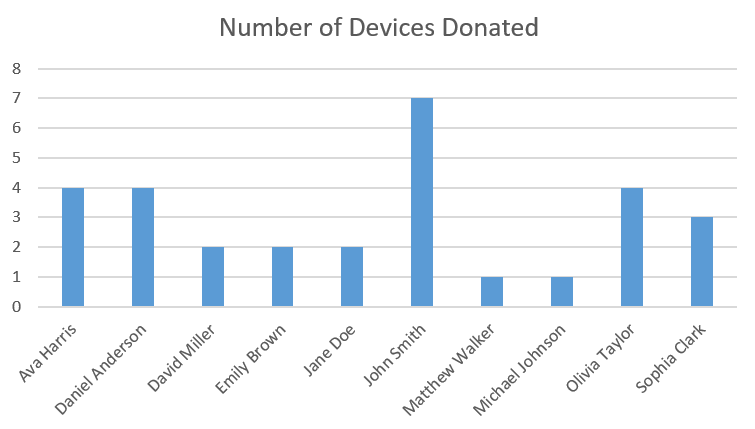
-- Data Question 3: Who are our top donors?

select d.donor\_firstname + ' ' + d.donor\_lastname as donor\_name, COUNT(i.inventory\_donor\_id) as number\_of\_devices

from donors d

left join inventorys as i on d.donor\_id = i.inventory\_donor\_id

group by d.donor\_lastname, d.donor\_firstname



--Data Question 4: How many students from each district have been approved for device allocation?

select d.district\_name, count(s.student\_id) as Number\_of\_Approved\_Students

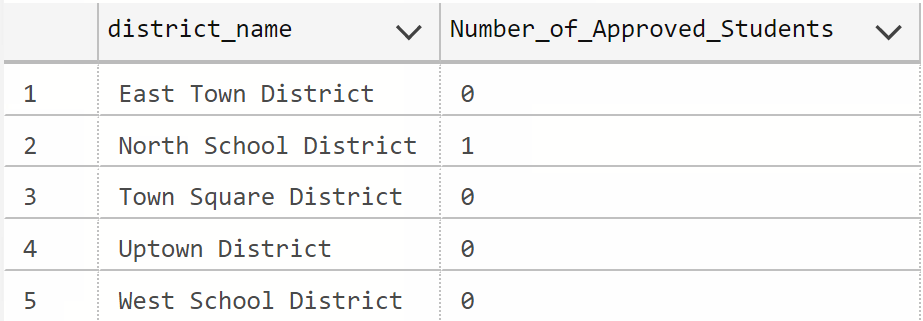
from students s

right join order\_logs as o on o.order\_log\_student\_id = s.student\_id

left join schools as sc on s.student\_school\_id = sc.school\_id

right join districts as d on d.district\_id = sc.school\_district\_id

Group by d.district\_name



--Data Question 5: How many laptops and tablets are currently available in the inventory?

select \* from inventorys

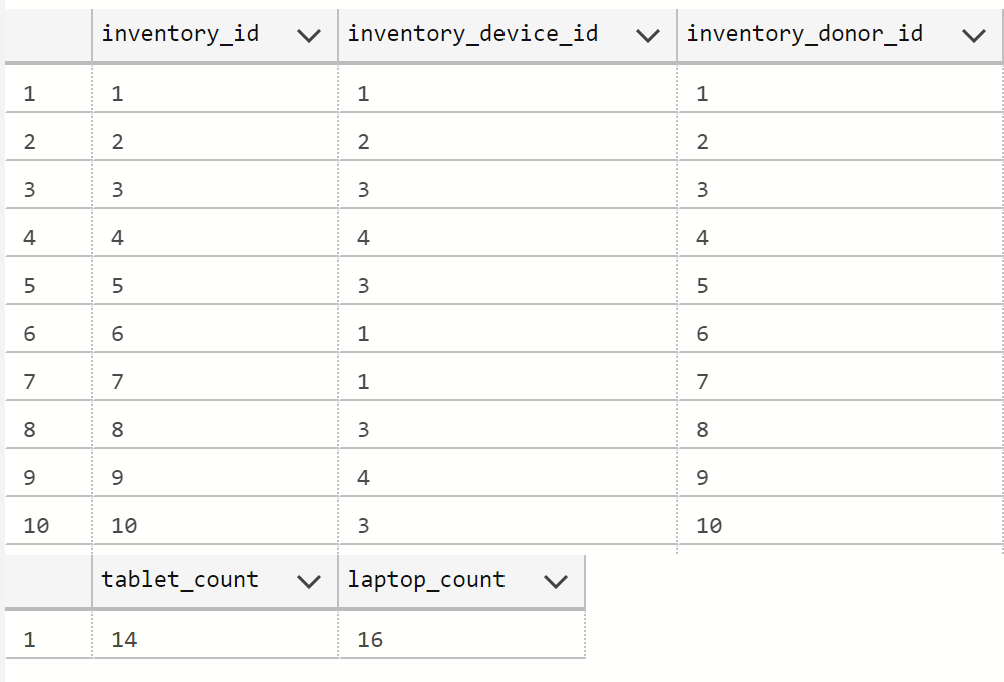
SELECT

SUM(CASE WHEN inventory\_device\_id IN (1,2) THEN 1 ELSE 0 END) AS tablet\_count,

SUM(CASE WHEN inventory\_device\_id IN (3,4) THEN 1 ELSE 0 END) AS laptop\_count

FROM

inventorys



-- Data Question 6: How will students be measured against GPA, disability, grade, and district?

SELECT

s.student\_id,

s.student\_firstname + ' ' + s.student\_lastname as Student\_name,

s.student\_gpa,

dr.disability\_rank\_type,

g.grade\_rank\_name,

r.request\_date,

sc.school\_name,

dis.district\_name,

(

(s.student\_gpa / 4.0) \* 0.3 +

dr.disability\_rank\_priority \* 0.3 +

g.grade\_rank\_rank \* 0.2 +

disr.district\_rank\_rank \* 0.2

) AS composite\_score,

RANK() OVER (ORDER BY

(s.student\_gpa / 4.0) \* 0.3 +

dr.disability\_rank\_priority \* 0.3 +

g.grade\_rank\_rank \* 0.2 +

disr.district\_rank\_rank \* 0.2 DESC

) AS student\_rank

FROM

requests r

left join students as s on r.request\_student\_id = s.student\_id

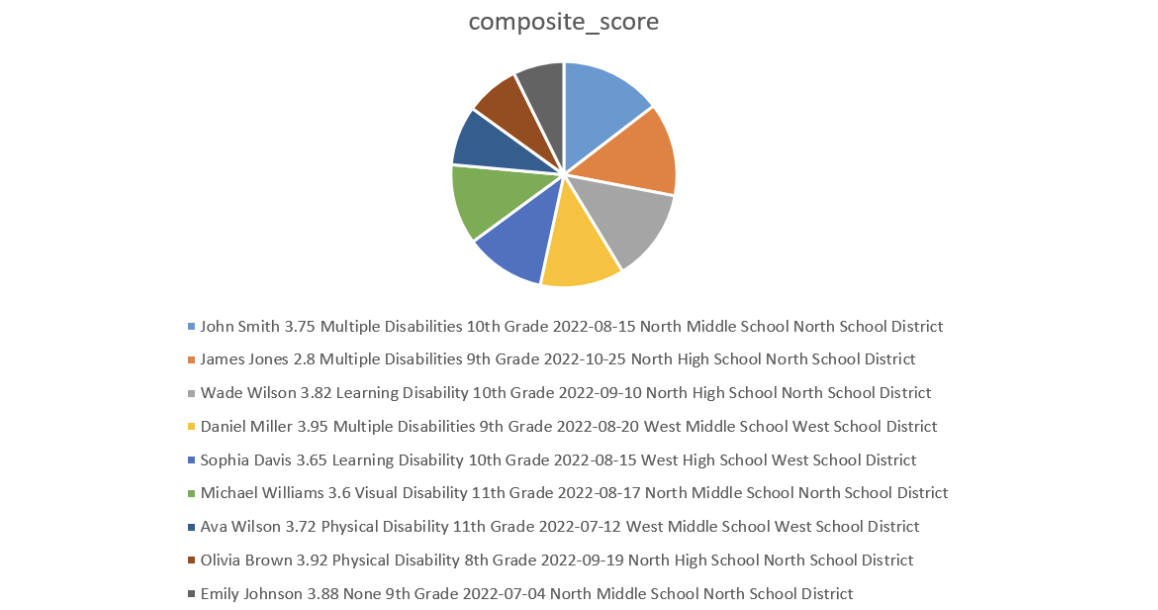
left join disability\_ranks as dr on s.student\_disability\_id = dr.disability\_rank\_id

left join schools as sc on s.student\_school\_id = sc.school\_id

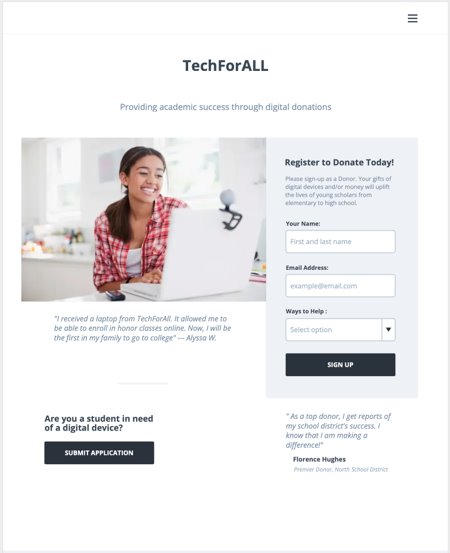
left join grade\_ranks as g on g.grade\_rank\_id = s.student\_grade\_year

left join districts as dis on sc.school\_district\_id = dis.district\_id

left join district\_ranks as disr on disr.district\_rank\_district\_id = dis.district\_id



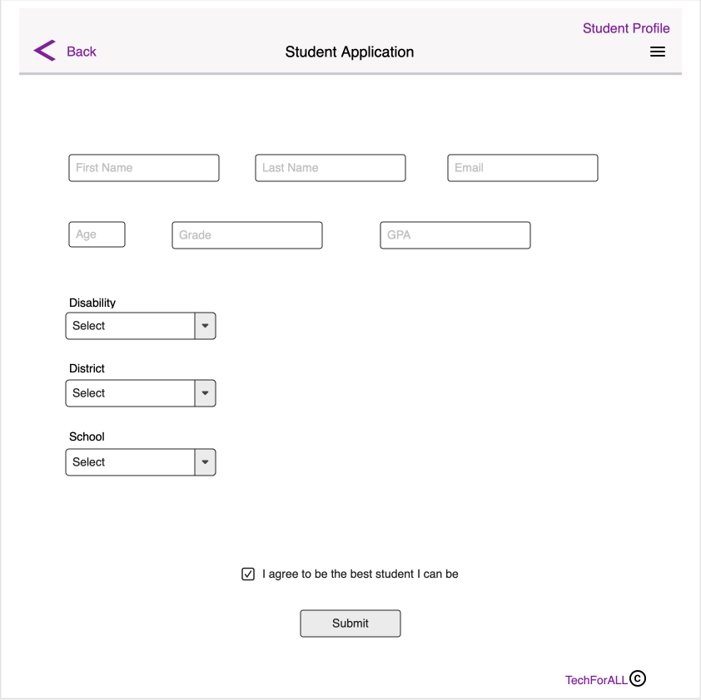
# Application Mock-ups

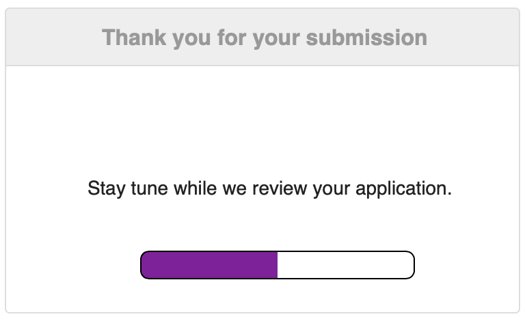


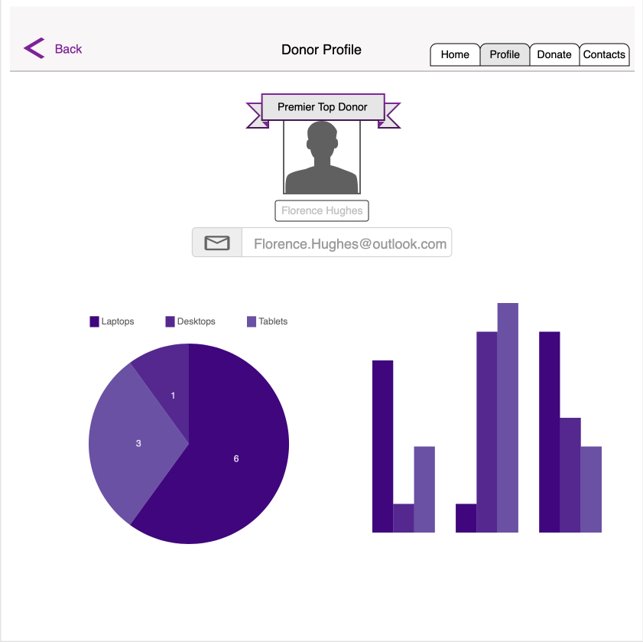
The TechForAll interface acts as a bridge between the donors and the students that need digital equipment.

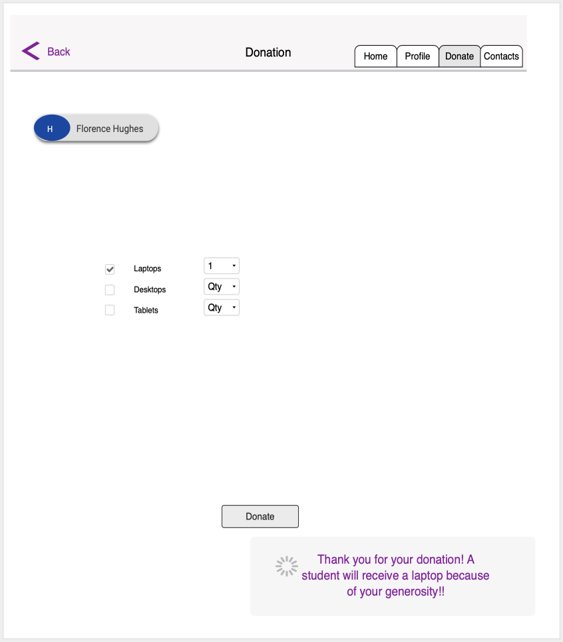
Once a donor signs up with their name and email, their profile is created. From their profile, donors are now able to make donations, see their contribution status, as well as observe how well TechForAll is performing.

Students are also able to submit applications for equipment.









# Reflection

Throughout this project, we learned the importance of well-kept data and an organized database. The use of the logical model really helped us in shaping the database and making sure we had all the components. If we had to do it again, we think more input data would have been very helpful since it would have made our data questions more useful. Right now, there isn’t enough data to make any large statements on which students and from what district are benefiting the most from our charity. None of us were too familiar with Microsoft Access and it proved very difficult to find proper models to showcase what we have done. In the end we think we all learned a valuable amount of SQL and better understand the use of procedures, views, and triggers.

# Summary

Making this database taught us how to use SQL to better understand the use of databases. The charity database we built for TechForALL we feel can be used by an actual company. We really enjoyed this course and what it taught us. The more we worked on the projects and had to figure out roadblocks, the more we got out of this course. Overall, we feel everything we deployed and learned was very useful and has encouraged us to continue learning SQL and better our understanding.