Spartan Market

Team 17

Maan Singh

Tien Ly

Ge Ou

Project Requirement

- **Project Description**

The database application we are proposing is titled ‘Spartan Market.’ The application is to serve students who enjoy shopping and want to save money from the shopping. “Spartan Market” has cheaper men’s, women’s, books and stationery from around the world. Items come directly from factories around the world, and we lower the cost of renting stores and hiring workers. We only have warehouses and internet to manage the market. Our pricing items are cheaper than the others. Additionally, when we get the location from the customer, we would give the lower pricing based on his/her location that items were produced there. Because the long distance delivery fee will be saved.

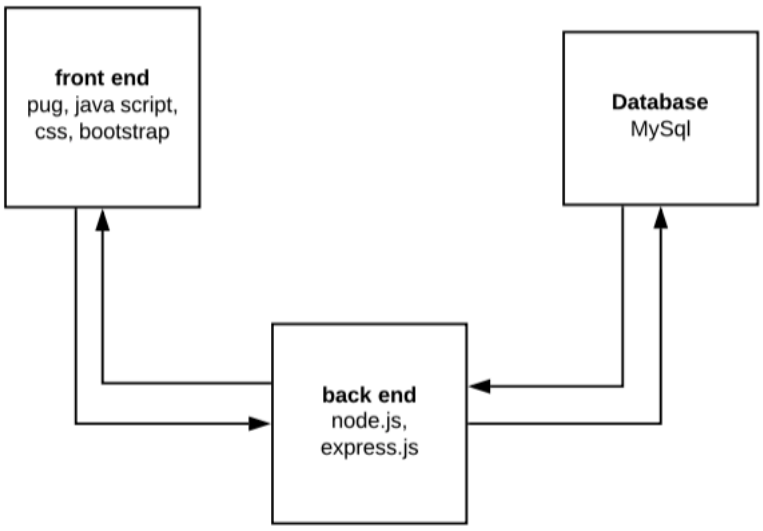
Nowadays, people spend a lot of time on social media and watching videos. Although people can relieve stress when they use these applications, when they are off, people feel lonely and make them want to go online again which can lead to an addiction. These forms of entertainment do not bring real happiness to people.

According to Felecia in “10 Benefits Of Reading”, she mentions that reading helps people to learn and improves memory, develop good analytical skills, expand their vocabulary, improve writing skills, relaxation, improve concentration, inspire them, even reading with their kids to increase time with families, improve their language skills, and wrap up. We want to create a platform for people who like to read and make that experience worthwhile for them.

The application would have many kinds of books such as Children’s Books, Novels, Magazines, Textbooks, etc. We also plan on selling school supplies and swag and we want to make this a go-to application for SJSU students. Unlike the school bookstore, we want to allow users to get supplies at a lower price.

- Stakeholders
    - SJSU Students - The targeted user base for our application are the students of the university, so they serve as the primary stakeholders in this project.
    - Team 17 - As developers of this application, we are heavily invested in this project and we want to be able to have this application used by our fellow students.
    - Professor Wu - Professor Mike Wu will be overlooking our project to completion and will be our best resource if we need anything.

- **System Environment**
    - Structure of the system

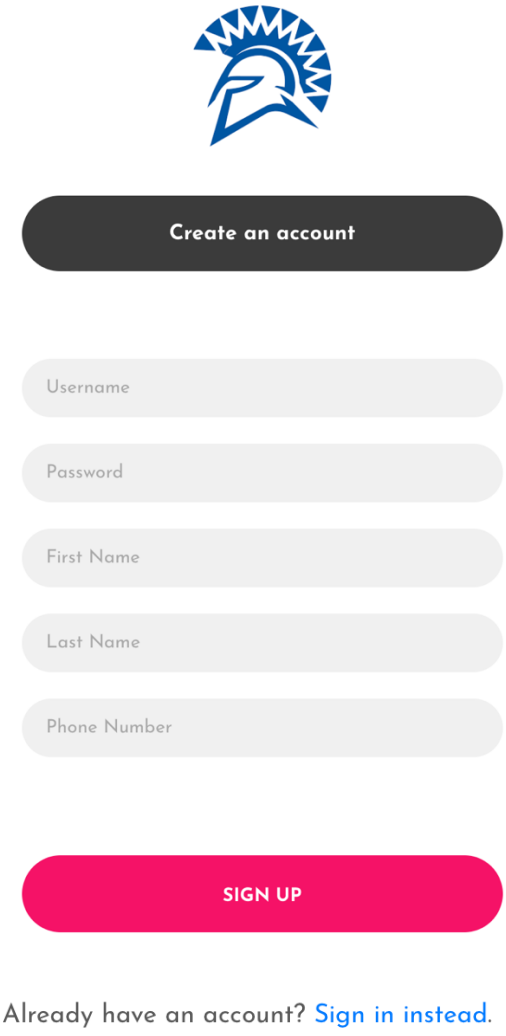


    - RDBMS used is MySQL.
    - Languages used is JavaScript.
    - Software for backend is Node.js, Expres.js server.
    - Software for frontend is Pug, HTML, CSS, Bootstrap, JavaScript.

- **Functional Requirements**
  - ◊ Describe users and how users can access your system

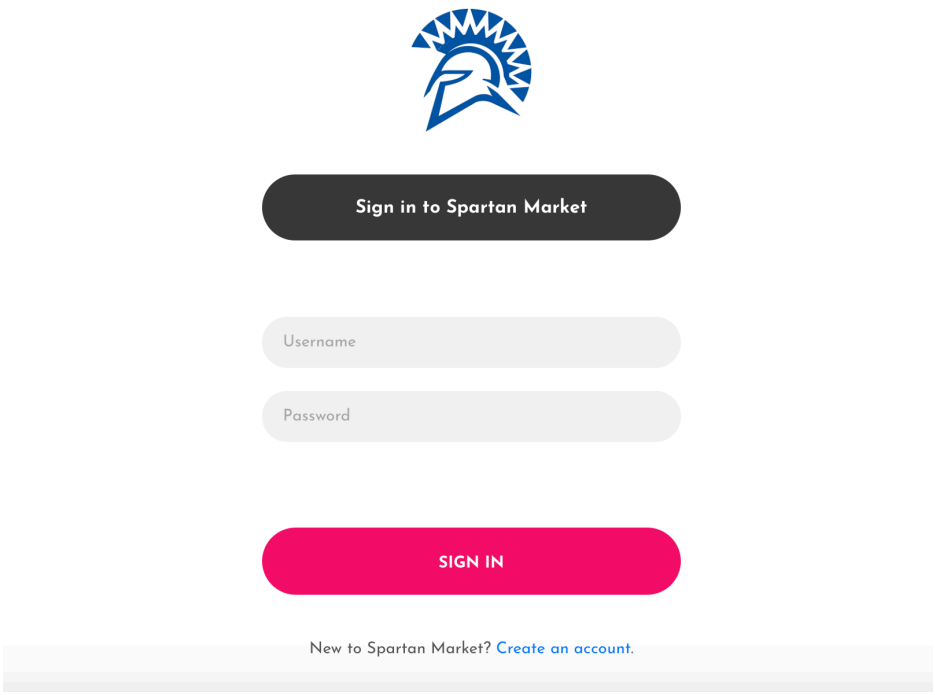
    Users could:

    - create a new account: users could create a new account to set their username, password, their first name and last name, and their phone number.

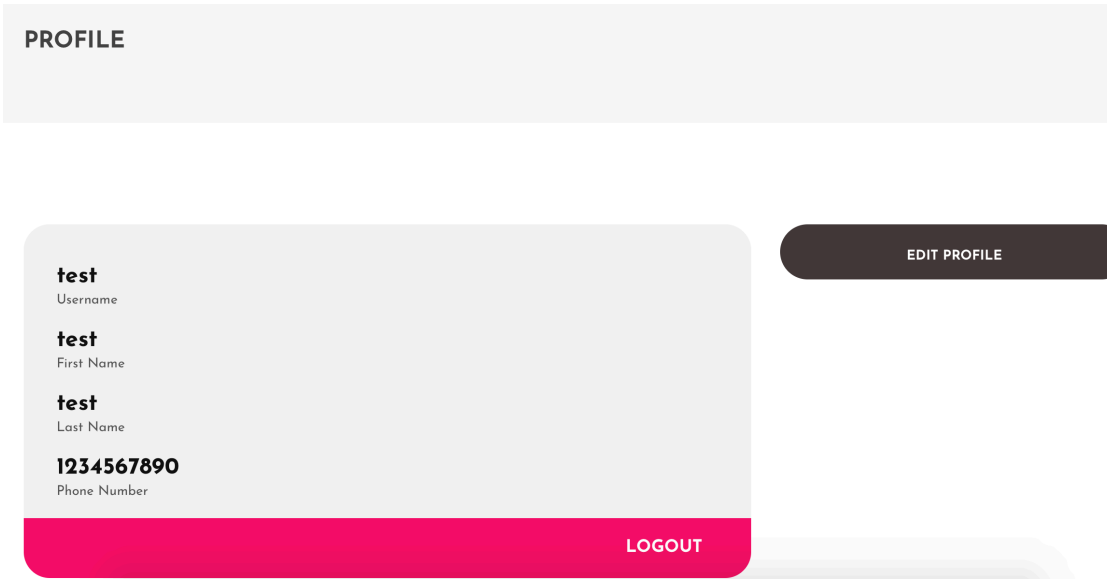


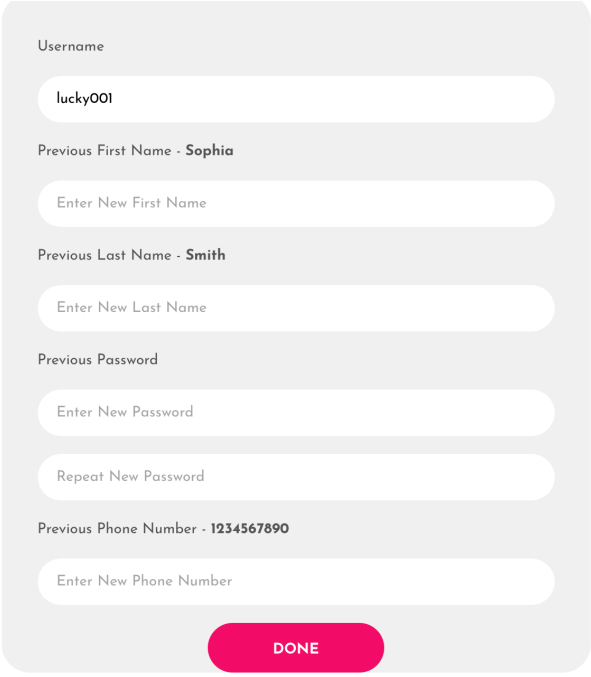
    - login using user ID and password: users can login through their username and password which they have create already.

We store the information in MySql table ‘users’.



- save user’s information: when users login, they go to profile will find there information over there. And they could change their contact information through “edit profile”. Also link to MySql table ‘users’.





- save user's cart: when customers add items to cart, the carts will store items and show number of items. Such as

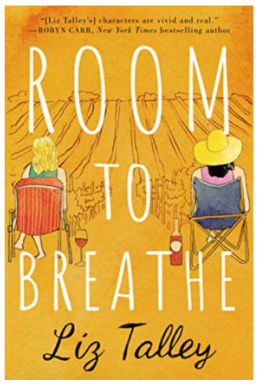
◊ Describe each functionality/features, functional processes and I/O(s).

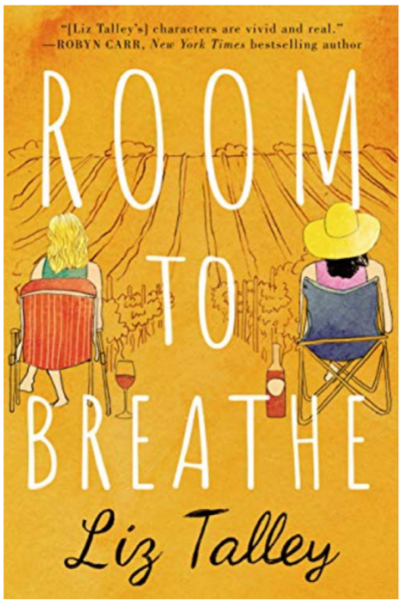
- Items
  - Category: Women, Men, Books, Stationery
  - Item ID
  - Name
  - Price
  - Description
  - Image

Spartan Market Search for an item ... Profile Log Out Shopping Cart

Home Women Men Books Stationery

BOOKS

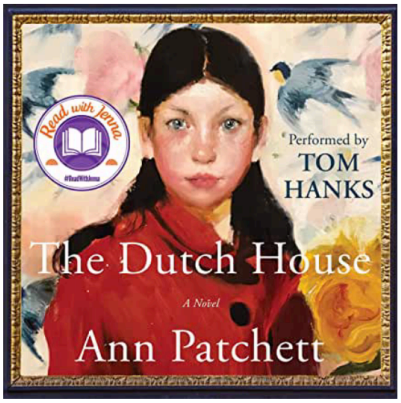




ADD TO CART

Room to Breathe **$4.99**

A funny, emotional novel full of southern charm about a mother and daughter ready to start over. Liz Talley delivers. Her dialogue is crisp and smart, her characters are vivid and real, her stories are unputdownable. —Robyn Carr, New York Times bestselling author



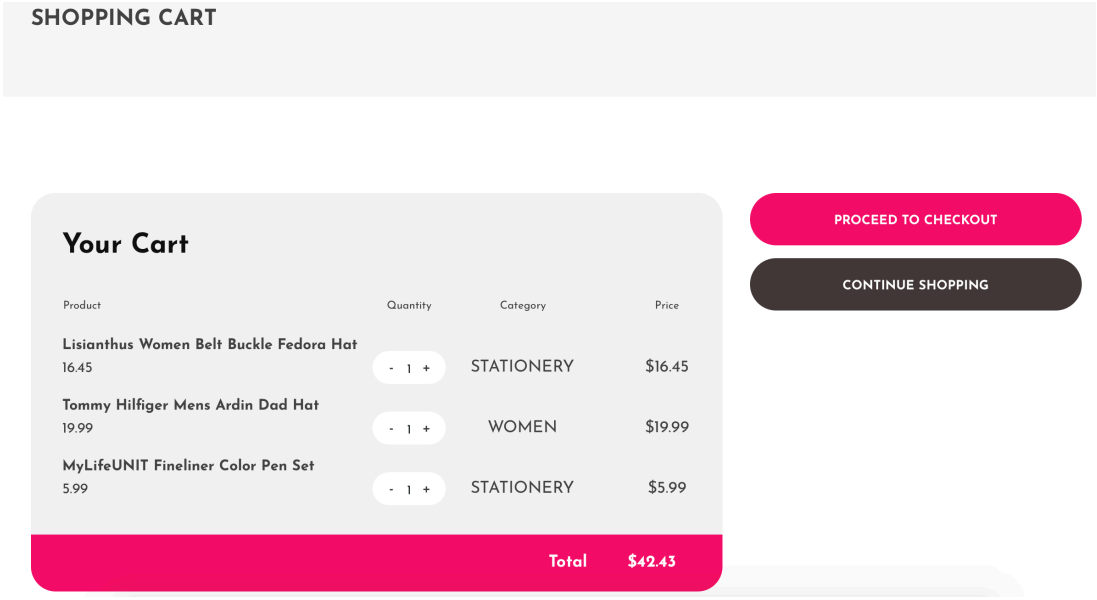
ADD TO CART

The Dutch House: A Novel **$5.99**

Audiobook performed by Tom Hanks. From the New York Times best-selling author of Commonwealth and State of Wonder comes Ann Patchett's most powerful novel to date: a richly moving story that explores the indelible bond between two siblings, the house of their childhood, and a past that will not let them go. The Dutch House is the story of a paradise lost, a tour de force that digs deeply into questions of inheritance, love, and forgiveness, of how we want to see ourselves, and of who we really are.

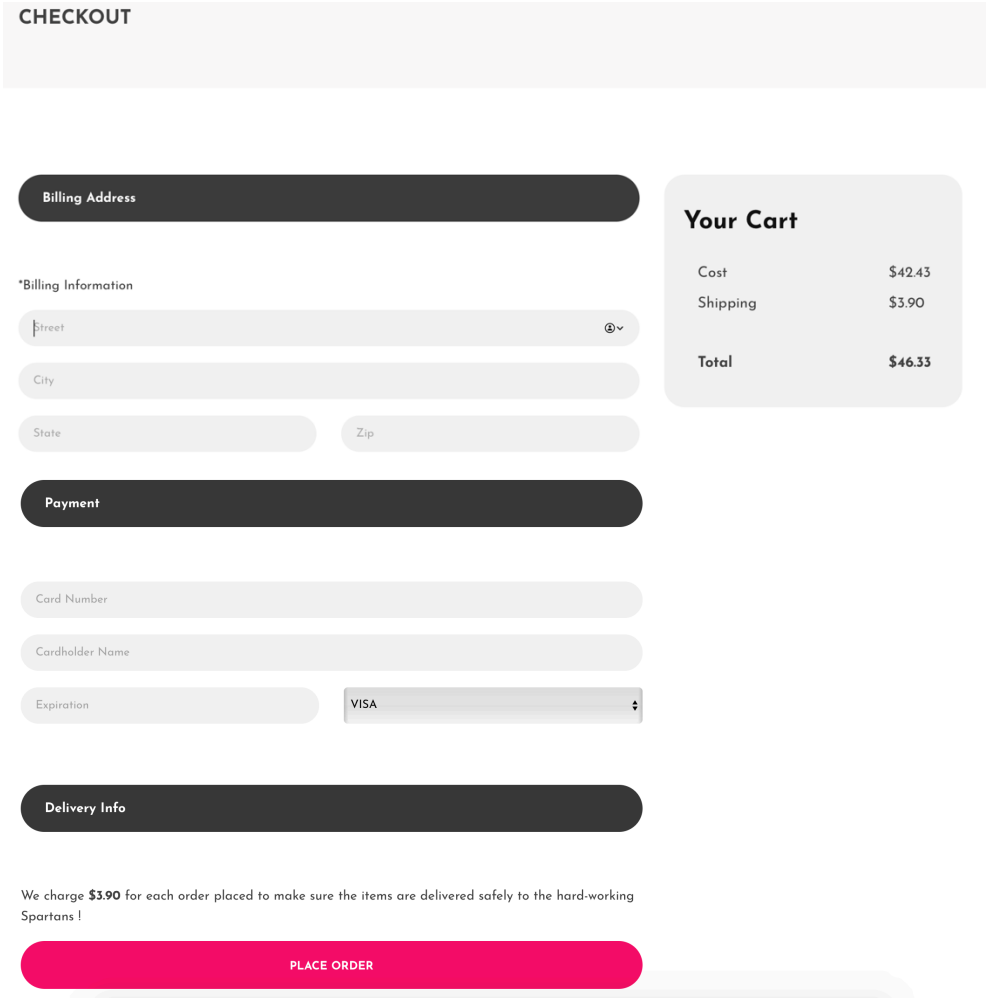
- Cart
  - Items which users choose

- The amount of each item



  - Payment page
    - Items
    - Total payment
    - Enter address to ship

- Confirm payment method



  - Contact us
    - Website information
    - Email to contact

- **Non-functional Issues**

Types of Graphical User Interface (GUI) and Software tools to build GUI

Security, access control, …

Software to build GUI

➢ Pug is a uses case sensitive syntax to generate html.

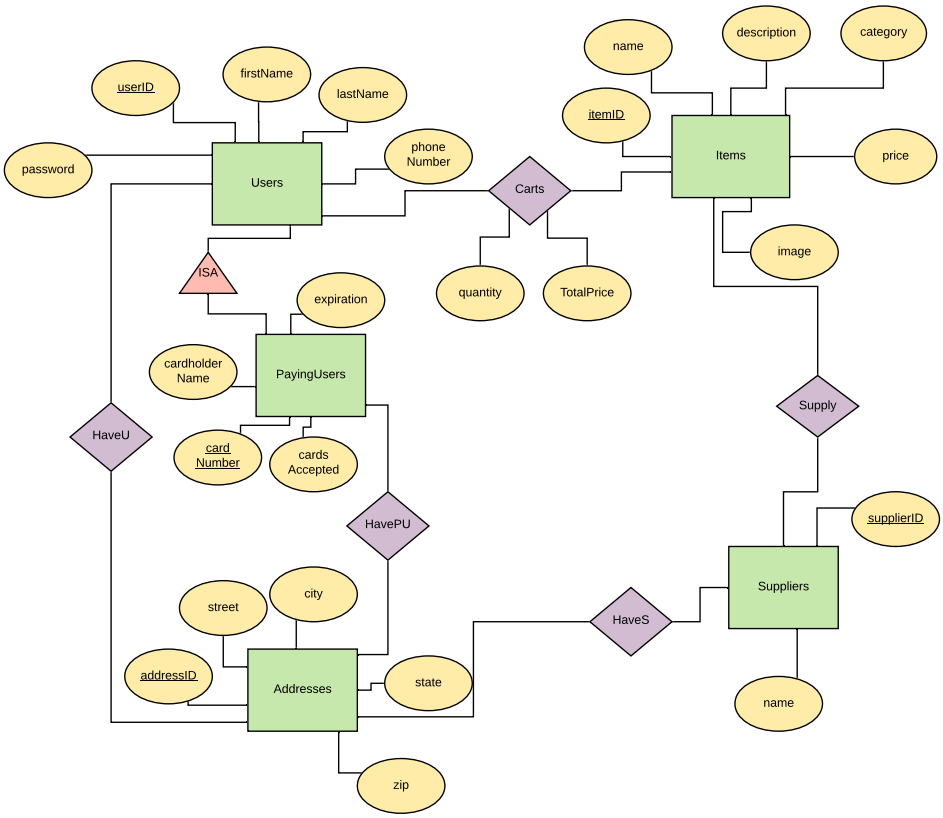
➢ JavaScript is written into an HTML page.

➢ Css describing the presentation of Web pages, including colors, layout, and fonts.

➢ Bootstrap is a framework to design websites.

➢ Colorlib template: we used template from colorlib (Joefrey) but not pages for a better and intuitive UI.

Project Design



1. Entity Sets

    a. *Users* – This set represents the ‘registered’ users in the application. By ‘registering’, the users provide us with personal data such as first name, last name, address, phone number and, set up a password for account creation.

b. *PayingUsers* – Users become PayingUsers when they are in the middle of purchasing any items off the website. These users are asked to input their credit card information like, cardholder name, card number and address. An important point to note here, is that as PayingUsers have the same attributes as Users relation along with some attributes of its own, there is an ISA relationship between them.

c. *Addresses* – This entity set collects all the addresses that are input in the website including, users', paying users' and suppliers' addresses. Each entity in Addresses has a unique ID that will be used as foreign keys for its corresponding relationships.

d. *Items* – This entity set represents all the items that are being sold in the website and have information like, name, description, stock, category, price, shipping days and is linked to a Supplier entity.

e. *Suppliers* – This has information for all the suppliers who are selling the items on the website. Each entity is identified by a primary ID key, name, type of product associated with a supplier.

2. Relationships

a. *HaveU, HavePU, HaveS* – These three relationships link a User, PayingUser and a Supplier to entities in Address, respectively. In our design, a User and a Supplier can have many addresses and

also, the same address can be used by many Users, PayingUsers and Suppliers so, this is a Many-to-Many relationship.

b. *Supply* – This relates a Supplier to an Item. An entity in Supply has information such as itemID and supplierID.

c. *Carts* – Every paying user needs to have a Cart associated with it, so the items can be put there before a purchase. An entity in Carts has information such as userID, itemID, total price and quantity.

3. Schemas

a. Users (User ID, Password, First Name, Last Name, Phone Number);

b. Items (Item ID, Name, Description, Category, Price, image);

c. Payingusers (CardNumber, CardHolderName, Expiration, CardsAccepted);

d. Addresses (addressID, Street, City, State, Zip);

e. Supplies (Supplier ID, Name);

f. Supply (itemID, supplierID);

g. HaveU (userID, addressesID);

h. HavePU (PayingUsersCardNumber, addressesID);

i. HaveS (supplierID, addressesID);

Tables

1. Addresses

```
1 • SELECT * FROM cs157a.addresses;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

AddressID	Street	City	State	Zip
1	2150 Monterey Hwy	San Jose	CA	95112
2	989 Story Rd	San Jose	CA	95122
3	979 Story Rd	San Jose	CA	95122
4	969 Story Rd	San Jose	CA	95122
5	959 Story Rd	San Jose	CA	95122
6	949 Story Rd	San Jose	CA	95122
7	939 Story Rd	San Jose	CA	95122
8	929 Story Rd	San Jose	CA	95122
9	2191 Monterey Hwy	San Jose	CA	95125
10	2131 Monterey Hwy	San Jose	CA	95112
11	2121 Monterey Hwy	San Jose	CA	95112
12	2011 Monterey Hwy	San Jose	CA	95112
13	370 Umbarger Rd	San Jose	CA	95111
14	354 Umbarger Rd	San Jose	CA	95111
15	230 Umbarger Rd	San Jose	CA	95111
16	test	test	ca	99999
17	tasman	sanjose	ca	98888
18	redwood	sanjose	ca	98888

Result Grid Form Editor Field Types Query Stats

2. Carts

```
1 • SELECT * FROM cs157a.carts;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

UserID	ItemID	Quantity	TotalPrice
lucky001	1111	1	10.99
lucky001	1114	1	4.99
lucky001	1122	1	33.15
lucky002	1112	1	2.99
lucky003	1113	1	1.99
lucky004	1114	1	4.99
lucky005	1115	1	6.99
lucky006	1116	10	79.9
lucky007	1117	10	109.9
lucky008	1118	10	39.9
lucky009	1119	100	599
lucky010	1120	10	164.5
lucky011	1121	1	19.99
lucky012	1122	1	33.15
lucky013	1123	1	14.99
lucky014	1124	1	10.99
lucky015	1125	1	19.99
NULL	NULL	NULL	NULL

Result Grid Form Editor Field Types Query Stats

3. Havepu

```
1 •  SELECT * FROM cs157a.havepu;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

CardNumber	AddressID
1111	3
1234	1
1345	8
1567	13
1789	12
2222	4
2345	7
3123	2
3333	5
4444	10
4567	6
5555	15
6666	14
7777	9
7890	11
NULL	NULL

Result Grid Form Editor Field Types Query Stats

4. haveS

```
1 •  SELECT * FROM cs157a.haveS;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

SupplierID	AddressID
11111	15
11112	14
11113	13
11114	12
11115	10
11116	9
11117	8
11118	7
11119	5
11120	11
11121	6
11122	4
11123	3
11124	2
11125	1
NULL	NULL

Result Grid Form Editor Field Types Query Stats

5. haveu

```
1 • SELECT * FROM cs157a.haveu;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

UserID	AddressID
lucky001	1
lucky002	2
lucky003	3
lucky004	4
lucky005	5
lucky006	6
lucky007	7
lucky008	8
lucky009	9
lucky010	10
lucky011	11
lucky012	12
lucky013	13
lucky014	14
lucky015	15
NULL	NULL

Result Grid | Form Editor | Field Types | Query Stats

6. items

```
1 • SELECT * FROM cs157a.items;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

ItemID	Name	Description	Image	Category	Price
1111	Cut and Run	Twin sisters separated by the past are reunited…	1.png	BOOK	10.99
1112	My big Fat Fake Wedding	Now a Washington Post and Amazon Charts Be…	2.png	BOOK	2.99
1113	Winter Cottage	An Amazon Charts bestseller. A gripping novel a…	3.png	BOOK	1.99
1114	Room to Breathe	A funny, emotional novel full of southern charm…	4.png	BOOK	4.99
1115	The Vine Witch	A young witch emerges from a curse to find her…	5.png	BOOK	6.99
1116	Under Lying	In this gripping novel of suspense, the disappea…	6.png	BOOK	7.99
1117	Pour Judgment	A week of tits, booze, and fun in the sun? Wher…	7.png	BOOK	10.99
1118	Hello, Darkness	From the #1 New York Times bestselling author…	8.png	BOOK	3.99
1119	The Dutch House: A Novel	Audiobook performed by Tom Hanks. From the…	9.png	BOOK	5.99
1120	Lisianthus Women Belt Buckle Fedora Hat	Imported Material: 65% cotton, 35% polyester A…	10.png	WOMEN	16.45
1121	Tommy Hilfiger Mens Ardin Dad Hat	100% Cotton Imported Adjustable closure Hand…	11.png	MEN	19.99
1122	Kangol Unisex Tropic 504 Ventair	100% Polyester Made in the USA and Imported…	12.png	WOMEN	33.15
1123	Carhartt Womens Odessa Cap	100% Cotton Imported Hook and Loop closure…	13.png	WOMEN	14.99
1124	Funky Junque Solid Ribbed Beanie Slouc…	UNISEX: Great for both women and men. The c…	14.png	WOMEN	10.99
1125	Tommy Hilfiger Mens Dad Hat Avery	100% Cotton Imported Adjustable closure Hand…	15.png	MEN	19.99
1126	MyLifeUNIT Fineliner Color Pen Set	0.4mm Colored Fine Liner Sketch Drawing Pen,…	16.png	STATIO…	5.99
1127	3C4G 36015 Best Day Ever Super Station…	All you need for doodling, drawing and writing in…	17.png	STATIO…	16.62
1128	AmazonBasics Daily Planner and Journal…	Stay on task and organized with daily, weekly a…	18.png	STATIO…	16.62
1129	Blank Note Cards with Envelopes	Flat Note Cards, 4x6 Blank Back Note Cards, A…	19.png	STATIO…	12.99
1130	Soucolor Super Large Capacity Canvas P…	DIMENSIONS: 19.8x8.6x4.5cm(L*W*T). Lots of…	20.png	STATIO…	6.99
NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

7. payingusers

```
1 • SELECT * FROM cs157a.payingusers;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

CardNumber	CardHolderName	Expiration	CardsAccepted
1111	Emma Williams	2020-10	VISA
1234	Sophia Smith	2020-10	VISA
1345	Avery Wilson	2020-10	VISA
1567	Leah Baker	2020-10	VISA
1789	Zoey Martin	2020-10	VISA
2222	Mia Jones	2020-10	VISA
2345	Grace Miller	2020-10	VISA
3123	Jacob Johnson	2020-10	VISA
3333	Addison Brown	2020-10	VISA
4444	Aubrey White	2020-10	VISA
4567	Lily Davis	2020-10	VISA
5555	Ashley Cooper	2020-10	VISA
6666	Anna Adams	2020-10	VISA
7777	Sofia Thomas	2020-10	VISA
7890	Lillian Harris	2020-10-12	VISA
NULL	NULL	NULL	NULL

Result Grid Form Editor Field Types Query Stats

8. suppliers

```
1 • SELECT * FROM cs157a.suppliers;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

SupplierID	Name
11111	Tincidunt Limited
11112	Nullam Vitae Diam Institute
11113	Natoque Limited
11114	Nam LLP
11115	Lorem Idsum Associates
11116	A Tortor Nunc Company
11117	Accumsan Laoreet LLP
11118	Non Foundation
11119	Molex Inc.
11120	NEC Corporation
11121	Prent Corporation
11122	NVIDIA Coporation
11123	Multek Corporation
11124	Nan Ya Printed Circuit Boar...
11125	Nippon Mektron, Ltd.
NULL	NULL

Result Grid Form Editor Field Types Query Stats

9. supply

```
1 • SELECT * FROM cs157a.supply;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

ItemID	SupplierID
1111	11111
1112	11112
1113	11113
1114	11114
1115	11115
1116	11116
1117	11117
1118	11118
1119	11119
1120	11120
1121	11121
1122	11122
1123	11123
1124	11124
1125	11125
NULL	NULL

Result Grid Form Editor Field Types Query Stats

10. users

```
1 • SELECT * FROM cs157a.users;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

UserID	Password	FirstName	LastName	PhoneNumber
lucky001	qwert1234	Sophia	Smith	1234567890
lucky002	yuiop1234	Jacob	Johnson	1234567891
lucky003	asdfg5678	Emma	Williams	1234567892
lucky004	hjklm5678	Mia	Jones	1234567893
lucky005	qwert1234	Addison	Brown	1234567894
lucky006	yuiop1234	Lily	Davis	1234567895
lucky007	asdfg5678	Grace	Miller	1234567896
lucky008	yuiop1234	Avery	Wilson	1234567897
lucky009	asdfg5678	Sofia	Thomas	1234567898
lucky010	yuiop1234	Aubrey	White	1234567899
lucky011	asdfg5678	Lillian	Harris	1234567880
lucky012	hjklm5678	Zoey	Martin	1234567881
lucky013	hjklm5678	Leah	Baker	1234567882
lucky014	yuiop1234	Anna	Adams	1234567883
lucky015	hjklm5678	Ashley	Cooper	1234567884
test	test123	test	test	1234567890
tien	tien	Tien	Ly	4086678102
NULL	NULL	NULL	NULL	NULL

Result Grid Form Editor Field Types Query Stats

The tables are in BCNF or 3NF.

Implementation

- Functions associated with query:
    - Home page shows all items from mysql.

```
//Home Page
router.get('/', checkauthorization, function (req, res, next) {
  var sql = "SELECT * FROM items";
  var sql1 = "SELECT SUM(Quantity) as iic FROM carts WHERE UserID = '" + req.session.userId + "'";

  db.query(sql1, function (error, result1, fields) {
    db.query(sql, function (error, result2, fields) {
      res.render('home', { items: result2, itemincart: result1[0].iic });
    });
  });
});
```

    - Men page shows men's stuff from mysql, SELECT * FROM items WHERE Category = 'MEN'

```
router.get('/men', checkauthorization, function (req, res, next) {
  var sql = "SELECT * FROM items WHERE Category = 'MEN'";
  var sql1 = "SELECT SUM(Quantity) as iic FROM carts WHERE UserID = '" + req.session.userId + "'";

  db.query(sql1, function (error, result1, fields) {
    db.query(sql, function (error, result2, fields) {
      res.render('page', { titlename: 'Men', name: 'MEN\'S APPAREL', items: result2, itemincart: result1[0].iic });
    });
  });
});
```

    - Women page shows women's stuff from mysql, SELECT * FROM items WHERE Category = 'WOMEN'

```
router.get('/women', checkauthorization, function (req, res, next) {
  var sql = "SELECT * FROM items WHERE Category = 'WOMEN'";
  var sql1 = "SELECT SUM(Quantity) as iic FROM carts WHERE UserID = '" + req.session.userId + "'";

  db.query(sql1, function (error, result1, fields) {
    db.query(sql, function (error, result2, fields) {
      res.render('page', { titlename: 'Women', name: 'WOMEN\'S APPAREL', items: result2, itemincart: result1[0].iic });
    });
  });
});
```

    - Books page shows books from mysql, SELECT * FROM items WHERE Category = 'BOOK'

```
router.get('/books', checkauthorization, function (req, res, next) {
  var sql = "SELECT * FROM items WHERE Category = 'BOOK'";
  var sql1 = "SELECT SUM(Quantity) as iic FROM carts WHERE UserID = '" + req.session.userId + "'";

  db.query(sql1, function (error, result1, fields) {
    db.query(sql, function (error, result2, fields) {
      res.render('page', { titlename: 'Books', name: 'BOOKS', items: result2, itemincart: result1[0].iic });
    });
  });
});
```

- Stationery page shows stationeries from mysql, SELECT * FROM items WHERE Category = ‘STATIONERY’

```
router.get('/stationery', checkauthorization, function (req, res, next) {
  var sql = "SELECT * FROM items WHERE Category = 'STATIONERY'";
  var sql1 = "SELECT SUM(Quantity) as iic FROM carts WHERE UserID = '" + req.session.userId + "'";

  db.query(sql1, function (error, result1, fields) {
    db.query(sql, function (error, result2, fields) {
      res.render('page', { titlename: 'Stationery', name: 'STATIONERY', items: result2, itemincart: result1[0].iic });
    });
  });
});
```

- Search using key words of items’ name, SELECT * FROM items WHERE Name LIKE

```
router.post('/search', checkauthorization, function (req, res, next) {
  var sql = "SELECT * FROM items WHERE Name LIKE '%" + req.body.search + "%'";
  var sql1 = "SELECT SUM(Quantity) as iic FROM carts WHERE UserID = '" + req.session.userId + "'";

  db.query(sql1, function (error, result1, fields) {
    db.query(sql, function (error, result2, fields) {
      res.render('page', { titlename: 'Search', name: 'Result for ' + req.body.search, items: result2, itemincart: result1[0].iic });
    });
  });
});
```

- Functions associated with insertion:
  - Registration: insert users’ information to mysql.

```
//Registration
router.get('/registration', checkunauthorization, function (req, res, next) {
  res.render('registration');
});

router.post('/signup', checkunauthorization, function (req, res, next) {
  var users = {
    "UserID": req.body.userid,
    "FirstName": req.body.firstname,
    "LastName": req.body.lastname,
    "Password": req.body.password,
    "PhoneNumber": req.body.phonenumber,
  }
  var sql = "SELECT * FROM users WHERE UserID = '" + users.UserID + "'";

  db.query(sql, function (error, results, fields) {
    if (results.length) {
      res.render('registration', { message: "User ID already exists!" });
    } else {
      db.query('INSERT INTO users SET ?', users, function (error, results, fields) {
        res.render('landing', { message: "Succesfully! Your account has been created." });
      });
    }
  });
});
```

- Functions associated with updating:

- Cart button decrease: users can click the button to minus amount of the products.

```
//cart button decrease
router.get('/dec/:id', checkauthorization, function (req, res, next) {
  var sql = "SELECT Price FROM items WHERE ItemID = '" + req.params.id + "'";
  var sql1 = "SELECT items.ItemID, Name, Price, Category, Quantity, TotalPrice FROM items, carts WHERE carts.ItemID = '"
  + req.params.id + "' AND UserID = '" + req.session.userId + "' AND items.ItemID = carts.ItemID";

  db.query(sql, function (error, result, fields) {
    db.query(sql1, function (error, result1, fields) {
      if (result1[0].Quantity > 1) {
        var temp1 = result1[0].Quantity - 1, temp2 = result1[0].TotalPrice - result[0].Price;
        var sql2 = "UPDATE carts SET Quantity='" + temp1 + "', TotalPrice='" + temp2 + "' WHERE UserID='" + req.session.userId
        + "' AND ItemID='" + req.params.id + "'";

        db.query(sql2, function (error, result2, fields) {
          res.redirect('back');
        });
      } else {
        var sql2 = "DELETE FROM carts WHERE UserID='" + req.session.userId + "' AND ItemID='" + req.params.id + "'";

        db.query(sql2, function (error, result2, fields) {
          res.redirect('back');
        });
      }
    });
  });
});
```

- Cart button increase: users can click the button to add amount of the products.

```
//cart button increase
router.get('/inc/:id', checkauthorization, function (req, res, next) {
  var sql = "SELECT Price FROM items WHERE ItemID = '" + req.params.id + "'";
  var sql1 = "SELECT items.ItemID, Name, Price, Category, Quantity, TotalPrice FROM items, carts WHERE carts.ItemID = '"
  + req.params.id + "' AND UserID = '" + req.session.userId + "' AND items.ItemID = carts.ItemID";

  db.query(sql, function (error, result, fields) {
    db.query(sql1, function (error, result1, fields) {
      var temp1 = result1[0].Quantity + 1, temp2 = result1[0].TotalPrice + result[0].Price;
      var sql2 = "UPDATE carts SET Quantity='" + temp1 + "', TotalPrice='" + temp2 + "' WHERE UserID='"
      + req.session.userId + "' AND ItemID='" + req.params.id + "'";

      db.query(sql2, function (error, result2, fields) {
        res.redirect('back');
      });
    });
  });
});
```

- Profile update: users can update their information through edit profile.

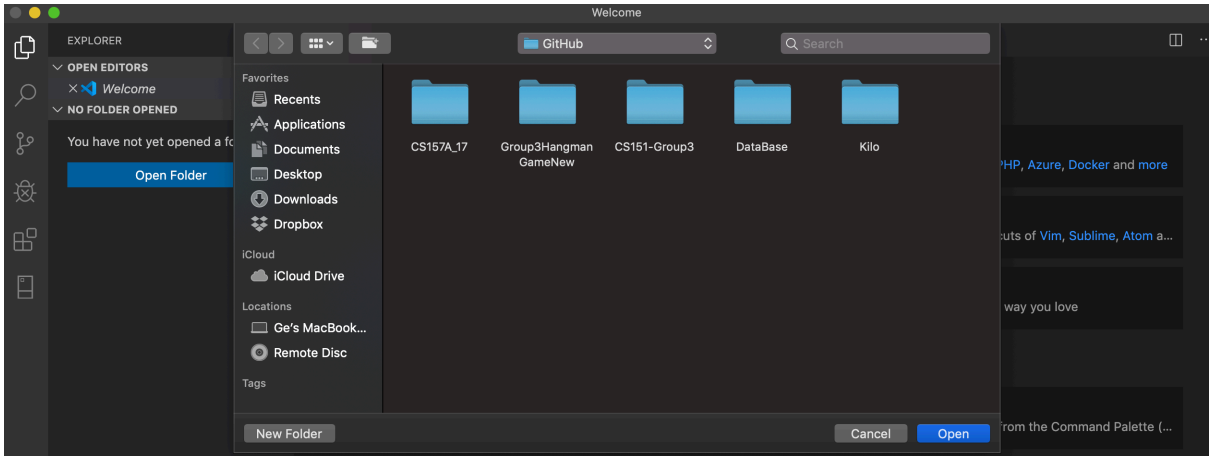
```
router.get('/editprofile', checkauthorization, function(req, res, next) {
  console.log(req.session);
  var sql =
    "SELECT SUM(Quantity) as iic FROM carts WHERE UserID = '" +
    req.session.userId +
    "'";

  db.query(sql, function(error, results, fields) {
    res.render('editprofile', {
      user: req.session.user,
      itemincart: results[0].iic
    });
  });
});
```

- Functions associated with deleting:
  - Delete the cart when users have checkouted.

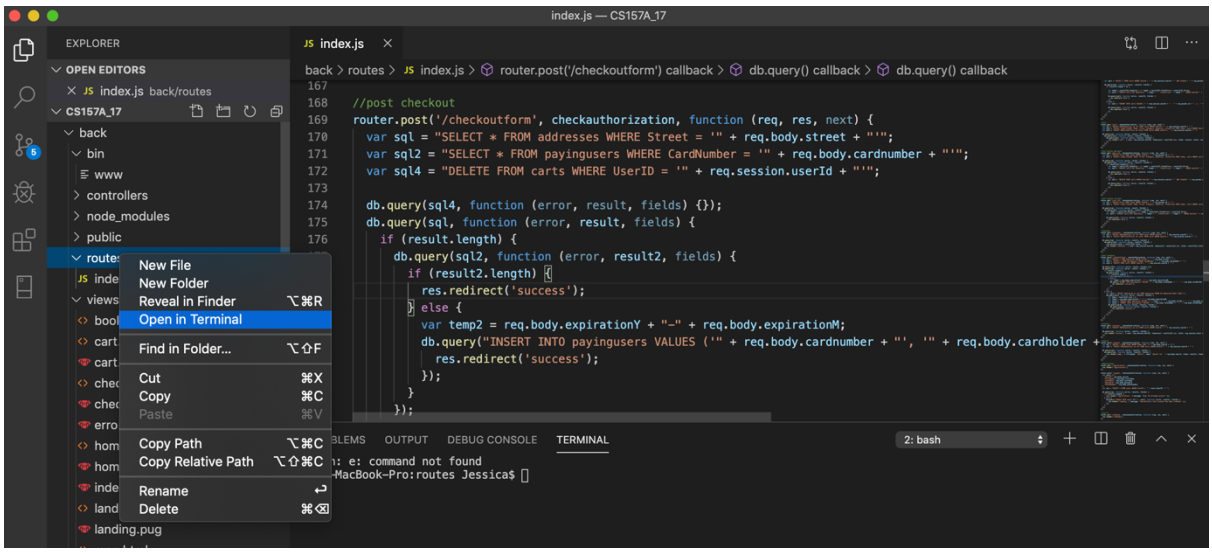
```
router.post('/checkoutform', checkauthorization, function (req, res, next) {
  var sql = "SELECT * FROM addresses WHERE Street = '" + req.body.street + "'";
  var sql2 = "SELECT * FROM payingusers WHERE CardNumber = '" + req.body.cardnumber + "'";
  var sql4 = "DELETE FROM carts WHERE UserID = '" + req.session.userId + "'";
```

Procedures of how to set up and run the system:

1. Open folder by using VSC (Visual Studio Code)



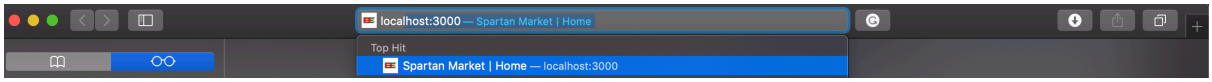
2. Open routes in Terminal



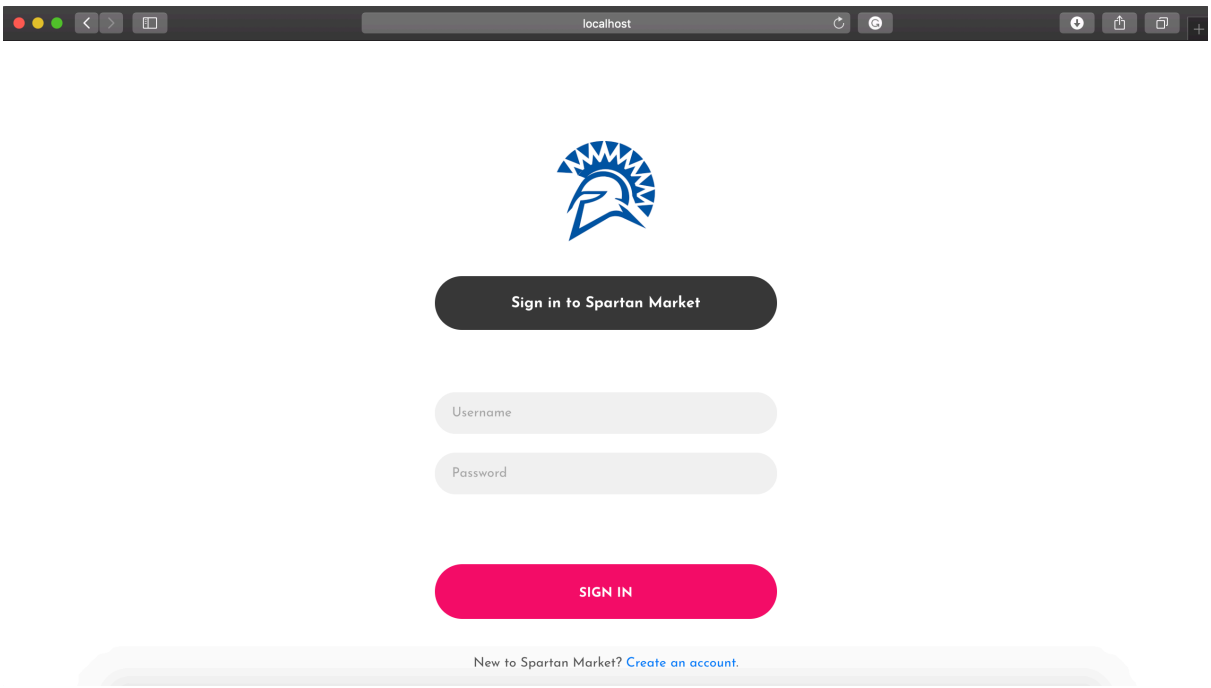
3. npm install
4. Run Database.sql file in Workbench.
5. npm start from ‘back’ directory: Database is connected.

```
Ges-MacBook-Pro:routes Jessica$ npm start

> cs157a-17@0.0.0 start /Users/Jessica/Documents/GitHub/CS157A_17/back
> nodemon ./bin/www

[nodemon] 2.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www`
Database is connected ...
```

6. Open the browser, using localhost:3000 to find the window.



7. It’s ready to use.



Conclusion

- Ge Ou:
    - Learned from the DB project: At first, I have learned 3-tiered architecture, which connect the whole project from front end, back end, and database. Second, I have learned ERD from the project design. Such as identify the entities, weak entity, attributes, relationships, inheritance, etc. Third, I familiar with JavaScript language through the project, and using html design pages.
    - Future improvement of our DB application: users can choose their stored billing information, card information, and delivery information from. We can open our store to users to sell their unused goods. Tables that users can insert items and belong to USED-attribute. Then users can add and delete their items.
- Tien Ly:
    - This is the first time I work and study about NodeJS also JavaScript. From other class, I knew about language Java, PHP,

HTML, CSS but not NodeJS. From this project, I know how the session works, how to set up route, view, and database for NodeJS and MySQL. I also learn more things from bootstrap to set up template for front end. Pug engine is a good program for front end. It is simpler than html and has easier syntax. We can code directly mixing front end and back end data.

  - I learned about how to connect NodeJS with database MySQL and query to get, update, store, delete information in database. It was hard when I started but when I understand it, it is easy.
  - The project has many points to improve. The password needs to hash for security. User account needs the history of order/item they have purchased, needs pages to edit their information and change password, needs page to enter their address. The project needs function forgot password to look password for user. Higher improve: creating admin account to control users account also the items in the shop. Admin account should be able to add, delete, change the items information.

- Maan Singh:
  - I am really happy with our project and it is a great achievement for me that I can put on my resume! It was a great time learning and implementing the project as I believe the best way to learn coding is to work on a project at the same time and this project did the same for me. I would like to thank my amazing teammates Tien Ly and Ge Ou who I had a pleasure working with and of course, our great professor Mike Wu who I have learned many things from!
  - I have usually worked on backend for my previous projects so I decided to take over front-end for this project. I learned Bootstrap, CSS, HTML and Pug.js while working on the front-end and I

cannot be prouder of what I have learned!

- As I was done with the front-end while my teammates were still implementing, I decided to help them and on the back-end and as this was my first time using mysql with node.js, it was really fun to learn and see how the changes from the front-end are affecting the database! I wrote a few routes for the pages and from it I learned how important it is to use the query in Workbench and see what is really going on under the hood for debugging purposes and also the importance of the type, format and size expectation from the data!
- Improvements: Our project could be improved in various ways. The password could be hashed for safety and we could have a forget password feature. Also, we do save address and card info of a user when a user is making a purchase, but that data is never used later. We could have the UI show and let the user select saved information, so the user does not have to type the same information over and over again! Also, as we keep using the application, we keep on adding data and database becomes encumbered with data which could cause the application to run slow while using queries, so we could have an admin account that can take care of deletion through the front-end. Also, it will be useful if the users could also sell items!

Works Cited

Smith, Felecia. 10 Benefits Of Reading. Technobezz, 14 Augest. 2018. https://www.technobezz.com/10-benefits-of-reading/

Joefrey. “Divisima - Free Clothing Website Template 2019.” Colorlib, 15 Jan. 2019, https://colorlib.com/wp/template/divisima/?fbclid=IwAR2-8TWoZc4rQ31ynGpdQWeXyM5qdfpRjHcwGZ4mHF29EkLlS3Nq32rMu3k.