Team 2 Documentation

Downloads: MongoDB and Stripe

**11/19/2020**

To do:

1. Add Storage component to the website.
   1. User gives information and date of **Pick Up**
   2. Item is then stored, and insurance can be added
   3. Customer then pays for the storage
   4. If item is not picked up by Pick Up time, then it is auctioned off
2. Create Schema Tables based off MongoDB
3. Document Site flow
4. About us page needs information

**11/19/2020** – (Notes)

Security for user authentication is done using cookies.

Database security is done using Manager/Admin server authentication.

Users are identified through Email, Phone number and Username

Errors –

1. Tracking Number Error (unknown tracking numbers do not throw error)
2. Kiosk Dashboard does not work for some accounts

System Requirement Specifications

**I Introduction**

**1.1 Project Purpose**

The purpose of this product is to provide customers with a solution to handle items airports will not allow. We want to guarantee safety and quickness when handling items.

**1.2 Document Conventions**

|  |
| --- |
| DB                                                                       Database |
| DDB                                                                    Distributed Database |
| SM                                                                       Site Map |

**1.3 Intended Audience**

This project is intended for customers at specified airports. This will be implemented by the stakeholders for customer use. This system will be useful for anyone within the airport.

**1.4 Project Scope**

The purpose of the shipping and storage system is to give customers an easier time when it comes to flying. The system will handle the occasions where customers may lose an item due to a TSA ban and ensure it’s safety until it is back in the customer's care. This will also help airports during the scanning process because it will allow customers to have items on the no flight list shipped or stored beforehand.

**II Description**

**2.1 Product Perspective**

The shipping and storage system will store information such as:

**Customer Information:**

This will include customer shipping information as well as items shipped or stored with the company using the airport kiosk.

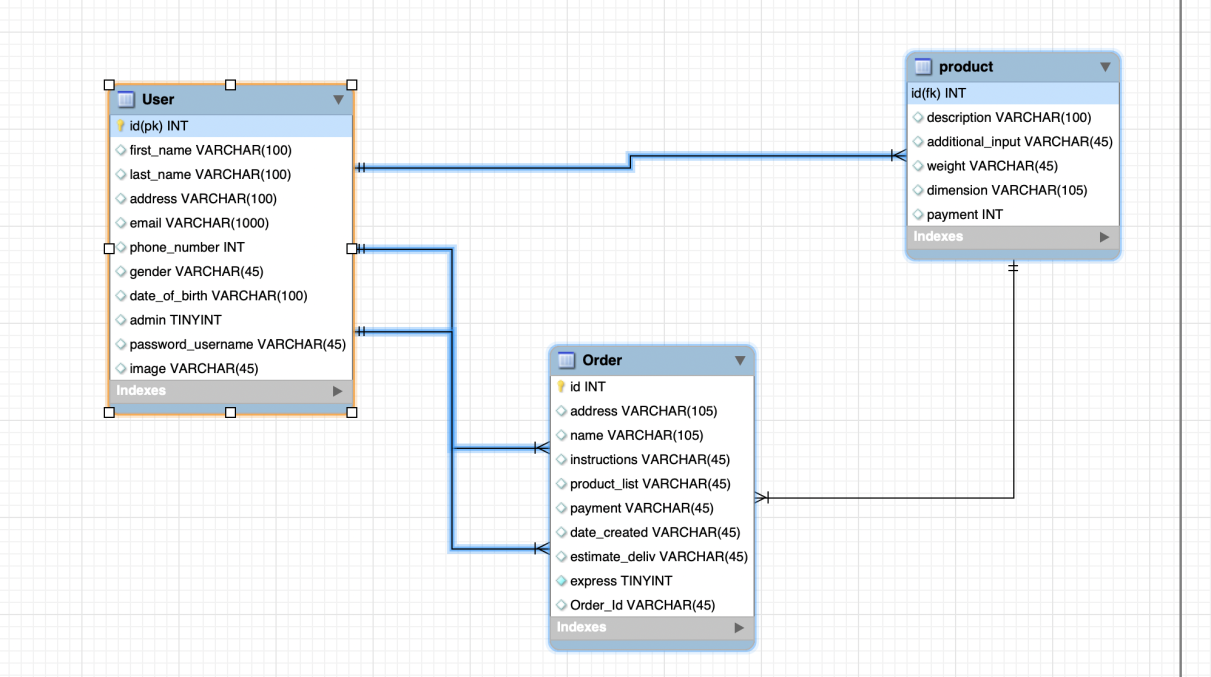
**Order Information:**

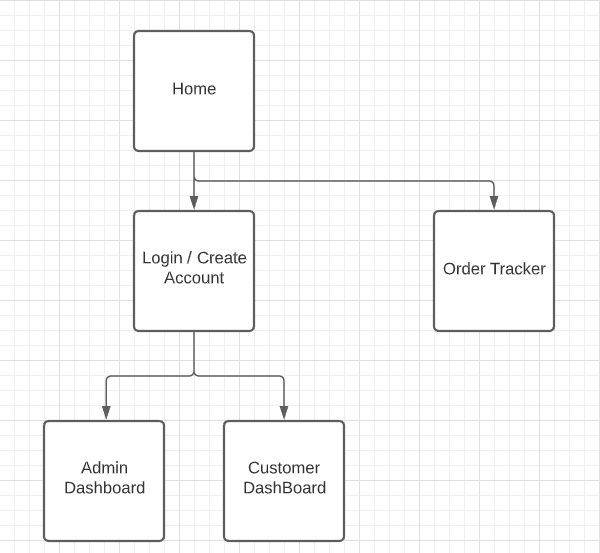
This will include customer orders as well as tracking for up to date information on a customer’s item arrival time.

**Storage Information:**

This will include the items placed into storage, dates in which the item can be held as well as payments for storage use.

**2.2 Product Features**

****

****

**2.3 User Class and Characteristics**

Customers using the system will be able to manage their orders on items shipped and stored and make payments on storage and shipping. The user should be able to track their order or storage status while it is in the care of the company. The system includes the following functions:

Customer Functions:

* Track orders using auto generated tracking number
* Make payments on items through system
* Cancel an item shipping or storage
* Create a support ticket

Employee Functions:

* Create an order for customers
* Track customer orders using the auto generated tracking number
* Cancel or modify existing orders for customers
* Handle support tickets processed

Admin Functions:

* Add/Delete airport kiosk location
* Onboard/Offboard incoming employees
* Update pricing for shipping and storage
* Create/Modify shipping and storage conditions

**2.4 Operating Environment**

Database: MongoDB

Platform: React/Javascript

**2.5 Design and Implementation Constraints**

There were no constraints placed by the stakeholder, but we remained on HTML/CSS,NodeJs MangoDB, Javascript, and React.

**III Interface Requirements**

**3.1 User Interface**

Front-end software: HTML/CSS

Back-end software: NodeJs MongoDB, Javascript

**3.2 Hardware Interface**

Any web browser or the airport kiosk

**3.3 Software Interface**

HTML/CSS was used due to its versatility among multiple web devices.

MongoDB was used due to the programming language being in javascript.

**IV Nonfunctional Requirements**

**4.1 Performance Requirements**

**4.2 Security Requirements**

User schema:--------------------------------------------------------------------------------------------------------------------------------------------------

new Schema({

userId: {

type: String,

required: true,

unique: true,

},

username: String,

firstName: { type: String },

lastName: { type: String },

createdAt: { type: Date, required: true },

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

phone: String,

address: String,

city: String,

zip: String,

apt: String,

phone: { type: String, required: true, },

isAdmin: { type: Boolean, default: false },

role: { type: String, enum: ["admin", "user", "kiosk clerk", "staff", ], default: 'user' },

permission: { type: Number, require: true, default: 0 }, //0 = user

status: Boolean,

avatar: String,

airportLocation: {

airportName: String, airportAddress: String,

airportZip: String,

airportCity: String,

airportCountry: String,

}

})

Package Schema:----------------------------------------------------------------------------------------------------------------------------------------------

new Schema({

createdAt: { type: Date },

orderId: { type: String },

trackingId: { type: String },

packages: [{

packageName: { type: String },

package\_name: { type: String },

no\_of\_packages: { type: String },

weight\_per\_package: String,

dimension\_per\_package: String

}, ],

senderFirstName: { type: String },

senderLastName: { type: String },

senderPhone: { type: String },

senderEmail: { type: String },

airportLocation: {

airportName: String,

airportAddress: String,

airportZip: String,

airportCity: String,

airportCountry: String,

},

processedBy: { type: String },

receiverFirstName: { type: String },

receiverLastName: { type: String },

receiverEmail: { type: String },

receiverPhone: { type: String },

receiverAddress: { type: String },

receiverCity: { type: String },

receiverApt: { type: String },

receiverZip: { type: String },

receiverCountry: { type: String },

courierCompany: { type: String },

totalCost: { type: Number },

stripeCharge: {

id: String,

amount: Number,

created: Number,

livemode: Boolean,

paid: Boolean,

status: String,

},

payBy: { type: String },

packageProcessed: { date: Date, isProcessed: Boolean },

packageShipped: { date: Date, isShipped: Boolean },

packageDelivered: { date: Date, isDelivered: Boolean },

})

Location Schema:-----------------------------------------------------------------------------------------------------------------------------

new Schema({

name:String,

address:String,

zipCode:String,

state:String,

country:String,

city:String,

})

class LocationClass{

static async getAllLocations(){

const locations=await this.find({});

if(!locations){

return []

}

return locations;

}

static async addLocation(location){

// get the modified locations list;

try{

console.log(location)

var location=await this.create(location);

console.log('----------------create location')

}catch(err){

console.log(err.message)

throw new Error(err.message)

}

return {status:'ok'}

}

static async deleteLocation(location){

try{

var location=await this.deleteOne({name:location.name,country:location.country})

}catch(err){

console.log(err.message);

throw new Error(err.message)

}

return {status:'ok'}

}

}