**Introduction**

**Author Name: Rahul Nathu Muthe**

**College: Pimpri Chinchwad College of Engineering and Research Ravet Pune**

**Mobile No: 7768862452**

**Email:** [**rahul.muthe10@gmail.com**](mailto:rahul.muthe10@gmail.com)

**Project Introduction**

**Project Title:** Inventory Management System for Retailers

**Overview:** Retail inventory management is the process of ensuring you carry merchandise that shoppers want, with neither too little nor too much on hand. By managing inventory, retailers meet customer demand without running out of stock or carrying excess supply.

In practice, effective retail inventory management results in lower costs and a better understanding of sales patterns. Retail inventory management tools and methods give retailers more information with which to run their businesses. Applications have been developed to help retailers to track and manage stocks related to their own products. The System will ask retailers to create their account by providing essential details. Retailers can access their accounts by logging to the application.

Once retailers successfully login to the application they can update their inventory details, also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The System will automatically send an email alert to the retailers if there is no stock found in their account. So that they can order new stock.

**Purpose:** The main purpose of Inventory Management system is to manage an inventory, customer, product, invoices and reports. It manages all information about inventory. The project totally builds at administrative end and thus only administer guaranteed access. The purpose of project is to build an application program to reduce manual work for managing inventory.

**Literature Survey**

**Existing System of Inventory Management System:**

In the existing system the exams are done only manually but in proposed system we have to computerize the exams using this application.

1. More man power.
2. Time consuming.
3. Consumes large volume of pare work.
4. Needs manual calculations.
5. No direct role for the higher officials

**Proposed System of Inventory Management System:**

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system

1. Reduces the manual work.
2. Ensure data accuracies.
3. Proper control of the higher officials.
4. Minimize manual data entry.
5. Greater efficiency
6. Minimum time required

**Theoretical Analysis**

**Diagram:**

**Software Requirements:**

Client Side:

1. Any latest web browser like chrome, Firefox, etc.

Server Side:

1. Docker
   1. Only docker is required as the application is converted into docker image to deploy on docker.

**Hardware Requirements:**

Client Side:

1. Any modern pc or mobile device with active internet connection

Server Side:

1. Ram: Min 2 GB
2. Processor: Min 2 Core CPU

**Working**

Typically, Inventory Management Systems are used by firms that either sell a product or manufacture a product for the purpose of accounting all the tangible goods that allow for a sale of a finished product, or parts for making a product. This inventory management system can be used to store the details of the inventory, update the inventory based on the sale details, generate receipts for sales, and generate sales and inventory reports periodically. This inventory management software has one module, Admin. Admin has the authority to add, update and delete an inventory. This inventory management software also has its own intelligently managed support system. This intelligent support system allows admin to view and manage various inventories.

**Modules of Inventory Management system:**

**Dashboard:** Show total customer, product and total sell

**Inventory:** In this section owner of inventory add, edit, delete inventory.

**Customer:** In this section customer added in system after successful addition they can buy product. Customer can generate bill for their purchased product. Also, owner can delete customer

**Invoices:** Here invoice and bill details available

**Report:** View all previous invoices

**Profile:** In this section owner can change personal information and password

**Logout:** Logout from system.

**Conclusion:**

Thus, we have successfully completed our project on ‘Inventory Management System’. We have included many features that are necessary for an Inventory System.

The features are as follows:

* Checking of Alerts
* Product Information
* Customer Information
* Billing Module

While making the software, every effort has been taken to make a very easy to use Graphical User Interface (GUI). I have tried my best to include as much features as we can in the available time limit. Some additional advance features can also be implemented like

* Using a barcode system
* Recording of customer information for the study of buying habits but these features are kept for future development.

**Bibliography**

**References:**

1. [www.startvbdotnet.com](http://www.startvbdotnet.com)
2. <https://www.netsuite.com>
3. <https://cbsetoday.com>
4. <https://nevonprojects.com>

**Appendix:**

**Code of Application**

**App.py**

**from flask import Flask, render\_template, redirect, request, url\_for, flash, session, Markup**

**from flask\_migrate import Migrate**

**from flask\_login import login\_user, login\_required, logout\_user**

**from forms import \***

**from models import User, db, login\_manager, Customer, Stock, Cart, Billing**

**from sqlalchemy import func**

**from flask\_mail import Mail, Message**

**app = Flask(\_\_name\_\_)**

**app.config['SECRET\_KEY'] = 'supersecretkeyformyappdonttrytohack'**

**app.config['SQLALCHEMY\_DATABASE\_URI'] = 'mysql://root:@localhost/inventoryapp'**

**app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False**

**app.config['MAIL\_SERVER'] = 'smtp.gmail.com'**

**app.config['MAIL\_PORT'] = 465**

**app.config['MAIL\_USERNAME'] = 'rahulprojectmail@gmail.com'**

**app.config['MAIL\_PASSWORD'] = 'mypass' #not correct**

**app.config['MAIL\_USE\_TLS'] = False**

**app.config['MAIL\_USE\_SSL'] = True**

**mail = Mail(app)**

**db.init\_app(app)**

**Migrate(app, db)**

**login\_manager.init\_app(app)**

**login\_manager.login\_view = "login"**

**@app.route('/')**

**def index():**

**return redirect(url\_for('login'))**

**@app.errorhandler(404)**

**@login\_required**

**def page\_not\_found(e):**

**user = session['user']**

**return render\_template('404.html', user=user), 404**

**@app.route('/register', methods=['GET', 'POST'])**

**def register():**

**form = RegistrationForm()**

**if form.validate\_on\_submit():**

**email = User.query.filter\_by(email=form.email.data).first()**

**if email is None:**

**username = User.query.filter\_by(username=form.username.data).first()**

**if username is None:**

**register = User(firstname=form.firstname.data, lastname=form.lastname.data,**

**email=form.email.data, username=form.username.data, password=form.password.data)**

**db.session.add(register)**

**db.session.commit()**

**flash('Congrats you have registered successfully!, Please check your inbox')**

**welcome = form.firstname.data + ' ' + form.lastname.data + ' Welcome to Inventory Management System'**

**msg = Message(**

**'Welcome',**

**sender='rahulprojectmail@gmail.com',**

**recipients=[form.email.data]**

**)**

**msg.body = welcome**

**mail.send(msg)**

**return redirect(url\_for('login'))**

**else:**

**flash('Username already exists, please try again!')**

**return redirect(url\_for('register'))**

**else:**

**flash('Email already exists, please try logging in!')**

**return redirect(url\_for('register'))**

**return render\_template('signup.html', form=form)**

**@app.route('/login', methods=['GET', 'POST'])**

**def login():**

**form = LoginForm()**

**if form.validate\_on\_submit():**

**user = User.query.filter\_by(username=form.username.data).first()**

**if user is not None:**

**if user.check\_password(form.password.data):**

**session['user'] = form.username.data**

**login\_user(user)**

**next = request.args.get('next')**

**if next == None or not next[0] == '/':**

**next = url\_for('dashboard')**

**return redirect(next)**

**flash('Username/password not found!')**

**return redirect(url\_for('login'))**

**return render\_template('login.html', form=form)**

**@app.route('/dashboard')**

**@login\_required**

**def dashboard():**

**user = session['user']**

**heading = 'Dashboard'**

**user\_name = db.session.query(User.id).filter(User.username == user).first()**

**c\_rows = db.session.query(Customer).filter\_by(user\_id=user\_name.id).count()**

**s\_rows = db.session.query(Stock).filter\_by(user\_id=user\_name.id).count()**

**final\_amount = Billing.query.with\_entities(func.sum(Billing.total\_amount).label("totalamount")).filter\_by(user\_id=user\_name.id).first()**

**totalamt = final\_amount.totalamount**

**return render\_template('dashboard.html', user=user, heading=heading,**

**totalamt=totalamt, c\_rows=c\_rows, s\_rows=s\_rows)**

**@app.route('/inventory')**

**@login\_required**

**def inventory():**

**user = session['user']**

**heading = 'Inventory'**

**prod\_name = db.session.query(User.id).filter(User.username == user).first()**

**view\_products = Stock.query.filter\_by(user\_id=prod\_name.id)**

**return render\_template('inventory.html', user=user, heading=heading, view\_products=view\_products)**

**@app.route('/addproduct', methods=['GET', 'POST'])**

**@login\_required**

**def addproduct():**

**form = AddProduct()**

**user = session['user']**

**heading = 'Add New Product'**

**if form.validate\_on\_submit():**

**user\_name = db.session.query(User).filter(User.username == user).first()**

**item\_name = form.item\_name.data**

**item\_price = form.item\_price.data**

**item\_qty = form.item\_qty.data**

**item = Stock.query.filter\_by(item\_name=item\_name, user\_id=user\_name.id).first()**

**if item:**

**flash('Product already exists')**

**return redirect(url\_for('addproduct'))**

**else:**

**add\_product = Stock(item\_name=item\_name, item\_price=item\_price, item\_qty=item\_qty, user\_id=user\_name.id)**

**db.session.add(add\_product)**

**db.session.commit()**

**flash(Markup('Product added successfully! To view products <a href="/inventory" '**

**'class="alert-link">click here</a>'))**

**return redirect(url\_for('addproduct'))**

**return render\_template('addproduct.html', form=form, heading=heading)**

**@app.route('/updateproduct/<string:id>', methods=['GET', 'POST'])**

**@login\_required**

**def updateproduct(id):**

**user = session['user']**

**heading = 'Update Product Details'**

**product = Stock.query.filter\_by(id=id).one()**

**form = AddProduct(obj=product)**

**if form.validate\_on\_submit():**

**product = Stock.query.get(id)**

**form.populate\_obj(product)**

**db.session.commit()**

**flash('Product details updated Successfully')**

**return redirect(url\_for('inventory'))**

**return render\_template('addproduct.html', form=form,  user=user, heading=heading)**

**@app.route('/deleteproduct/<string:id>')**

**@login\_required**

**def deleteproduct(id):**

**Stock.query.filter\_by(id=id).delete()**

**db.session.commit()**

**flash('Product has been deleted!')**

**return redirect(url\_for('inventory'))**

**@app.route('/customers', methods=['GET', 'POST'])**

**@login\_required**

**def customers():**

**heading = 'Customers'**

**user = session['user']**

**user\_name = db.session.query(User.id).filter(User.username == user).first()**

**view\_customers = Customer.query.filter\_by(user\_id=user\_name.id)**

**return render\_template('customers.html', view\_customers=view\_customers, user=user, heading=heading)**

**@app.route('/addcustomer', methods=['GET', 'POST'])**

**@login\_required**

**def addcustomer():**

**form = AddCustomer()**

**user = session['user']**

**heading = 'Add New Customer'**

**if form.validate\_on\_submit():**

**user\_name = db.session.query(User).filter(User.username == user).first()**

**firstname = form.firstname.data**

**lastname = form.lastname.data**

**email = form.email.data**

**address = form.address.data**

**customer = Customer.query.filter\_by(email=email, user\_id=user\_name.id).first()**

**if customer:**

**flash(Markup('Customer already exists'))**

**return redirect(url\_for('addcustomer'))**

**else:**

**add\_customer = Customer(firstname=firstname, lastname=lastname, email=email, address=address,**

**user\_id=user\_name.id)**

**db.session.add(add\_customer)**

**db.session.commit()**

**flash(Markup('Customer added successfully! To view customerss <a href="/customers" class="alert-link">'**

**'click here</a>'))**

**return redirect(url\_for('addcustomer'))**

**return render\_template('addcustomer.html', form=form, user=user, heading=heading)**

**@app.route('/updatecustomer/<string:id>', methods=['GET', 'POST'])**

**@login\_required**

**def updatecustomer(id):**

**user = session['user']**

**heading = 'Add New Customer'**

**customer = Customer.query.filter\_by(id=id).one()**

**form = AddCustomer(obj=customer)**

**if form.validate\_on\_submit():**

**customer = Customer.query.get(id)**

**form.populate\_obj(customer)**

**db.session.commit()**

**flash('Customer details updated Successfully')**

**return redirect(url\_for('customers'))**

**return render\_template('addcustomer.html', form=form, user=user, heading=heading)**

**@app.route('/deletecustomer/<string:id>')**

**@login\_required**

**def deletecustomer(id):**

**Customer.query.filter\_by(id=id).delete()**

**db.session.commit()**

**flash('Customer has been deleted!')**

**return redirect(url\_for('customers'))**

**@app.route('/invoices')**

**@login\_required**

**def invoices():**

**user = session['user']**

**heading = 'Invoices'**

**user\_name = db.session.query(User.id).filter(User.username == user).first()**

**view\_customers = Customer.query.filter\_by(user\_id=user\_name.id).all()**

**return render\_template('invoices.html', user=user, heading=heading, view\_customers=view\_customers)**

**@app.route('/billing/<string:id>', methods=['GET', 'POST'])**

**@login\_required**

**def billing(id):**

**user = session['user']**

**user\_name = db.session.query(User.id).filter(User.username == user).first()**

**final\_amount = Cart.query.with\_entities(func.sum(Cart.total\_amount).label("totalamount")).filter\_by(customer\_id=id).first()**

**totalamt = final\_amount.totalamount**

**if totalamt is None:**

**flash('Invoice is Empty! Click on Create Bill to create new one!')**

**return redirect(url\_for('customers'))**

**else:**

**customer\_name = db.session.query(Customer.email).filter(Customer.id == id).first()**

**bill = Billing(customer\_name=customer\_name[0], total\_amount=totalamt, user\_id=user\_name.id)**

**db.session.add(bill)**

**Cart.query.filter\_by(customer\_id=id).delete()**

**db.session.commit()**

**emailid = customer\_name[0]**

**msg = Message(**

**'Bill Generated',**

**sender='rahulprojectmail@gmail.com',**

**recipients=[emailid]**

**)**

**msg.body = 'Your bill amount is: ' + str(totalamt)**

**mail.send(msg)**

**# flash('Bill has been generated!!')**

**return render\_template('billing.html', cost=totalamt, customer\_name=customer\_name[0], user=user)**

**@app.route('/reports')**

**@login\_required**

**def reports():**

**user = session['user']**

**heading = 'Reports'**

**user\_name = db.session.query(User.id).filter(User.username == user).first()**

**view\_bill = Billing.query.filter\_by(user\_id=user\_name.id).all()**

**final\_amount = Billing.query.with\_entities(func.sum(Billing.total\_amount).label("totalamount")).filter\_by(user\_id=user\_name.id).first()**

**totalamt = final\_amount.totalamount**

**return render\_template('reports.html', user=user, heading=heading, view\_bill=view\_bill, totalamt=totalamt)**

**@app.route('/profile', methods=['GET','Post'])**

**@login\_required**

**def profile():**

**# form = RegistrationForm()**

**user = session['user']**

**heading = 'Profile'**

**user\_id = db.session.query(User.id).filter(User.username == user).first()**

**user\_name = User.query.filter\_by(username=user).one()**

**# user = Customer.query.filter\_by(id=id).one()**

**form = ProfileForm(obj=user\_name)**

**if form.validate\_on\_submit():**

**user\_name = User.query.get(user\_id)**

**form.populate\_obj(user\_name)**

**db.session.commit()**

**flash('Customer details updated Successfully')**

**return redirect(url\_for('profile'))**

**return render\_template('profile.html', user=user, heading=heading, form=form)**

**@app.route('/viewbill/<string:id>')**

**@login\_required**

**def viewbill(id):**

**user = session['user']**

**view\_bill = Cart.query.filter\_by(customer\_id=id).all()**

**final\_amount = Cart.query.with\_entities(func.sum(Cart.total\_amount).label("totalamount")).filter\_by(customer\_id=id).first()**

**totalamt = final\_amount.totalamount**

**if totalamt is None:**

**flash('Invoice is Empty!')**

**return redirect(url\_for('invoices'))**

**else:**

**for customern in view\_bill:**

**customerr = customern.customer\_name**

**return render\_template('viewbill.html', view\_bill=view\_bill, totalamt=totalamt, id=id, user=user, customerr=customerr)**

**@app.route('/createinvoice/<string:id>', methods=['GET', 'POST'])**

**@login\_required**

**def createinvoice(id):**

**user = session['user']**

**form = InvoiceForm()**

**customer\_fname = db.session.query(Customer.firstname).filter(Customer.id == id).first()**

**customer\_lname = db.session.query(Customer.lastname).filter(Customer.id == id).first()**

**user\_name = db.session.query(User.id).filter(User.username == user).first()**

**view\_products = Stock.query.filter\_by(user\_id=user\_name.id).all()**

**form.products.choices = [(products.id, products.item\_name) for products in view\_products]**

**final\_amount = Cart.query.with\_entities(func.sum(Cart.total\_amount).label("totalamount")).filter\_by(**

**customer\_id=id).first()**

**totalamt = final\_amount.totalamount**

**if form.validate\_on\_submit():**

**customer\_name = db.session.query(Customer.email).filter(Customer.id == id).first()**

**s\_products = form.products.data**

**product\_name = db.session.query(Stock.item\_name).filter(Stock.id == s\_products).first()**

**items = Stock.query.filter\_by(id=s\_products).first()**

**quantity = form.item\_qty.data**

**if quantity > items.item\_qty:**

**flash('Not enough stock available of selected product!!')**

**return render\_template('createinvoice.html', form=form, customer\_fname=customer\_fname[0],**

**customer\_lname=customer\_lname[0], id=id, user=user)**

**else:**

**product\_price = db.session.query(Stock.item\_price).filter(Stock.id == s\_products).first()**

**total = int(quantity) \* int(product\_price[0])**

**invoice = Cart(customer\_name=customer\_name[0], item\_name=product\_name[0], item\_price=product\_price[0],**

**item\_quantity=quantity, total\_amount=total, user\_id=user\_name.id, customer\_id=id)**

**db.session.add(invoice)**

**product\_quantity = db.session.query(Stock.item\_qty).filter(Stock.id == s\_products).first()**

**available\_quantity = product\_quantity[0] - quantity**

**items.item\_qty = available\_quantity**

**db.session.commit()**

**flash('Product added to Cart Successfully!!!')**

**user\_email = db.session.query(User.email).filter(User.username == user).first()**

**useremail = user\_email[0]**

**if available\_quantity < 10:**

**prod\_name = db.session.query(Stock.item\_name).filter(Stock.id == s\_products).first()**

**flash('Available quantity is less than 10. Total available quantity is: '+ str(available\_quantity))**

**quantity\_alert = 'You will soon run out of quantity of product: ' \**

**+ prod\_name[0] + '\nPlease update your stock as soon as possible'**

**msg = Message(**

**'Quantity Alert',**

**sender='rahulprojectmail@gmail.com',**

**recipients=[useremail]**

**)**

**msg.body = quantity\_alert**

**mail.send(msg)**

**final\_amt = Cart.query.with\_entities(func.sum(Cart.total\_amount).label("totalamount")).filter\_by(**

**customer\_id=id).first()**

**totalamtn = final\_amt.totalamount**

**view\_bill = Cart.query.filter\_by(customer\_id=id).all()**

**for customern in view\_bill:**

**customerr = customern.customer\_name**

**return render\_template('createinvoice.html', form=form, customerr=customerr, totalamt=totalamt,**

**view\_bill=view\_bill, customer\_fname=customer\_fname[0],**

**customer\_lname=customer\_lname[0], totalamtn=totalamtn, id=id, user=user)**

**return render\_template('createinvoice.html', view\_products=view\_products, form=form,**

**customer\_fname=customer\_fname[0], customer\_lname=customer\_lname[0],**

**totalamt=totalamt, id=id, user=user)**

**@app.route('/logout')**

**@login\_required**

**def logout():**

**logout\_user()**

**return redirect(url\_for('login'))**

**if \_\_name\_\_ == "\_\_main\_\_":**

**app.run(debug=True)**

**forms.py**

**from flask\_wtf import FlaskForm**

**from wtforms import (StringField, SubmitField, PasswordField, IntegerField, TextAreaField, SelectField, SelectMultipleField)**

**from wtforms.validators import DataRequired, Email, EqualTo, Length**

**class LoginForm(FlaskForm):**

**username = StringField('Username', validators=[DataRequired()])**

**password = PasswordField('Password', validators=[DataRequired()])**

**submit = SubmitField('Login')**

**class RegistrationForm(FlaskForm):**

**firstname = StringField('First Name', validators=[DataRequired(), Length(min=3, max=50)])**

**lastname = StringField('Last Name', validators=[DataRequired(), Length(min=3, max=50)])**

**email = StringField('Email', validators=[DataRequired(), Email()])**

**username = StringField('Username', validators=[DataRequired(), Length(min=4, max=20, message='Username must be min 4 to max 20 characters!')])**

**password = PasswordField('Password', validators=[DataRequired(), EqualTo('pass\_confirm', message='Passwords Must Match!')])**

**pass\_confirm = PasswordField('Confirm password', validators=[DataRequired()])**

**submit = SubmitField('Register')**

**class ProfileForm(FlaskForm):**

**firstname = StringField('First Name', validators=[DataRequired(), Length(min=3, max=50)])**

**lastname = StringField('Last Name', validators=[DataRequired(), Length(min=3, max=50)])**

**email = StringField('Email', validators=[DataRequired(), Email()],  render\_kw={'disabled':'disabled'})**

**username = StringField('Username', validators=[DataRequired(), Length(min=4, max=20)], render\_kw={'disabled':'disabled'})**

**password = PasswordField('New Password', validators=[DataRequired(), EqualTo('pass\_confirm', message='Passwords Must Match!')])**

**pass\_confirm = PasswordField('Confirm password', validators=[DataRequired()])**

**submit = SubmitField('Update')**

**class AddCustomer(FlaskForm):**

**firstname = StringField('Name', validators=[DataRequired(), Length(min=3, max=50)])**

**lastname = StringField('Last Name', validators=[DataRequired(), Length(min=3, max=50)])**

**email = StringField('Email', validators=[DataRequired(), Email(message='Please enter valid email!'), Length(min=7, max=150)])**

**address = TextAreaField('Address', validators=[Length(max=200)])**

**submit = SubmitField('Submit')**

**class AddProduct(FlaskForm):**

**item\_name = StringField('Product Name', validators=[DataRequired(), Length(max=100)])**

**item\_price = IntegerField('Price', validators=[DataRequired()])**

**item\_qty = IntegerField('Quantity', validators=[DataRequired()])**

**submit = SubmitField('Submit')**

**class CreateBill(FlaskForm):**

**customer = SelectField('Select Customer', choices=[])**

**products = SelectField('Select Product', choices=[])**

**item\_qty = IntegerField('Quantity', validators=[DataRequired()])**

**submit = SubmitField('Add To Bill')**

**class InvoiceForm(FlaskForm):**

**products = SelectField('Products', choices=[])**

**item\_qty = IntegerField('Quantity', validators=[DataRequired()])**

**submit = SubmitField('Add To Cart')**

**models.py**

**from werkzeug.security import generate\_password\_hash,check\_password\_hash**

**from flask\_login import UserMixin**

**from flask\_login import LoginManager**

**from flask\_sqlalchemy import SQLAlchemy**

**from datetime import datetime**

**db = SQLAlchemy()**

**login\_manager = LoginManager()**

**@login\_manager.user\_loader**

**def load\_user(user\_id):**

**return User.query.get(user\_id)**

**# user module**

**class User(db.Model, UserMixin):**

**# table name (overridden)**

**\_\_tablename\_\_ = 'user'**

**# columns of table**

**id = db.Column(db.Integer, primary\_key=True)**

**firstname = db.Column(db.String(100))**

**lastname = db.Column(db.String(100))**

**email = db.Column(db.String(150), unique=True, index=True)**

**username = db.Column(db.String(64), unique=True, index=True)**

**password\_hash = db.Column(db.String(128))**

**# relations between tables (user as parent table)**

**customer\_id = db.relationship('Customer', backref='user', lazy='dynamic')**

**stock\_id = db.relationship('Stock', backref='user', lazy='dynamic')**

**cart\_id = db.relationship('Cart', backref='user', lazy='dynamic')**

**billing\_id = db.relationship('Billing', backref='user', lazy='dynamic')**

**def \_\_init\_\_(self, firstname, lastname, email, username, password):**

**self.firstname = firstname**

**self.lastname = lastname**

**self.email = email**

**self.username = username**

**self.password\_hash = generate\_password\_hash(password) # hashed the password**

**# function to check password**

**def check\_password(self, password):**

**return check\_password\_hash(self.password\_hash, password)**

**class Customer(db.Model):**

**id = db.Column(db.Integer, primary\_key=True)**

**firstname = db.Column(db.String(100))**

**lastname = db.Column(db.String(100))**

**email = db.Column(db.String(64))**

**address = db.Column(db.String(200))**

**created\_at = db.Column(db.DateTime, default=datetime.now)**

**user\_id = db.Column(db.Integer, db.ForeignKey('user.id', ondelete='CASCADE'))**

**cart\_id = db.relationship('Cart', backref='customer', lazy='dynamic')**

**def \_\_init\_\_(self, firstname, lastname, email, address, user\_id):**

**self.firstname = firstname**

**self.lastname = lastname**

**self.email = email**

**self.address = address**

**self.user\_id = user\_id**

**# stock module**

**class Stock(db.Model):**

**# columns of table**

**id = db.Column(db.Integer, primary\_key=True)**

**item\_name = db.Column(db.String(100), nullable=True)**

**item\_price = db.Column(db.Integer, nullable=True)**

**item\_qty = db.Column(db.Integer, nullable=True)**

**user\_id = db.Column(db.Integer, db.ForeignKey('user.id', ondelete='CASCADE'))**

**def \_\_init\_\_(self, item\_name, item\_price, item\_qty, user\_id):**

**self.item\_name = item\_name**

**self.item\_price = item\_price**

**self.item\_qty = item\_qty**

**self.user\_id = user\_id**

**class Cart(db.Model):**

**id = db.Column(db.Integer, primary\_key=True)**

**customer\_name = db.Column(db.String(100))**

**item\_name = db.Column(db.String(100))**

**item\_price = db.Column(db.Integer)**

**item\_quantity = db.Column(db.Integer)**

**total\_amount = db.Column(db.Integer)**

**generated\_at = db.Column(db.DateTime, default=datetime.now)**

**user\_id = db.Column(db.Integer, db.ForeignKey('user.id', ondelete='CASCADE'))**

**customer\_id = db.Column(db.Integer, db.ForeignKey('customer.id', ondelete='CASCADE'))**

**def \_\_init\_\_(self, customer\_name, item\_name, item\_price, item\_quantity, total\_amount, user\_id, customer\_id):**

**self.customer\_name = customer\_name**

**self.item\_name = item\_name**

**self.item\_price = item\_price**

**self.item\_quantity = item\_quantity**

**self.total\_amount = total\_amount**

**self.user\_id = user\_id**

**self.customer\_id = customer\_id**

**# def final\_amount(total\_amount):**

**class Billing(db.Model):**

**id = db.Column(db.Integer, primary\_key=True)**

**customer\_name = db.Column(db.String(100))**

**total\_amount = db.Column(db.Integer)**

**generated\_at = db.Column(db.DateTime, default=datetime.now)**

**user\_id = db.Column(db.Integer, db.ForeignKey('user.id', ondelete='CASCADE'))**

**def \_\_init\_\_(self, customer\_name, total\_amount, user\_id):**

**self.customer\_name = customer\_name**

**self.total\_amount = total\_amount**

**self.user\_id = user\_id**

**docker file:**

**FROM python:3.6.5-alpine**

**WORKDIR /app**

**ADD . /app**

**RUN set -e; \**

**apk add --no--cache --virtual .build-deps \**

**gcc \**

**libc-dev \**

**linux-headers \**

**mariadb-dev \**

**python3-dev \**

**postgresql-dev \**

**;**

**COPY requirements.txt /app**

**RUN pip install -r requirements.txt**

**CMD [ "python","app.py"]**