**CHAPTER - 1**

**INTRODUCTION**

**1.1 INTRODUCTION OF PROJECT**

This whole report is about the “VASTU ” project done for students. This project lets students buy and sell items off of the other students in the same college or different.

Vastu is an online marketplace built using the Flask framework, a platform only for students where they can buy or sell items such as books, electronics, fashion items, etc. Sometimes when a student wants to move out of college or wants to find things cheaper than the normal rate for his simple college purpose, they can look up the site pretty easily. A student can list his/ her item for sale and also offer other students his service and earn money. The only requirement to register on this platform is to be able to upload an id given by the college to prove his identity. A student can offer a negotiable amount via a chat option. Privacy and bad behavior would be monitored too for misuse purposes. There are many categories to upload items or choose and complete transparency towards which college the product is uploaded from is maintained by asking the users to upload at the registration.

**1.2 OBJECTIVES OF A PROJECT**

The main objective of this project is to provide a way for user’s to earn money through his services or reselling the items that the student feels not needing anymore and to make the best use of his items for other students at a cheaper rate than normal. The system allows students to log in and register with their corresponding college id as proof of identity. The students can negotiate and finalize a place to exchange via messages.

**1.3 SCOPE OF THE PROJECT**

The system allows the user to login into the application through their corresponding email and password. The signup process includes their username, email, password, and college id which is stored in the MySQL database.

The name of the application is “VASTU” which provides a user interface and certain features which includes

* Storing student information
* Storing the student messages of negotiation of items.
* Storing students uploaded items.
* Option for offering a price to the user via messages.
* Viewing the products with different categories.

**1.4 MODULES DESCRIPTION**

This project consists of various modules as follows:

* **Profile**

In this module, The user can view their information such as their name, email college name, and joining date. Users get all the rights to view, update and delete any information on the site which belongs to them once they are logged in.

* **Chat**

This module provides us with functionality to negotiate with other students of their items via message which includes the students to view the messages, send or delete them. The chat module stores messages, user information, and item information.

* **Home**

This module provides us the items that other users have uploaded except the logged-in user. The logged-in user can send an offer via the offer button present inside each item box.

* **Categories**

This module contains items of different categories a user can choose from for his certain needs.

* **MyAds**

This module contains the items uploaded by the logged-in user. The user can delete the product once it is negotiated and sold to other students.

* **Sell**

This module contains the functionality to upload the student’s item to the system.

* **Log in**

This module contains the log-in functionality for the users to logged-in to their respective accounts to use the service.

* **Register**

This module contains the signup functionality for the user to register to the system.

**CHAPTER - 2**

**SYSTEM ANALYSIS**

**2.1 SYSTEM STUDY**

The system analysis is a detailed appraisal of the existing system. This appraisal includes how the system works and what it does. It also includes finding out in more detail- what are the problems with the system and what the user requires from the new system or any new change in the system. The output of this phase results in a detailed model of the system. The model describes the system functions and data and system information flow.

**2.2 EXISTING SYSTEM**

The existing system of an online selling platform has features like selling, chatting, and buying features which is good for the current market scenario.

**2.3 PROPOSED SYSTEM**

The proposed system enhances the existing system with features restricted to students to maintain the decorum. Students usually want to buy their items at a cheaper rate with their trusted contacts. The system provides the native feature of selling and buying items at a cheaper rate from the trusted people.

**CHAPTER -3**

**SYSTEM DESIGN**

System design is the process of defining the architecture, models, interface, and data for a system to satisfy specified requirements. System design could be the application of system theory to product development. System design implies a systematic approach but either way, the process is systematic where it considers all related variables of the system that need’s to be created - from the architecture to the required hardware and the software, right down to the data and how it travels and transforms throughout its travel system. System design then overlaps with system analysis engineering and system architecture.

**3.1 FLASK FRAMEWORK**

**Flask** is a micro [web framework](https://en.wikipedia.org/wiki/Web_framework) written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). It is classified as a [microframework](https://en.wikipedia.org/wiki/Microframework) because it does not require particular tools or libraries.[[2]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-2) It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several common framework-related tools.

**COMPONENTS**

The microframework Flask is part of the *Pallets Projects* (formerly *Pocono*) and is based on several others of them.

* **Werkzeug**

Werkzeug is a utility library for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)), in other words, a toolkit for [Web Server Gateway Interface](https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface) (WSGI) applications, and is licensed under a [BSD License](https://en.wikipedia.org/wiki/BSD_licenses). Werkzeug can realize software objects for request, response, and utility functions. It can be used to build a custom [software framework](https://en.wikipedia.org/wiki/Software_framework) on top of it and supports Python 2.7 and 3.5 and later.[[15]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-AR-Werkzeug-15)[[16]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-16)

* **Jinja**

Jinja, also by Ronacher, is a [template engine](https://en.wikipedia.org/wiki/Template_engine_(web)) for the Python programming language and is licensed under a BSD License. Similar to the [Django web framework](https://en.wikipedia.org/wiki/Django_(web_framework)), it handles templates in a [sandbox](https://en.wikipedia.org/wiki/Sandbox_(computer_security)).

* **MarkupSafe**

MarkupSafe is a [string](https://en.wikipedia.org/wiki/String_(computer_science)) handling library for the Python programming language, licensed under a BSD license. The eponymous MarkupSafe [type](https://en.wikipedia.org/wiki/Class_(computer_programming)) extends the Python string type and marks its contents as "safe"; combining MarkupSafe with regular strings automatically escapes the unmarked strings while avoiding double escaping of already marked strings.

* **ItsDangerous**

ItsDangerous is a safe [data serialization](https://en.wikipedia.org/wiki/Serialization) library for the Python programming language, licensed under a BSD license. It is used to store the [session](https://en.wikipedia.org/wiki/Session_(computer_science)) of a Flask application in a [cookie](https://en.wikipedia.org/wiki/HTTP_cookie) without allowing users to tamper with the session contents.

**3.2 DATAFLOW DIAGRAM**

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into detail, which can later be processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether the process will operate in sequence or parallel, unlike a traditionally structured flow chart that focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to systems design. This context-level DFD is next “exploded”, to produce a LEVEL 1 DFD that shows some of the detail of the system being modeled. The LEVEL 1 DFD shows how the system is divided into sub-systems(process), each of which deals with one or more of the data flows to or from an external agent, and which together provide all the functionality of the system. It also identifies internal data stores that must be present for the system to do its job and shows the flow of data between the various parts of the system.

**DFD SYMBOLS**

****

**CONSTRUCTING A DFD:**

* Several rules of thumb are used in drawing DFD’s.
* The process should be named and numbered for easy reference. Each name should be representative of the process.
* The direction of flow is from top to bottom and from left to right. Data traditionally flows from source to destination although they may back to the source.
* One way to indicate is to draw a long flow line back to the source. An alternative way to repeat the source symbols as a destination. Since it is used more than once in the DFD, it is marked with a short diagonal.
* A DFD typically shows the minimum contents of the data source.

**SILENT FEATURES OF DFD’s**

* DFD shows the flow of data. Now of control loops and decisions are controlled consideration does not appear in a DFD.
* The DFD does not indicate the time factor involved in any process whether the data flow takes place daily, weekly monthly, or yearly.
* The sequence of events is not brought out on the DFD

**Types of data flow diagram:**

* Current physical
* Current Logical
* New Physical
* New Logical

**Current Physical -**  In the DFD process level includes the name of people are their position are the name of a computer system that might provide some of the overall system-processing. Levels included an identification of the technology used to process the data flows and data stores and often levels with the name of actual physical media on which data are should such as file folders, computer files, business forms are computer tapes.

**Current Logical -** The physical aspects of the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

**New Logical -** This is like a current logical model if the user were completely happy with the le user was completely happy with the functionally of the current system but had problems with how it was implemented typically through the new logical model will differ from the current logical model while having additional functions, absolute function removal and inefficient flows recognized.

**New Physical -** The new physical representation only the physical implication of the new system.

**3.3 ER DIAGRAM**

An Entity-Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects, or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education, and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals, and connecting lines to depict the interconnectedness of entities, relationships, and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

**3.3.1 COMPONENTS OF ER DIAGRAM**

ER Diagrams are composed of entities, relationships, and attributes. They also depict cardinality, which defines relationships in terms of numbers. Here’s a glossary:

**Entity**

A definable thing—such as a person, object, concept, or event—can have data stored about it. Think of entities as nouns. Examples: a customer, student, car, or product. Typically shown as a rectangle.

**Entity type:** A group of definable things, such as students or athletes, whereas the entity would be the specific student or athlete. Other examples: customers, cars, or products.

**Entity set**: Same as an entity type, but defined at a particular point in time, such as students enrolled in a class on the first day. Other examples: Customers who purchased last month, cars currently registered in Florida. A related term is an instance, in which the specific person or car would be an instance of the entity set.

**Entity categories:** Entities are categorized as strong, weak, or associative. A strong entity can be defined solely by its attributes, while a weak entity cannot. An associative entity associates entities (or elements) within an entity set.

**Entity keys:** Refers to an attribute that uniquely defines an entity in an entity set. Entity keys can be super, candidate, or primary.

**Super key:** A set of attributes (one or more) that together define an entity in an entity set. **Candidate key:** A minimal super key, meaning it has the least possible number of attributes to still be a super key. An entity set may have more than one candidate key.

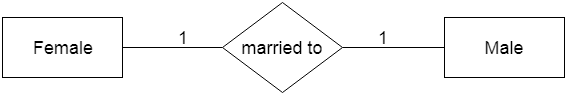
**Primary key:** A candidate key chosen by the database designer to uniquely identify the entity set. Foreign key: Identifies the relationship between entities.

**3.3.2 E R DIAGRAM RELATIONSHIP**

How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.

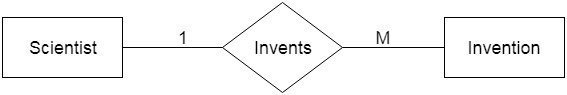
**One-to-One Relationship:** When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry one male, and a male can marry one female**.**

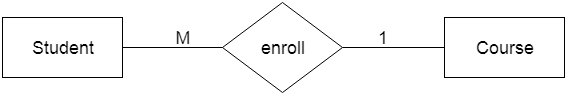


**One-to-many relationship:** When only one instance of the entity on the left, and more than one instance of an entity on the right associate with the relationship then this is known as a one-to-many relationship.

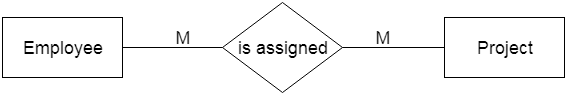
For example, Scientists can invent many inventions, but the invention is done by only specific scientists.



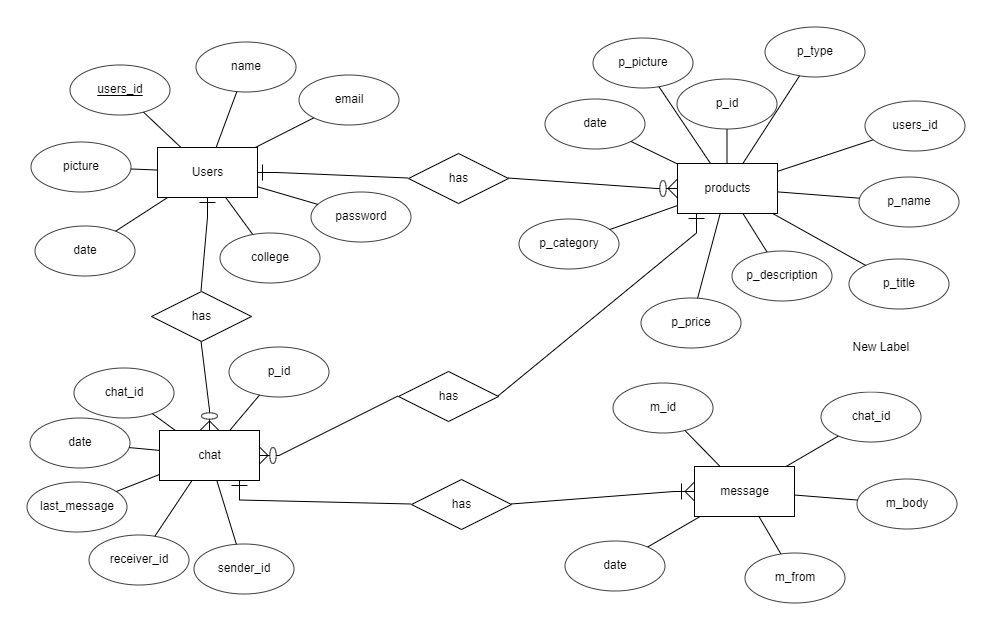
**Many-to-one relationship:** When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship. For example, a Student enrolls for only one course, but a course can have many students.

****

**Many-to-many relationship:** When more than one instance of the entity on the left, and more than one instance of an entity on the right associate with the relationship then it is known as a many-to-many relationship. For example, Employees can assign by many projects and projects can have many employees.



**3.3.2 VASTU E-R DIAGRAM**



**3.4 CODING**

**app.py**

**from datetime import datetime**

**from flask import Flask,render\_template,request,session,redirect,jsonify,flash**

**from flask.helpers import url\_for**

**import mysql.connector**

**import os**

**from werkzeug.utils import secure\_filename**

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

**app.secret\_key = os.urandom(24)**

**app.config['UPLOAD\_FOLDER'] = 'static/uploads'**

**app.config['MAX\_CONTENT\_LENGTH'] = 16\*1024\*1024**

**ALLOWED\_EXTENSIONS = set(['png','jpg','jpeg','gif'])**

**# Database connection**

**mydb = mysql.connector.connect(host="localhost",user="root",password="",database="vastu")**

**cursor = mydb.cursor()**

**# home page**

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

**def home():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try :**

**current\_user\_id = int(session.get('user\_id'))**

**cursor.execute("SELECT users.college,products.\*,users.name FROM users \**

**JOIN products ON users.users\_id = products.users\_id WHERE NOT users.users\_id=%s ORDER BY products.date DESC",(current\_user\_id,))**

**result = cursor.fetchall()**

**return render\_template("index.html",session\_user\_name=session.get('user\_name'),result=result)**

**except Exception as e:**

**return (str(e))**

**# profile page**

**@app.route("/profile")**

**def profile():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try:**

**# fetch user profile information**

**cursor.execute("SELECT \* FROM users where users\_id = %s ",(session.get('user\_id'),))**

**result = cursor.fetchall()**

**user\_name = result[0][1]**

**user\_email = result[0][2]**

**user\_college = result[0][4]**

**# Formatting the date**

**user\_joined\_date = datetime.date(result[0][5])**

**user\_joined\_date = str(user\_joined\_date.day)+ " "+ str(user\_joined\_date.strftime("%B"))+", "+str(user\_joined\_date.year)**

**user\_picture = result[0][6]**

**return render\_template("profile.html",**

**session\_user\_name=session.get('user\_name'),**

**user\_name=user\_name,**

**user\_email=user\_email,**

**user\_college=user\_college,**

**user\_picture=user\_picture,**

**user\_joined\_date=user\_joined\_date)**

**except Exception as e:**

**return (str(e))**

**# upload image on profile**

**def allowed\_file(filename):**

**return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS**

**@app.route("/profileimage",methods=['POST'])**

**def profile\_image():**

**if 'file' not in request.files:**

**flash('No file part')**

**return redirect(request.url)**

**file = request.files['file']**

**if file.filename == '':**

**flash('No image selected for uploading')**

**return redirect(request.url)**

**if file and allowed\_file(file.filename):**

**try:**

**current\_user\_id = int(session.get('user\_id'))**

**filename = secure\_filename(str(datetime.now())+" "+file.filename)**

**file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))**

**cursor.execute("UPDATE users SET picture=%s WHERE users\_id=%s",(filename,current\_user\_id))**

**mydb.commit()**

**return redirect('/profile')**

**except Exception as e:**

**return str(e)**

**else:**

**flash('Allowed image types are - png, jpg, jpeg, gif')**

**return redirect(request.url)**

**# category page**

**@app.route("/category")**

**def category():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**return render\_template("category.html",session\_user\_name=session.get('user\_name'))**

**# show category page**

**@app.route("/showcategory/<cat>")**

**def show\_category(cat):**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try:**

**current\_user\_id = int(session.get('user\_id'))**

**cursor.execute("SELECT users.college,products.\*,users.name FROM users \**

**JOIN products ON users.users\_id = products.users\_id WHERE NOT users.users\_id=%s \**

**AND products.p\_category=%s",(current\_user\_id,cat))**

**result = cursor.fetchall()**

**if cat=="sportandhobbies":**

**cat = "Sport and Hobbies"**

**elif cat=="computerandelectronics":**

**cat = "Computer and Electronics"**

**return render\_template("showcategory.html",session\_user\_name=session.get('user\_name'),category=cat.capitalize(),result=result)**

**except Exception as e:**

**return (str(e))**

**# myads page**

**@app.route("/myads")**

**def my\_ads():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try:**

**current\_user\_id = int(session.get('user\_id'))**

**cursor.execute("SELECT users.college,products.\* FROM users \**

**JOIN products ON users.users\_id = products.users\_id \**

**WHERE users.users\_id=%s ORDER BY products.date DESC",(current\_user\_id,))**

**result = cursor.fetchall()**

**return render\_template("myads.html",session\_user\_name=session.get('user\_name'),result=result)**

**except Exception as e:**

**return (str(e))**

**# delete post endpoints**

**@app.route("/deletemyads/<int:p\_id>")**

**def delete\_my\_ads(p\_id):**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try:**

**current\_user\_id = int(session.get('user\_id'))**

**cursor.execute("DELETE FROM products WHERE p\_id=%s AND users\_id=%s",(p\_id,current\_user\_id))**

**mydb.commit()**

**return redirect("/myads")**

**except Exception as e:**

**return str(e)**

**# View products**

**@app.route("/view/<int:p\_id>")**

**def view(p\_id):**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try:**

**cursor.execute("SELECT users.name,users.college,products.\* FROM users \**

**JOIN products ON users.users\_id = products.users\_id \**

**WHERE products.p\_id = %s",(p\_id,))**

**result = cursor.fetchall()**

**current\_user\_id = int(session.get('user\_id'))**

**return render\_template("view.html",session\_user\_name=session.get('user\_name'),result=result,current\_user\_id=current\_user\_id)**

**except Exception as e:**

**return (str(e))**

**# sell**

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

**def sell():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**return render\_template('sell.html',session\_user\_name=session.get('user\_name'))**

**# product submit page**

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

**def post\_submit():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try :**

**current\_user\_id = int(session.get('user\_id'))**

**if request.method=='POST':**

**if 'file' not in request.files:**

**flash('No file part')**

**return redirect(request.url)**

**file = request.files['file']**

**if file.filename == '':**

**flash('No image selected for uploading')**

**return redirect(request.url)**

**if file and allowed\_file(file.filename):**

**name = request.form.get('p\_name')**

**title = request.form.get('p\_title')**

**description = request.form.get('p\_description')**

**price = int(request.form.get('p\_price'))**

**category = request.form.get('p\_category')**

**pType = request.form.get('p\_type')**

**now = datetime.now()**

**filename = secure\_filename(str(datetime.now())+" "+file.filename)**

**file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))**

**cursor.execute("INSERT INTO products(users\_id,p\_name,p\_title,p\_description,p\_price,p\_category,date,p\_type,p\_picture) \**

**VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s)",(current\_user\_id,name,title,description,price,category,now,pType,filename))**

**mydb.commit()**

**return redirect('/myads')**

**else:**

**flash('Allowed image types are - png, jpg, jpeg, gif')**

**return redirect(request.url)**

**return render\_template('sell.html',session\_user\_name=session.get('user\_name'))**

**except Exception as e:**

**return (str(e))**

**# chat page**

**@app.route("/chat")**

**def chat():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try:**

**current\_user\_id = int(session.get('user\_id'))**

**# name , users\_id , chat\_id , p\_id , last\_message , p\_name**

**cursor.execute("SELECT users.name,users.users\_id,chat.chat\_id,chat.p\_id, chat.last\_message,products.p\_name,users.picture,products.p\_picture FROM users \**

**JOIN chat ON users.users\_id = chat.sender\_id OR users.users\_id = chat.receiver\_id \**

**JOIN products ON products.p\_id = chat.p\_id\**

**WHERE chat.sender\_id = %s OR chat.receiver\_id = %s ORDER BY chat.date DESC",(current\_user\_id,current\_user\_id))**

**result = cursor.fetchall()**

**res = []**

**# removing the data of current user**

**for x in result:**

**if x[1]!=current\_user\_id:**

**res.append(x)**

**return render\_template("chat.html",session\_user\_name=session.get('user\_name'),result=res)**

**except Exception as e:**

**return (str(e))**

**# offer post page**

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

**def offer():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try:**

**current\_user\_id = int(session.get('user\_id'))**

**if request.method =="POST":**

**p\_id = int(request.form.get("p\_id"))**

**message = request.form.get("message")**

**receiver\_id = int(request.form.get("receiver\_id"))**

**cursor.execute("SELECT chat\_id FROM chat WHERE ((sender\_id=%s AND receiver\_id=%s) OR (sender\_id=%s AND receiver\_id=%s)) \**

**AND p\_id=%s",(current\_user\_id,receiver\_id,receiver\_id,current\_user\_id,p\_id))**

**result = cursor.fetchall()**

**if len(result)==0:**

**cursor.execute("INSERT INTO chat(p\_id,sender\_id,receiver\_id,last\_message,date) VALUES(%s,%s,%s,%s,%s)",**

**(p\_id,current\_user\_id,receiver\_id,message,datetime.now()))**

**last\_id = cursor.lastrowid**

**mydb.commit()**

**cursor.execute("INSERT INTO message(chat\_id,m\_body,m\_from,date) VALUES(%s,%s,%s,%s)",(last\_id,message,current\_user\_id,datetime.now()))**

**mydb.commit()**

**return redirect('/chat')**

**res = str(result[0][0])**

**cursor.execute("INSERT INTO message(chat\_id,m\_body,m\_from,date) VALUES(%s,%s,%s,%s)",(int(res),message,current\_user\_id,datetime.now()))**

**mydb.commit()**

**cursor.execute("UPDATE chat SET last\_message=%s,date=%s WHERE chat\_id=%s",(message,datetime.now(),int(res)))**

**mydb.commit()**

**return redirect("/showchat/"+res)**

**except Exception as e:**

**return (str(e))**

**# Showcase of specific chat page**

**@app.route('/showchat/<int:chat\_id>')**

**def show\_chat(chat\_id):**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**current\_user\_id = int(session.get('user\_id'))**

**try:**

**cursor.execute("SELECT users.users\_id,users.name,users.college,products.p\_name,products.p\_picture,users.picture FROM users \**

**JOIN chat ON users.users\_id = chat.sender\_id OR users.users\_id = chat.receiver\_id \**

**JOIN products ON chat.p\_id = products.p\_id \**

**WHERE chat.chat\_id = %s",(chat\_id,))**

**result = cursor.fetchall()**

**res = []**

**# removing the data of current user**

**for x in result:**

**if x[0]!=current\_user\_id:**

**res.append(x)**

**print(res)**

**return render\_template("showchat.html",session\_user\_name=session.get('user\_name'),chat\_id=chat\_id,current\_user\_id=current\_user\_id,result=res)**

**except Exception as e:**

**return (str(e))**

**# Fetching message page**

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

**def fetch\_message():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try:**

**chat\_id = int(request.form.get('chat\_id'))**

**cursor.execute("SELECT m\_body,m\_from,date FROM message WHERE chat\_id=%s ORDER BY date ASC",(chat\_id,))**

**result = cursor.fetchall()**

**return jsonify({'result' : result})**

**except Exception as e:**

**return (str(e))**

**# send messgae post page**

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

**def send\_message():**

**# if the user is logged in**

**if 'user\_id' not in session:**

**return redirect('/login')**

**try :**

**message = request.form.get('message')**

**chat\_id = request.form.get('chat\_id')**

**current\_user\_id = int(session.get('user\_id'))**

**cursor.execute("INSERT INTO message(chat\_id,m\_body,m\_from,date) VALUES(%s,%s,%s,%s)",(chat\_id,message,current\_user\_id,datetime.now()))**

**mydb.commit()**

**cursor.execute("UPDATE chat SET last\_message=%s,date=%s WHERE chat\_id=%s",(message,datetime.now(),chat\_id))**

**mydb.commit()**

**return jsonify({'message': "done"})**

**except Exception as e:**

**return (str(e))**

**# login page**

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

**def login():**

**# if the user is already logged in**

**if 'user\_id' in session:**

**return redirect('/')**

**try:**

**if request.method=='POST':**

**email = request.form.get('email')**

**password = request.form.get('password')**

**cursor.execute('''SELECT \* FROM users where email=%s and password=%s LIMIT 1''',(email,password))**

**users = cursor.fetchall()**

**if len(users)>0:**

**session['user\_id'] = users[0][0]**

**session['user\_name'] = users[0][1]**

**return redirect('/')**

**else :**

**return redirect(url\_for('login',q="email or password not found"))**

**return render\_template('login.html',message=request.args.get("q", ""))**

**except Exception as e:**

**return (str(e))**

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

**def logout():**

**session.pop('user\_id',default=None)**

**session.pop('user\_name',default=None)**

**return redirect('/login')**

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

**def register():**

**# if the user is already logged in**

**if 'user\_id' in session:**

**return redirect('/')**

**try:**

**if request.method=='POST':**

**name = request.form.get('name')**

**email = request.form.get('email')**

**password = request.form.get('password')**

**college = request.form.get('college')**

**now = datetime.now()**

**cursor.execute("SELECT email FROM users where email = %s ",(email,))**

**myresult = cursor.fetchall()**

**if len(myresult)==0:**

**cursor.execute(''' INSERT INTO users(name,email,password,college,date) VALUES(%s,%s,%s,%s,%s) ''',(name,email,password,college,now))**

**mydb.commit()**

**cursor.execute("SELECT \* FROM users where email = %s ",(email,))**

**myresult = cursor.fetchall()**

**session['user\_id'] = myresult[0][0]**

**session['user\_name'] = myresult[0][1]**

**return redirect('/')**

**else :**

**return redirect(url\_for('register',q="email already registered"))**

**return render\_template('register.html',message=request.args.get("q", ""))**

**except Exception as e:**

**return (str(e))**

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

**app.run(host='localhost', debug=True)**

**login.html**

|  |
| --- |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <link rel="stylesheet" href="/static/login.css">  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <link rel="icon" href="/static/transfer.png">   <title>VASTU LOGIN</title> </head> <body>  <div class="login" >  <div class="text-center title">  <h4> VASTU LOGIN</h4>  </div>  <form action="/login" method="POST">  {% if message!="" %}  <div class="alert alert-danger text-center">{{message}}</div>  {% endif %}  <div class="form-group">  <label for="exampleInputEmail1">Email address</label>  <input type="email" name="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email">  </div>  <div class="form-group">  <label for="exampleInputPassword1">Password</label>  <input type="password" name="password" class="form-control" id="exampleInputPassword1" placeholder="Password">  </div>    <button type="submit" class="btn btn-primary btn-block" >Login</button>  <p style="margin-top: 10px;">Not a member? <a href="register" >Create account</a></p>    </form>  </div>   <!-- Optional JavaScript -->  <!-- jQuery first, then Popper.js, then Bootstrap JS -->  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>  <script src='https://kit.fontawesome.com/a076d05399.js' crossorigin='anonymous'></script> </body> </html>** |

**Register.html**

|  |
| --- |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <link rel="stylesheet" href="/static/login.css">  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <link rel="icon" href="/static/transfer.png">   <title>VASTU REGISTER</title> </head> <body style="background-image: url('/static/background.jpg');">    <div class="register">  <div class="text-center title">  <h4> VASTU REGISTER</h4>    </div>  <form action="/register" method="POST">  {% if message!="" %}  <div class="alert alert-danger text-center">{{message}}</div>  {% endif %}  <div class="form-group">  <label for="exampleInputName">Full name</label>  <input type="text" name="name" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter name" required>  </div>  <div class="form-group">  <label for="exampleInputEmail1">Email address</label>  <input type="email" name="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email" required>  </div>  <div class="form-group">  <label for="exampleInputPassword1">Password</label>  <input type="password" name="password" class="form-control" id="exampleInputPassword1" placeholder="Password" required>  </div>  <div class="form-group">  <label for="exampleInputPassword1">College</label>  <input type="text" name="college" class="form-control" id="exampleInputPassword1" placeholder="Enter College name" required>   </div>  <button type="submit" class="btn btn-primary btn-block" style="margin-bottom: 5px;">Register</button>  <p style="margin-top: 10px;">Already a member? <a href="login" >Login here</a></p>  </form>  </div>   <!-- Optional JavaScript -->  <!-- jQuery first, then Popper.js, then Bootstrap JS -->  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>  <script src='https://kit.fontawesome.com/a076d05399.js' crossorigin='anonymous'></script> </body> </html>** |

**Profile.html**

|  |
| --- |
| **{% extends 'base.html' %} {% block externalcss %}  <link rel="stylesheet" href="/static/profile.css"> {% endblock externalcss %} {% block title %}  Profile {% endblock title %} {% block userName %}  {{session\_user\_name}} {% endblock userName %} {% block body %}   <div class="container " style="margin-top: 10px;padding: 10px;">  <div class="row justify-content-md-center" style="border: 1px solid lightgrey;padding: 30px;box-shadow: -3px 3px 19px 0px rgba(0,0,0,0.75);  -webkit-box-shadow: -3px 3px 19px 0px rgba(0,0,0,0.75);  -moz-box-shadow: -3px 3px 19px 0px rgba(0,0,0,0.75);">  <div class="col-4 colLeft border">  <img src="/static/uploads/{{user\_picture}}" class="profile-image" alt="">  <form action="/profileimage" method = "POST" enctype = "multipart/form-data">  <input type="file" class="form-control" name="file" required>  <button type="submit" class="form-control btn bg-danger text-white">Update Picture</button>  </form>  </div>  <div class="col-6">  <ul class="list-group">  <li class="list-group-item active" aria-current="true">  <i class="material-icons">insert\_emoticon</i> Personal Information  <span style="float: right;"></span>  </li>  <li class="list-group-item">Name : <b>{{user\_name}}</b></li>  <li class="list-group-item">Email : <b>{{user\_email}}</b></li>  <li class="list-group-item">Join On : <b>{{user\_joined\_date}}</b></li>  <li class="list-group-item">College : <b>{{user\_college}}</b></li>  </ul>  </div>  </div>  </div>   {% endblock body %}** |

**chat.html**

|  |
| --- |
| **{% extends 'base.html' %} {% block title %}  Chat {% endblock title %} {% block userName %}  {{session\_user\_name}} {% endblock userName %} {% block activeChat %}  active {% endblock activeChat %} {% block body %}  <div class="container"> <div class="row">  <div class="col-lg-9">     <div class="list-group">  {% for values in result %}  <a href="/showchat/{{values[2]}}"   class="bg-light bg-gradient list-group-item border border-5 shadow list-group-item-action"  style="position:relative;padding:4px 100px;display: flex;justify-content: flex-start;align-items: center;">  <div style="position: relative;margin-right: 30px;">  <img src="/static/uploads/{{values[7]}}" style="height: 60px;width: 60px;" alt="">  <img src="/static/uploads/{{values[6]}}" style="height: 40px;width: 40px;border-radius: 50%;position: absolute;right: -16px;bottom: -10px;" alt="">  </div>  <div style="margin-top:12px;margin-right: 20px;display: flex;justify-content: center;align-items: flex-start;flex-direction: column;">  <h4>{{values[0]}}</h4>  <p>{{values[4]}}</p>  </div>  <div style="position: absolute;right: 100px;display: flex;align-items: center;">  <p class="badge bg-primary p-3 text-white">{{values[5]}}</p>  </div>  </a>  {% endfor %}   </div> </div>  </div>  </div>   {% endblock body %}** |

**sell.html**

|  |
| --- |
| **{% extends 'base.html' %} {% block title %}  Sell {% endblock title %} {% block userName %}  {{session\_user\_name}} {% endblock userName %} {% block activeSell %}  active  {% endblock activeSell %} {% block body %}  <div class="container">  <div class="row">  <div class="col-md-3"></div>  <div class="col-md-6 border p-3" style="background: lightgrey;font-size: 1.2rem;font-weight: bold;">  <form action="/postsubmit" method="POST" enctype="multipart/form-data">  <div class="form-group">  <label for=""><b>@</b> Post Title</label>  <input type="text" name="p\_title" autocomplete="off" class="form-control" placeholder="enter post title" required>  </div>  <div class="form-group">  <label for=""><b>@</b> Product Name</label>  <input type="text" name="p\_name" autocomplete="off" class="form-control" placeholder="enter product name" required>  </div>  <div class="form-group">  <label for=""><b>@</b> Price</label>  <input type="number" name="p\_price" autocomplete="off" class="form-control" placeholder="enter price in rs" required>  </div>  <div class="form-group">  <label for=""><b>@</b> Category</label>  <select class="form-control" name="p\_category" required>  <option value="books">Books</option>  <option value="stationary">Stationary</option>  <option value="mobiles">Mobiles</option>  <option value="sportandhobbies">Sport and Hobbies</option>  <option value="services">Services</option>  <option value="bikes">Bikes</option>  <option value="computerandelectronics">Computer and Electronics</option>  <option value="fashion">Fashion</option>  <option value="other">Other</option>  </select>  </div>  <div class="form-group">  <label for=""><b>@</b> Description</label>  <textarea type="text" name="p\_description" rows="3" autocomplete="off" class="form-control" placeholder="product description" required></textarea>  </div>  <div class="form-group">  <label for=""> <b>@</b> Type</label>  <select class="form-control" name="p\_type" required>  <option value="1">Sell</option>  <option value="2">Exchange</option>  </select>  </div>  <div class="form-group">  <label for=""><b>@</b> Picture</label><br>  <input type="file" name="file">  </div>  <button type="submit" class="btn btn-primary float-right btn-lg">Post &#9654;</button>  </form>  </div>  <div class="col-md-3"></div>   </div> </div>   {% endblock body %}** |

**Myads.html**

**{% extends 'base.html' %}**

**{% block title %}**

**My Ads**

**{% endblock title %}**

**{% block userName %}**

**{{session\_user\_name}}**

**{% endblock userName %}**

**{% block activeMyAds %}**

**active**

**{% endblock activeMyAds %}**

**{% block body %}**

**<div class="container">**

**<div class="row gy-5">**

**{% if result|length==0 %}**

**<div class="container alert p-5 alert-secondary text-center">**

**NO RESULT**

**</div>**

**{% else %}**

**{% for value in result %}**

**<div class="col-lg-4 col-md-5 col-sm-6" style="margin-bottom: 80px;width: 370px;height: 370px;">**

**<div class="card p-3" style="background: rgb(230, 227, 227);border-radius: 10px;">**

**<div style="display: flex;justify-content: center;align-items: center;width: 200px;height: 200px;margin: 0 auto;">**

**<img src="/static/uploads/{{value[10]}}" style="width: 200px;max-height: 200px;" alt="...">**

**</div>**

**<div class="card-body" >**

**<p class="card-title">**

**<span style="font-size: 1.2rem;font-weight: bold;">&#8377; {{value[6]}}</span>**

**<br>**

**{{value[4].upper()}}**

**</p>**

**<p class="card-text">**

**<i class="material-icons" style="font-size: 15px;">location\_on</i> {{value[0].capitalize()}}**

**<br>**

**<div style="display: flex;justify-content: space-between;">**

**<a class="btn btn-danger" href="/view/{{value[1]}}" >**

**View &#8649;**

**</a>**

**<div class="dropdown">**

**<button class="btn btn-success float-right dropdown-toggle" type="button" id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">**

**Action**

**</button>**

**<ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">**

**<!-- <li><a class="dropdown-item w3-hover-dark" href="#">Edit &#9935;</a></li> -->**

**<li><hr class="dropdown-divider"></li>**

**<li><a class="dropdown-item w3-hover-dark" href="/deletemyads/{{value[1]}}">Delete &#128465;</a></li>**

**</ul>**

**</div>**

**</div>**

**</p>**

**</div>**

**</div>**

**</div>**

**{% endfor %}**

**{% endif %}**

**</div>**

**</div>**

**<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-U1DAWAznBHeqEIlVSCgzq+c9gqGAJn5c/t99JyeKa9xxaYpSvHU5awsuZVVFIhvj" crossorigin="anonymous"></script>**

**{% endblock body %}**

**login.css**

**body {**

**width: 100vw;**

**height: 100vh;**

**overflow: hidden;**

**background-image: url('/static/background.jpg');**

**display: grid;**

**place-items: center;**

**}**

**.login {**

**width: 500px !important;**

**border-radius: 20px;**

**background-color: lightgray;**

**padding: 40px 40px;**

**box-shadow: 0px 4px 66px 0px rgba(0,0,0,0.75);**

**-webkit-box-shadow: 0px 4px 66px 0px rgba(0,0,0,0.75);**

**-moz-box-shadow: 0px 4px 66px 0px rgba(0,0,0,0.75);**

**}**

**.register {**

**width: 500px !important;**

**border-radius: 20px;**

**background-color: lightgray;**

**padding: 20px 40px;**

**box-shadow: 0px 4px 66px 0px rgba(0,0,0,0.75);**

**-webkit-box-shadow: 0px 4px 66px 0px rgba(0,0,0,0.75);**

**-moz-box-shadow: 0px 4px 66px 0px rgba(0,0,0,0.75);**

**}**

**.login > .title {**

**padding: 4px;**

**}**

**Profile.css**

**.profile-image {**

**/\* height: 250px;**

**width: 250px; \*/**

**width: 100%;**

**height: 100%;**

**/\* border-radius: 50%; \*/**

**}**

**.colLeft {**

**display: flex;**

**align-items: flex-end;**

**justify-content: center;**

**flex-direction: column;**

**}**

**.colLeft > form {**

**padding: 10px;**

**}**

**.colLeft > form > input {**

**margin-bottom: 10px;**

**}**

**.colLeft > form > button {**

**border-radius: 20px;**

**}**

**baseStyle.css**

**.w3-hover-dark:hover {**

**background:lightgray!important;**

**}**

**.profile\_link {**

**padding:5px 10px;**

**background: white;**

**color:black!important;**

**border-radius: 10px;**

**text-decoration: none;**

**text-transform: capitalize;**

**}**

**.profile\_link:hover {**

**background: rgb(19, 180, 19);**

**color: white !important;**

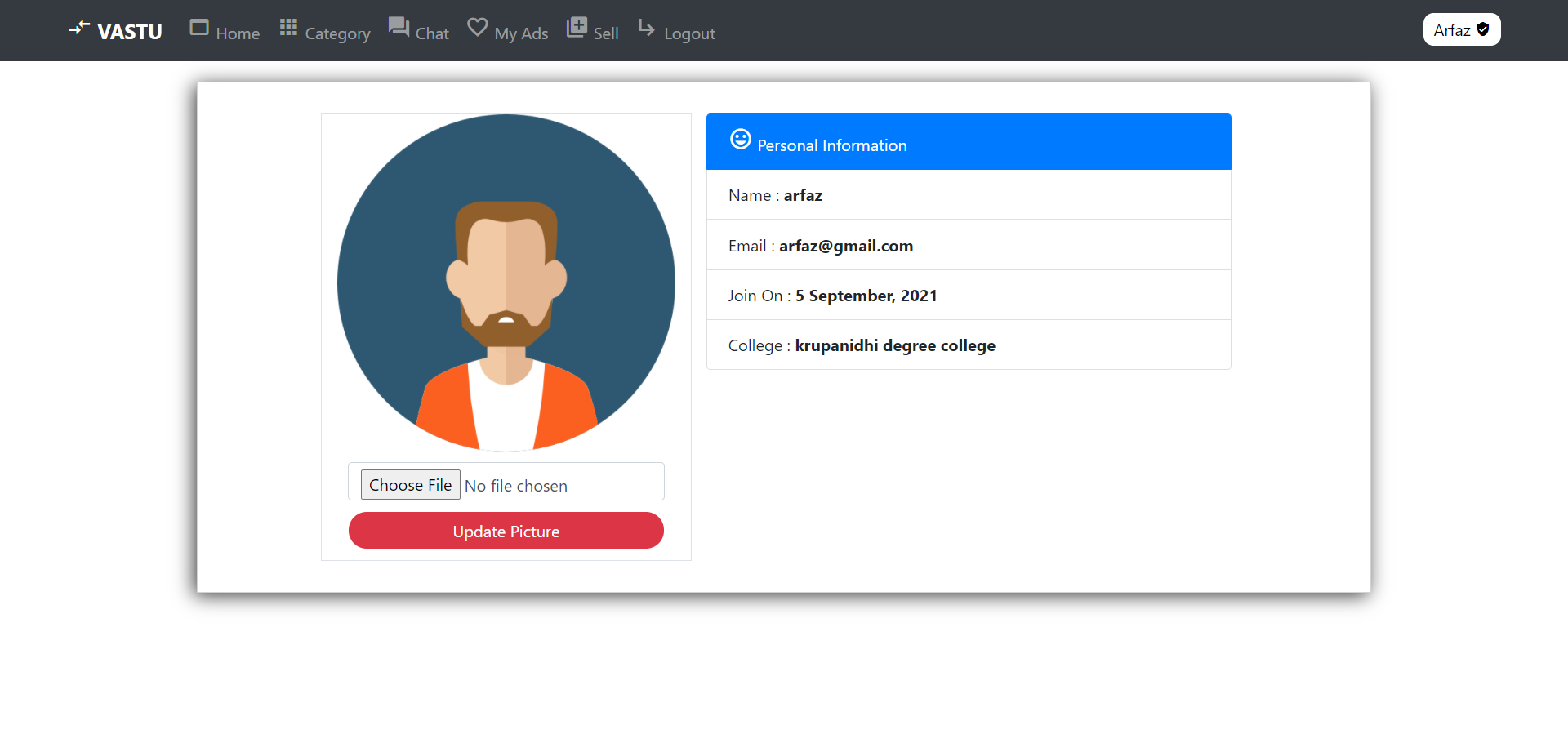
**text-decoration: none;**

**text-transform: capitalize;**

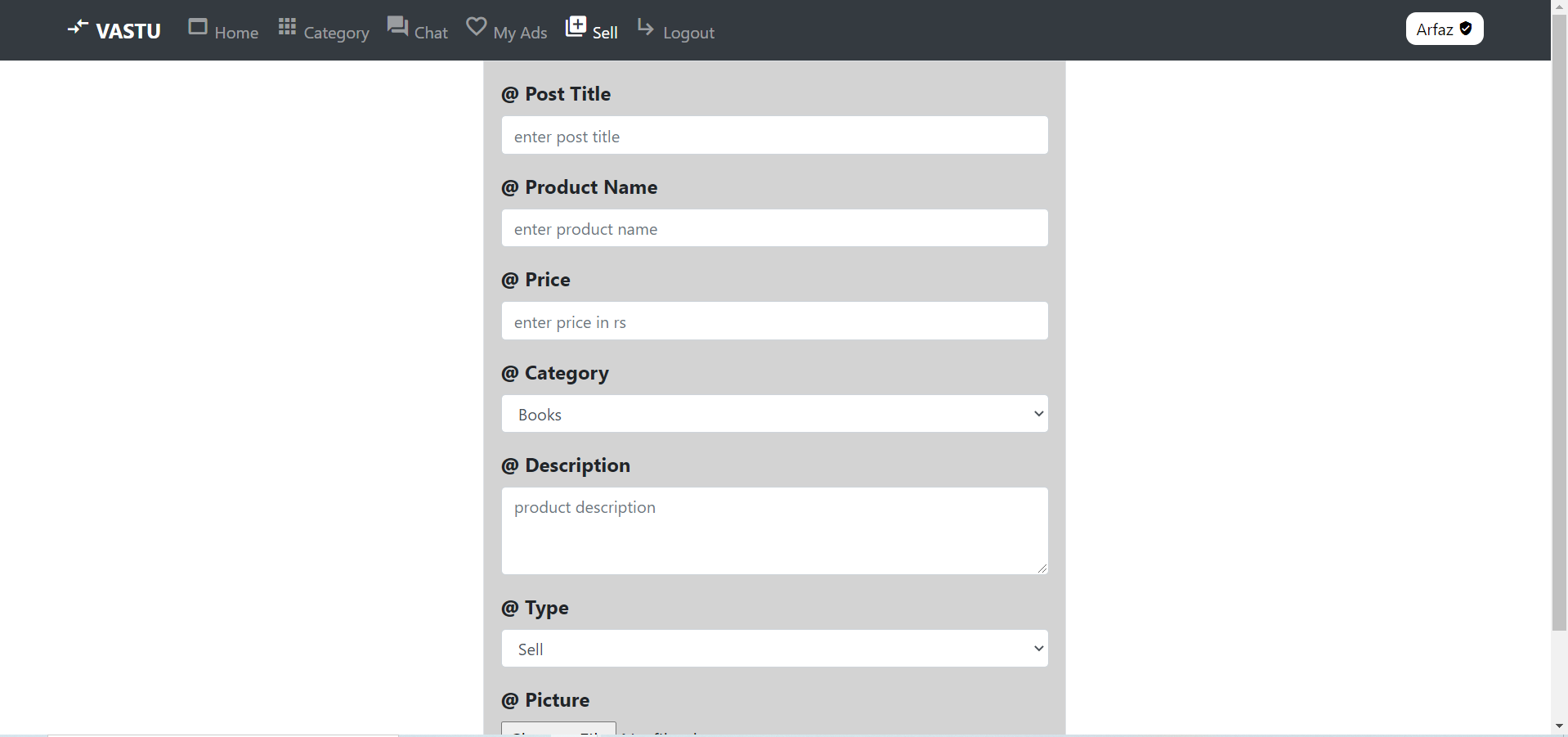
**}**

**3.5 SCREENSHOTS**

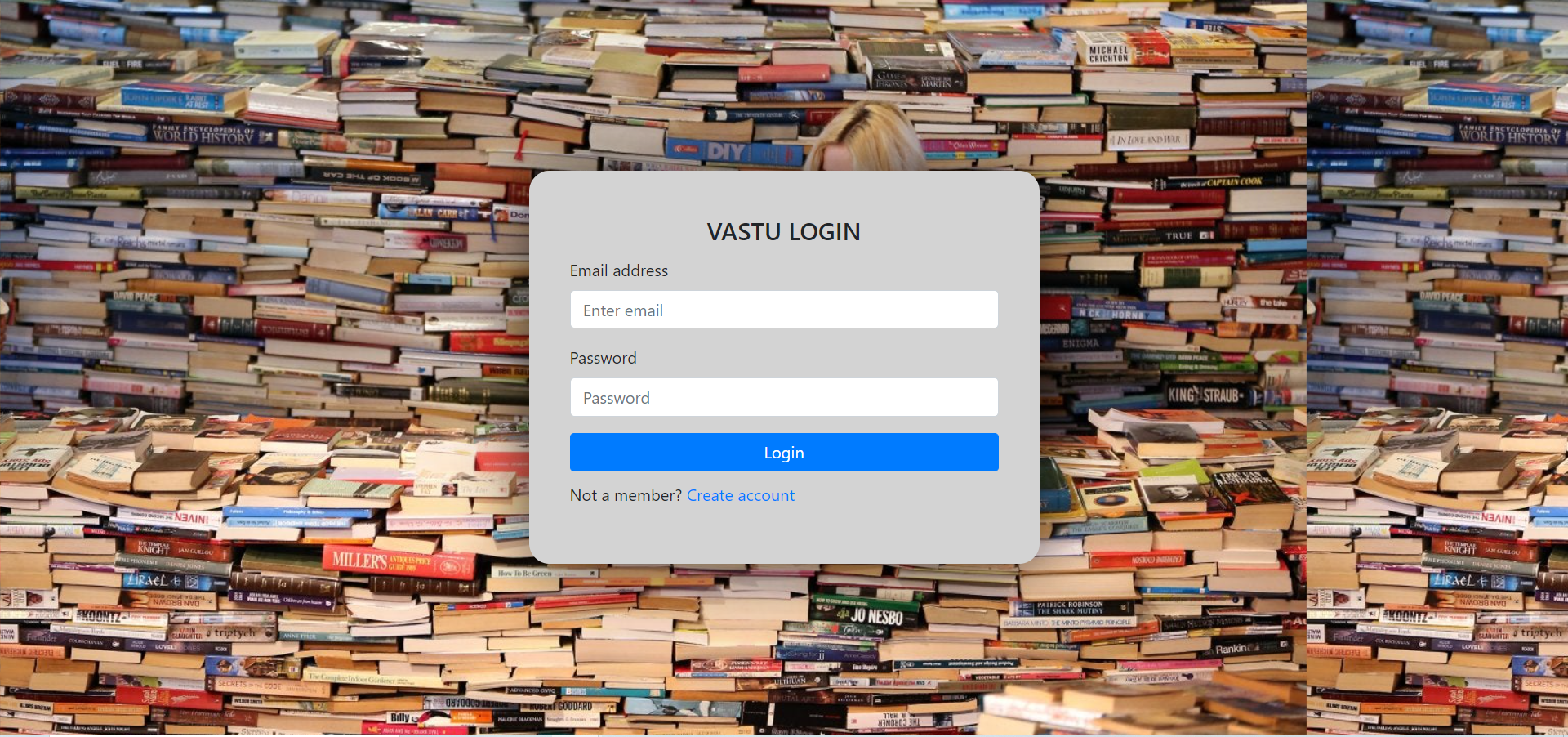
**Profile**

****

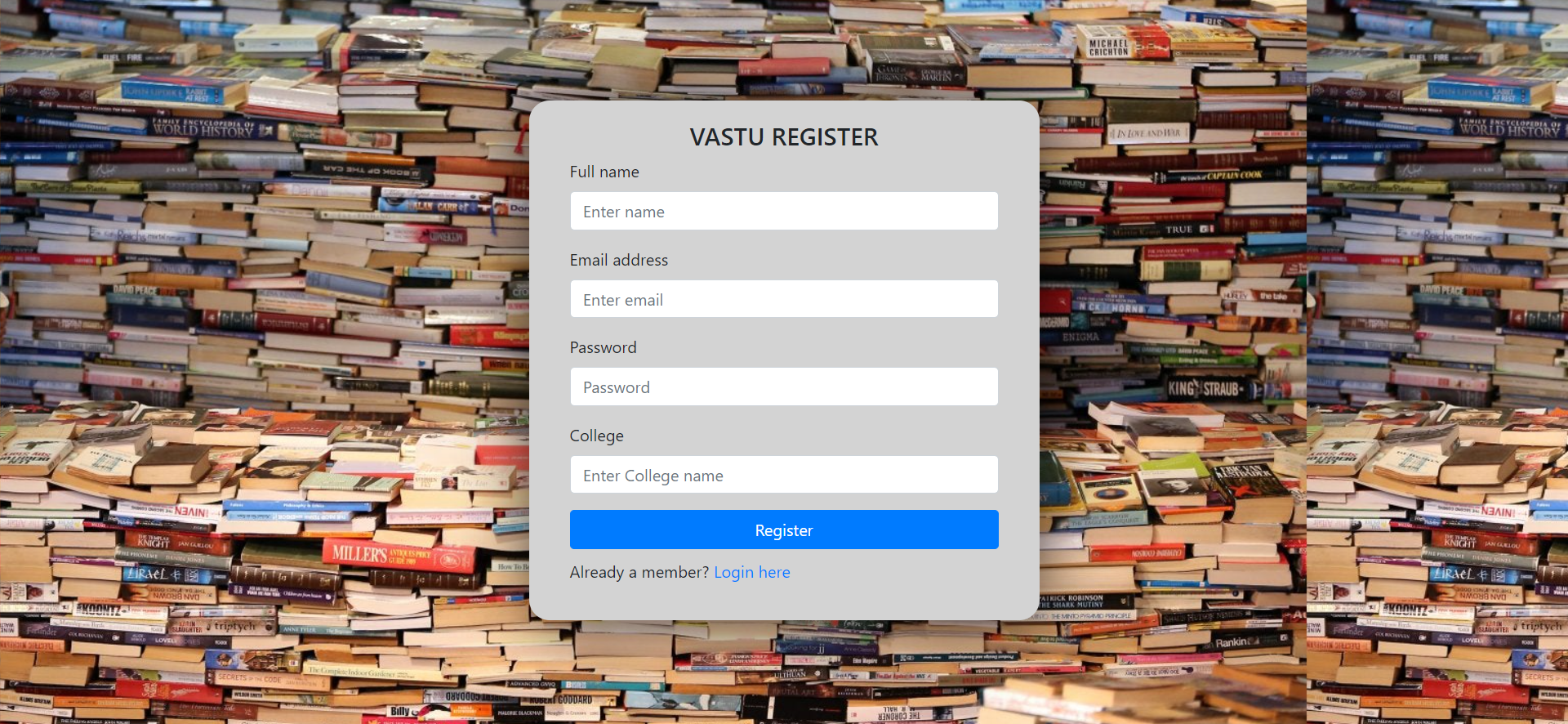
**Sell**

****

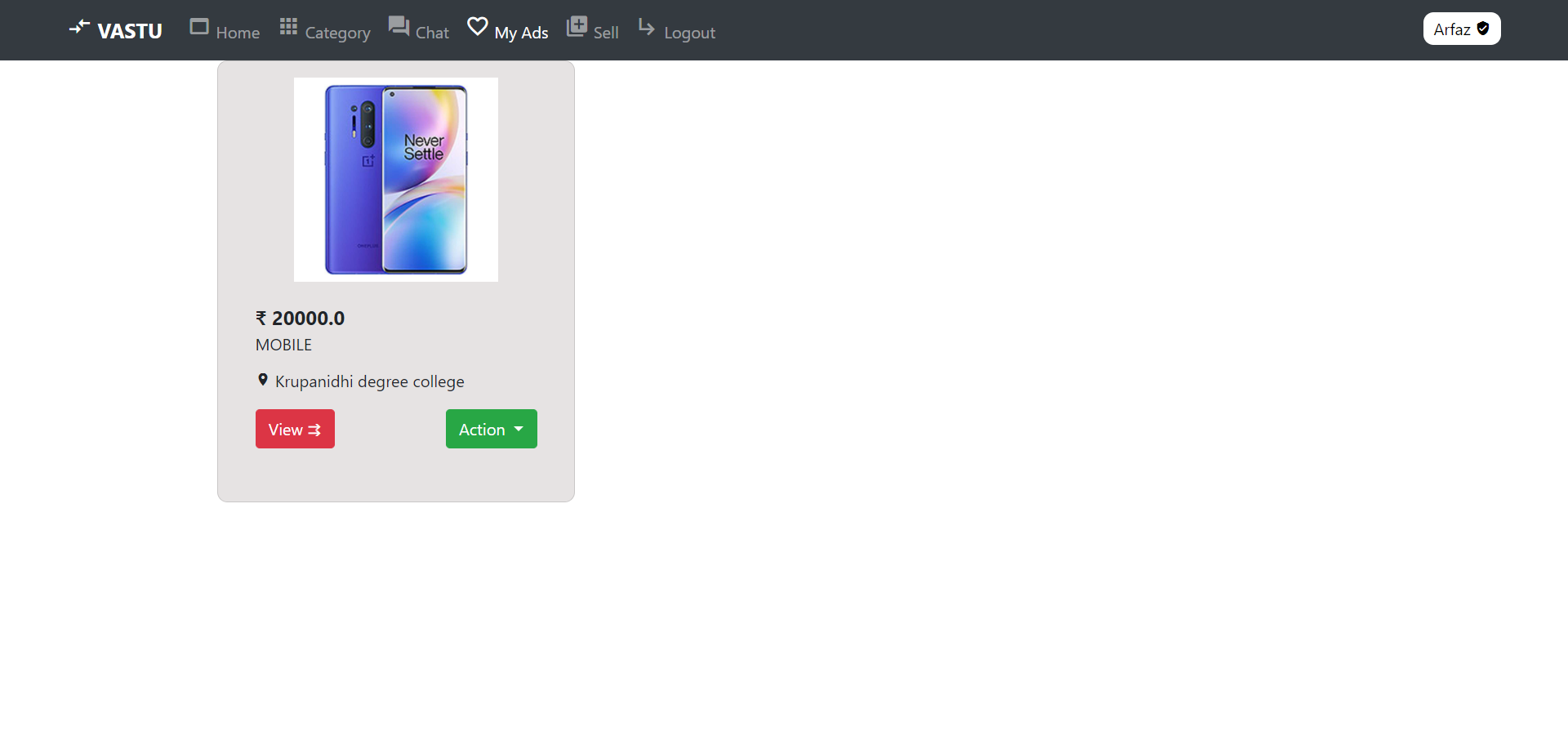
**Login**

****

**Register**

****

**Myads**

****

**CHAPTER - 4**

**SYSTEM IMPLEMENTATION**

**The process involved in Methodology**

The general methodology in developing a system is involved in different phases, which describe the systems life cycle, a model for developing software projects. The concept includes not only forward but also have the possibility to return that is cycled back to an activity previously completed. This cycle back or feedback may occur because of the failure with the system to meet a performance objective or because of changes in the redefinition of system activities.

**The requirement analysis phase** includes the identification of the problems, to identify the problem, we must know the information about the problem, the purpose of the evaluation for the problem is to be to know.

**System analysis phase** feasibility analysis the benefits of the various approaches and the determination of the alternative approaches methods like questionnaire’s an interview’s etc.

**Design phase**  Android is a process through which the requirements are translated into a representative of an android. One of the android requirements has been analyzed and specified, the s/w design involves three technical activities: design, coding, generation, and testing, The design of the system is in modular form i.e, the s/w is logically partitioned into components that perform functional characteristics.

**The development phase** includes a choice of a suitable s/w to solve the problem given. The various facilities and the sophistication in the selected s/w give a better development of the problem.

**Coding phased** is for translating the design of the system produced during the design phase into code in each programming, which can be executed by a computer and which performs the computation specified by the design.

**The testing phase** is done in various ways such as testing the algorithm, programming code, sample data debugging is also one of the following the above testing.

**4.1 SOFTWARE AND HARDWARE REQUIREMENTS**

**4.1.1 HARDWARE REQUIREMENTS**

* Processor - intel core i3-4005u (1.7GH and above)
* Ram - 4GB and above
* Hard disk - 500GB and above
* Device - ANY WINDOWS OR APPLE
* Space to execute - 80MB

**4.1.2 SOFTWARE REQUIREMENTS**

* FrontEnd - html5, css3, jquery, javascript, bootstrap
* Language - python
* Database - MySQL
* Technology Used - Visual studio code, sublime 3 text editor, windows 10
* Debugger - Any latest browser

**4.2 TECHNOLOGIES USED**

**4.2.1 JAVASCRIPT**

JavaScript (JS) is a lightweight, interpreted, or [just-in-time](https://en.wikipedia.org/wiki/Just-in-time_compilation) compiled programming language with [first-class functions](https://developer.mozilla.org/en-US/docs/Glossary/First-class_Function). While it is most well-known as the scripting language for Web pages, [many non-browser environments](https://en.wikipedia.org/wiki/JavaScript#Uses_outside_Web_pages) also use it, such as [Node.js](https://developer.mozilla.org/en-US/docs/Glossary/Node.js), [Apache CouchDB](https://couchdb.apache.org/), and [Adobe Acrobat](http://www.adobe.com/devnet/acrobat/javascript.html). JavaScript is a [prototype-based](https://developer.mozilla.org/en-US/docs/Glossary/Prototype-based_programming), multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

**4.2.2 BOOTSTRAP**

Bootstrap is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [CSS framework](https://en.wikipedia.org/wiki/CSS_framework) directed at responsive, [mobile-first](https://en.wikipedia.org/wiki/Responsive_web_design#Mobile_first,_unobtrusive_JavaScript,_and_progressive_enhancement) [front-end web development](https://en.wikipedia.org/wiki/Front-end_web_development). It contains [CSS](https://en.wikipedia.org/wiki/CSS)- and (optionally) [JavaScript](https://en.wikipedia.org/wiki/JavaScript)-based design templates for [typography](https://en.wikipedia.org/wiki/Web_design#Typography), [forms](https://en.wikipedia.org/wiki/Form_(HTML)), [buttons](https://en.wikipedia.org/wiki/Button_(computing)#HTML), [navigation](https://en.wikipedia.org/wiki/Web_navigation#Local_website_navigation), and other interface components.

**4.2.3 HTML5 AND CSS3**

**HTML5** is a [markup language](https://en.wikipedia.org/wiki/Markup_language) used for structuring and presenting content on the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web). It is the fifth and last[[3]](https://en.wikipedia.org/wiki/HTML5#cite_note-W3C_transfer_ZDNet-3) major [HTML](https://en.wikipedia.org/wiki/HTML) version that is a [World Wide Web Consortium (W3C)](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) recommendation. The current specification is known as the [HTML Living Standard](https://en.wikipedia.org/wiki/HTML#Transition_of_HTML_Publication_to_WHATWG). It is maintained by a consortium of the major browser vendors ([Apple](https://en.wikipedia.org/wiki/Apple_Inc.), [Google](https://en.wikipedia.org/wiki/Google), [Mozilla](https://en.wikipedia.org/wiki/Mozilla), and [Microsoft](https://en.wikipedia.org/wiki/Microsoft)), the [Web Hypertext Application Technology Working Group](https://en.wikipedia.org/wiki/Web_Hypertext_Application_Technology_Working_Group) (WHATWG).

**Cascading Style Sheets** (**CSS**) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language) such as [HTML](https://en.wikipedia.org/wiki/HTML).[[1]](https://en.wikipedia.org/wiki/CSS#cite_note-1) CSS is a cornerstone technology of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), alongside HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript).[[2]](https://en.wikipedia.org/wiki/CSS#cite_note-2)

CSS is designed to enable the separation of presentation and content, including [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface).[[3]](https://en.wikipedia.org/wiki/CSS#cite_note-3) This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple [web pages](https://en.wikipedia.org/wiki/Web_page) to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be [cached](https://en.wikipedia.org/wiki/Cache_(computing)) to improve the page load speed between the pages that share the file and its formatting.

**4.2.4 PYTHON**

**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Its design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects.[[30]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-7-30)

**4.2.4 JQUERY**

**jQuery** is a [JavaScript library](https://en.wikipedia.org/wiki/JavaScript_library) designed to simplify [HTML](https://en.wikipedia.org/wiki/HTML) [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) tree traversal and manipulation, as well as [event handling](https://en.wikipedia.org/wiki/Event_handling), [CSS animation](https://en.wikipedia.org/wiki/CSS_animation), and [Ajax](https://en.wikipedia.org/wiki/Ajax_(programming)).[[3]](https://en.wikipedia.org/wiki/JQuery#cite_note-jquery.com-3) It is [free, open-source software](https://en.wikipedia.org/wiki/Free_and_open_source_software) using the permissive [MIT License](https://en.wikipedia.org/wiki/MIT_License).[[4]](https://en.wikipedia.org/wiki/JQuery#cite_note-jqorg-license2-4) As of May 2019, jQuery is used by 73% of the 10 million most popular websites.[[5]](https://en.wikipedia.org/wiki/JQuery#cite_note-:0-5) [Web](https://en.wikipedia.org/wiki/World_Wide_Web) analysis indicates that it is the most widely deployed JavaScript library by a large margin, having at least 3 to 4 times more usage than any other JavaScript library.

**CHAPTER 5**

**SYSTEM SOFTWARE**

**5. SYSTEM TESTING**

**5.1 Introduction**

* In system testing the behavior of the whole system/product is tested as defined by the scope of the development project or product.
* It may include based on risks and or requirement specifications, business process, use-cases, or other high-level descriptions of system behavior, interactions with the operating system, and system resources.
* System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.
* System testing is carried out by specialist testers of independent testers.
* System testing should investigate both functional and non-functional requirements of the testing.

**5.2 UNIT TESTING**

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), unit testing is a [software testing](https://en.wikipedia.org/wiki/Software_testing) method by which individual units of [source code](https://en.wikipedia.org/wiki/Source_code)—sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures—are tested to determine whether they are fit for use. Unit tests are typically [automated](https://en.wikipedia.org/wiki/Test_automation) tests written and run by [software developers](https://en.wikipedia.org/wiki/Software_developer) to ensure that a section of an application (known as the "unit") meets its design and behaves as intended.[[2]](https://en.wikipedia.org/wiki/Unit_testing#cite_note-hamill-2) In [procedural programming](https://en.wikipedia.org/wiki/Procedural_programming), a unit could be an entire module, but it is more commonly an individual function or procedure. In [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming), a unit is often an entire interface, such as a class, but could be an individual method.[[3]](https://en.wikipedia.org/wiki/Unit_testing#cite_note-3) By writing tests first for the smallest testable units, then the compound behaviors between those, one can build up comprehensive tests for complex applications.[[2]](https://en.wikipedia.org/wiki/Unit_testing#cite_note-hamill-2)

To isolate issues that may arise, each [test case](https://en.wikipedia.org/wiki/Test_case) should be tested independently. Substitutes such as [method stubs](https://en.wikipedia.org/wiki/Method_stub), [mock objects](https://en.wikipedia.org/wiki/Mock_object),[[4]](https://en.wikipedia.org/wiki/Unit_testing#cite_note-mocksarentstubs-4) [fakes](https://en.wikipedia.org/wiki/Mock_object#Mocks.2C_fakes.2C_and_stubs), and [test harnesses](https://en.wikipedia.org/wiki/Test_harness) can be used to assist in testing a module in isolation.

During development, a software developer may code criteria or results that are known to be good, into the test to verify the unit's correctness. During test case execution, frameworks [log](https://en.wikipedia.org/wiki/Computer_data_logging) tests that fail any criterion and report them in a summary. For this, the most commonly used approach is test-function - expected value.

Writing and maintaining unit tests can be made faster by using [parameterized tests](https://en.wikipedia.org/wiki/Parameterized_test). These allow the execution of one test multiple times with different input sets, thus reducing test code duplication. Unlike traditional unit tests, which are usually closed methods and test invariant conditions, parameterized tests take any set of parameters. Parameterized tests are supported by [TestNG](https://en.wikipedia.org/wiki/TestNG), [JUnit](https://en.wikipedia.org/wiki/JUnit), and its .Net counterpart, [XUnit](https://en.wikipedia.org/wiki/XUnit). Suitable parameters for the unit tests may be supplied manually or in some cases are automatically generated by the test framework. In recent years support was added for writing more powerful (unit) tests, leveraging the concept of theories, test cases that execute the same steps, but using test data generated at runtime, unlike regular parameterized tests that use the same execution steps with input sets that are pre-defined.

**5.3 Integration testing**

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in [software testing](https://en.wikipedia.org/wiki/Software_testing) in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the [compliance](https://en.wikipedia.org/wiki/Regulatory_compliance) of a system or component with specified [functional requirements](https://en.wikipedia.org/wiki/Functional_requirement).[[1]](https://en.wikipedia.org/wiki/Integration_testing#cite_note-1) It occurs after [unit testing](https://en.wikipedia.org/wiki/Unit_testing) and before [validation testing](https://en.wikipedia.org/wiki/Software_verification_and_validation). Integration testing takes as its input [modules](https://en.wikipedia.org/wiki/Module_(programming)) that have been unit tested, groups them in larger aggregates, applies tests defined in an integration [test plan](https://en.wikipedia.org/wiki/Test_plan) to those aggregates, and delivers as its output the integrated system ready for [system testing](https://en.wikipedia.org/wiki/System_testing)**.**

**5.4 SYSTEM TESTING**

**System testing** is testing conducted on a complete integrated system to evaluate the system's compliance with its specified [requirements](https://en.wikipedia.org/wiki/Requirements).[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

System testing takes, as its input, all of the integrated components that have passed [integration testing](https://en.wikipedia.org/wiki/Integration_testing). The purpose of integration testing is to detect any inconsistencies between the units that are integrated (called *assemblages*). System testing seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] The actual result is the behavior produced or observed when a component or system is tested.[[1]](https://en.wikipedia.org/wiki/System_testing#cite_note-1)

System testing is performed on the entire system in the context of either [functional requirement](https://en.wikipedia.org/wiki/Functional_requirements) specifications (FRS) or [system requirement](https://en.wikipedia.org/wiki/Requirements_analysis) specification (SRS), or both. System testing tests not only the design but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software or hardware requirements specification(s).

**5.5 WHITE BOX TESTING**

**White Box Testing** is a software testing technique in which internal structure, design, and coding of software are tested to verify the flow of input-output and to improve the design, usability, and security. In white-box testing, code is visible to testers so it is also called Clear box testing, Open box testing, transparent box testing, Code-based testing, and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

**5.6 BLACK BOX TESTING**

Black-box testing is a method of [software testing](https://en.wikipedia.org/wiki/Software_testing) that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: [unit](https://en.wikipedia.org/wiki/Unit_test), [integration](https://en.wikipedia.org/wiki/Integration_testing), [system](https://en.wikipedia.org/wiki/System_testing), and [acceptance](https://en.wikipedia.org/wiki/Acceptance_test). It is sometimes referred to as specification-based testing.

**5.7 Verification and validation**

Tests were performed to find conformity with the requirements. The software was alpha-tested. There are two goals in preparing test plans.

Firstly, a properly detailed test plan demonstrates that the program specifications are understood completely. Secondly, the test plan is used during program testing to prove the correctness of the program.

**CHAPTER - 6**

**CONCLUSION**

While developing the system a conscious has been made to create and develop a software package, making use of available tools, techniques, and resources - that would generate a proper system for the **VASTU ONLINE MARKET PLATFORM.**

While making the system an eye has been kept on making it as user-friendly. As such one may hope that the system will be acceptable to any user and will adequately meet his/her needs. As in the case of any system development process where there are several shortcomings, there have been some shortcomings in the development of this system also.

**CHAPTER - 7**

**BIBLIOGRAPHY**

* <https://en.wikipedia.org>
* <http://stackoverflow.com/>
* [www.w3schools.com](http://www.w3schools.com)
* www.google.com