# About

Vastu is an online marketplace, a platform only for students where they can buy or sell items such as books, electronics, fashion items etc. Sometimes when students want to move out of college or want to find things cheaper than the normal rate, they can usually look up the site.A student can list his/her item for sale and can also offer other students his service or abilities to good use. The only requirement to register into this platform is to be able to showcase an id given by the college to approve his/her identity.

The students can also offer their services on the platform to earn money as a side hustle.A student can offer a negotiable amount via a chat option.

Privacy and bad behavior would also be monitored for safety reasons.

There are many categories to choose from and complete transparency towards which college the product is coming from is maintained.

# Technology

**Frontend** **-** Html, Css, Javascript, Jquery, Bootstrap

**Backend -** Python, Flask framework

**Database -** Mysql

**Hardware -** i3 processor, 4gb ram, 100gb hard disk

**Software -** Xampp, Visual studio code, any modern browser

# How to run

**Step1 :** Open xampp application and start **Apache** and **MySQL**

**Step2 :** Go to visual studio code and open terminal

**Step3 :** type in the terminal “**.\env\Scripts\activate.ps1**” and press enter

**Step4 :** To start the application, type **python app.py** and press enter

**Step5 :** Go to the browser and type [**http://localhost:5000**](http://localhost:5000)

**Step6 :** Your application is ready

# 

# Important imports in flask

**render\_template -** it renders html file from template folder

**request -** it is used to receive post or get request from form fields

**session -** It is used for creating and deleting a session which is in fact used in login and logout functionalities

**redirect -** it is used to redirect to another page

**jsonify -** It is used to convert normal values to json statement which is used in the showchat page where the page uses ajax post to send and receive message

**flash -** it is not used in the application, so no idea.

**secure\_filename -** Pass it a filename and it will return a secure version of it

ex - secure\_filename("My cool movie.mov")

output - 'My\_cool\_movie.mov'

**mysql.connector** - It is used to connect flask applications to the mysql database.

# Important terms in flask, python

**Endpoints -** the "endpoint" is an identifier that is used in determining what logical unit of your code should handle the request.

example - <http://localhost:5000/login>

**Templates Folder -** it contains all the html files which needs to be displayed on the browser

**Static Folder** - The static folder contains assets used by the templates, including CSS files, JavaScript files, and images.

# Important questions in MySQL

**Q: What is the name of the database?**

**->**  Vastu. It has four tables - 1) users 2) products 3) chat 4) messages

**Q: What is a cursor in a flask?**

cursor allows Python code to execute SQL command in a database session

In the application, **cursor.execute** and **cursor.fetchall** is used.

**cursor.execute() -**  It is used to execute sql commands such as **Select, Update, Delete and Insert.**

**cursor.fetchall() -** It is used to fetch data from a database where the command is **Select.**

**Note :** When sql commands **Update, Delete or Insert** are executed, We also use the **mydb.commit()** command to commit the changes. **mydb** is the connection variable to mysql.

# Important questions in flask

**Q: How to put a python variable in html?**

-> {{variable}} - we use double curly bracket and insert the variable to view the variable send by the variable in **render\_template()**

**Q: How to put python if else in html**

**->** { % if condition %}

{% endif %}

**Q: How do we use python code in plain html?**

-> We use **jinja** extensions to add if, if else, for loops.

# Template pages

**Login.html** - It contains login functionality.

**register.html** - It contains register functionality.

**index.html** - It contains the products uploaded by other student users where you have the functionality of viewing and offering a negotiation to the other user.

**category.html** - It contains the links to a list of categories that the students can upload on that particular one.

**chat.html** - It contains the chats the current logged in user has, either it be his product or the products that he was interested in and initiated the offer.

**myads.html** - It contains a list of products uploaded by the logged in user.

**sell.html** - It contains input form that is used to submit products by the logged in user.

**profile.html** - It contains the information about the logged in user such as name, email, college, date joined and profile picture which the logged in user can update.

**view.html** - It contains the information about a certain product, basically a viewing page for the product either by the logged in user or other users.

**showcategory.html** - After clicking the link from the category.html page, it comes to this page and shows the category specified by the user.

**showchat.html** - This page contains messages of a specific product of a user and the buyer.

**base.html** - It is the base file of most of the page as it contains bootstrap css and javascript files.

# Flask Endpoints

**localhost:5000/login** - login functionality is achieved through this endpoint.

**localhost:5000/logout** - User logouts by destroying the session created when it was logged in or registered.

**localhost:5000/register** - A user can register and redirect to the home page.

**localhost:5000** - we execute the sql command **select \*** to fetch all the products that other users have posted excluding the post that the logged user has posted.

**localhost:5000/profile** - A user fetches the information regarding his profile.

**localhost:5000/profileimage** - this endpoint is used to upload profile image to the database.

**localhost:5000/category** - It displays the category.html page.

**localhost:5000/showcategory/books** - It fetches the specified category from the database.

**localhost:5000/myads** - it fetches the products uploaded by the logged in user only.

**localhost:5000/deletemyads/id** - It deletes the products uploaded by the logged in user through the products id.

**localhost:5000/view/product\_id** - It fetches the information from the database only related to a certain product.

**localhost:5000/sell** - It displays the sell.html page.

**localhost:5000/postsubmit** - It uploads new products given by the user to the database.

**localhost:5000/chat** - It fetches chat information of a user.

**localhost:5000/offer -** it initiates a new conversation between the user and other user.

**localhost:5000/showchat/chat\_id** - It fetches messages about a certain product initiated between two users.

**localhost:5000/fetchMessage** - it is used to fetch messages continuously from the database which is done by jquery ajax.

**localhost:5000/sendMessage** - it uploads a message to the database.

# Other important files used

**Jquery** - It is imported only to use its feature called ajax. It sends POST AND GET requests to the server without refreshing the page.It is used in the message system where the user does not have to refresh the page again and again to fetch the message.Also while sending the message, the page won't reload as the ajax works in the background.

**Bootstrap** - Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.

**Material-icons** - We used this module to insert icons into our pages.

# Important Processes

**Q: How does the image upload process work?**

-> First we upload the picture taken from the form field to the directory static/upload. As soon as the directory contains the picture, Next step we take the pictures filename and upload it to the database. So when we want to access or display the picture onto our page, we could easily navigate through the directory using the filename stored in the database.

**Q: How does the login functionality work?**

-> First we fetch data from the database which corresponds to the user's email and password. If any data is sent back from the database, we create a session and log the user in. If no data is found, we show the user a message saying, no such user exists.

**Q: How does the register functionality work?**

-> First we check if the email is already present in the database or not, if it is not present. We insert the new data into the database else we show the user, it is already present.

**Q: How does the chat functionality work?**

-> It works just like a postman. The server checks for a new message from the database every 5 seconds. If any other message appears, it displays onto the page. We use javascript **setInterval()** to keep calling the fetch message function every five seconds and display it on the page. It works without us refreshing the page because we use **ajax** functionality on the page where the programmer has the ability to send **POST** or **GET** requests to the server without refreshing itself.

**Q: How does the logout functionality work?**

-> It works by destroying the sessions which are created when a user logs in or registers itself.

**Q: How is the data validated in the login or any form based?**

-> Basic validation is used. It just checks if the user has left the input empty or not.

# Possible Question On Project

**Q: How the application will validate the user’s identity whether he is from a college or not?**

-> No such validation has been provided as of now. The only validation till now would be to see the information when the user registers and enter’s his or her college name.It is surely a machine learning or human power work as it requires a precision to see the college id and validate if it is fake or not.

**Q: Explain the ER diagram?**

-> The database contains 4 tables which are users, products, chat and messages.

-> Users can have multiple products so the relationship between users and products would be one to many.

-> Users can have multiple chats so the relationship between users and chat would be one to many.

-> A product can have multiple chats as many users would be wanting to offer so the relationship between products would be one to many.

-> A chat can have multiple messages so the relationship between chat and messages would be one to many.

**Q: What are the future updates possible on this applications?**

-> A future update would be to have a field in the register where a user has to upload his identity card provided by the college to validate if he is a student or not.

-> Another update would be to have another functionality of exchange where the students can exchange each other's items.

**Q: What is flask?**

-> Flask is a lightweight web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.