**15CSE387 – Open Lab Python – Flask Assignment**

**Aadhith S.**

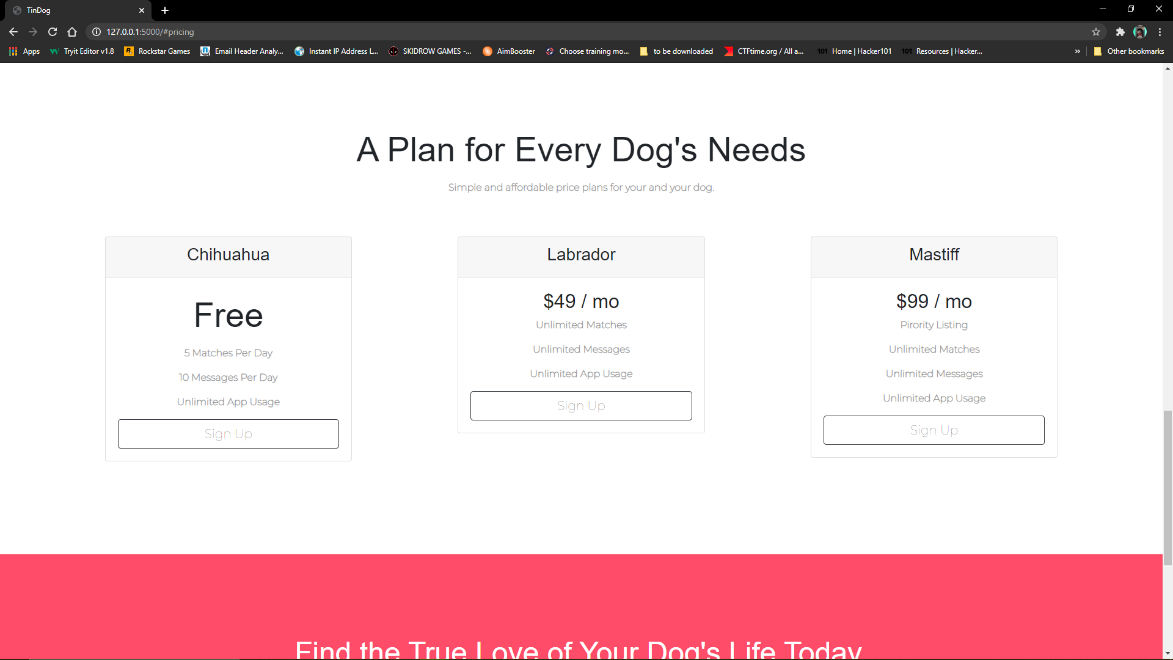
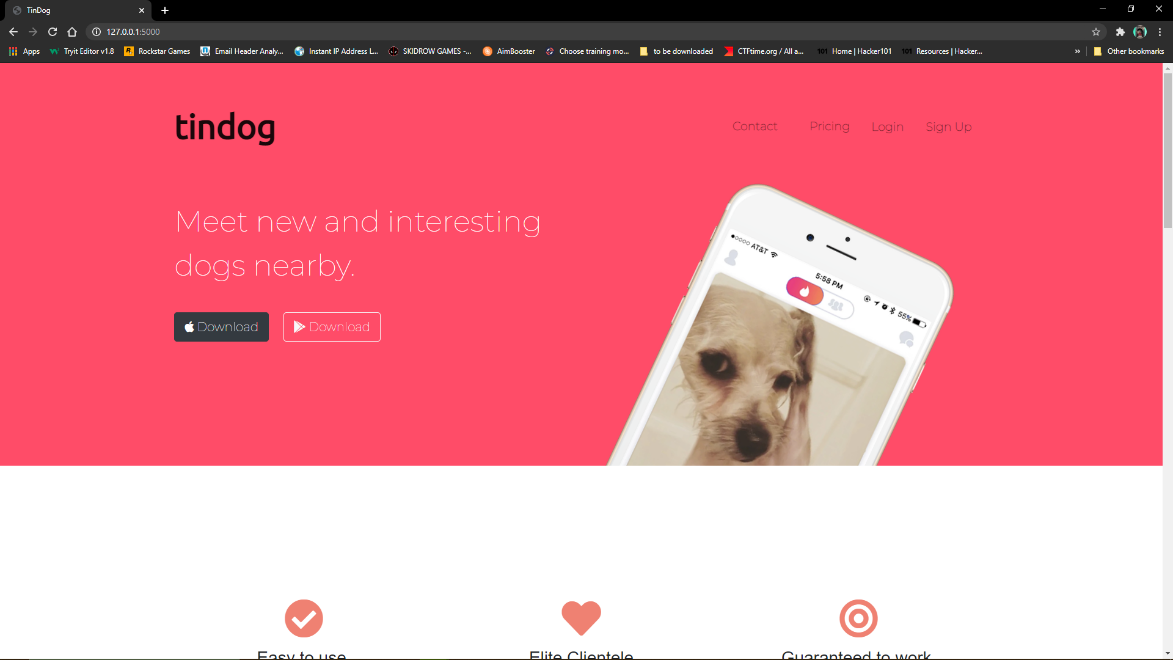
**CB.EN.U4CSE18101**

**Description:**

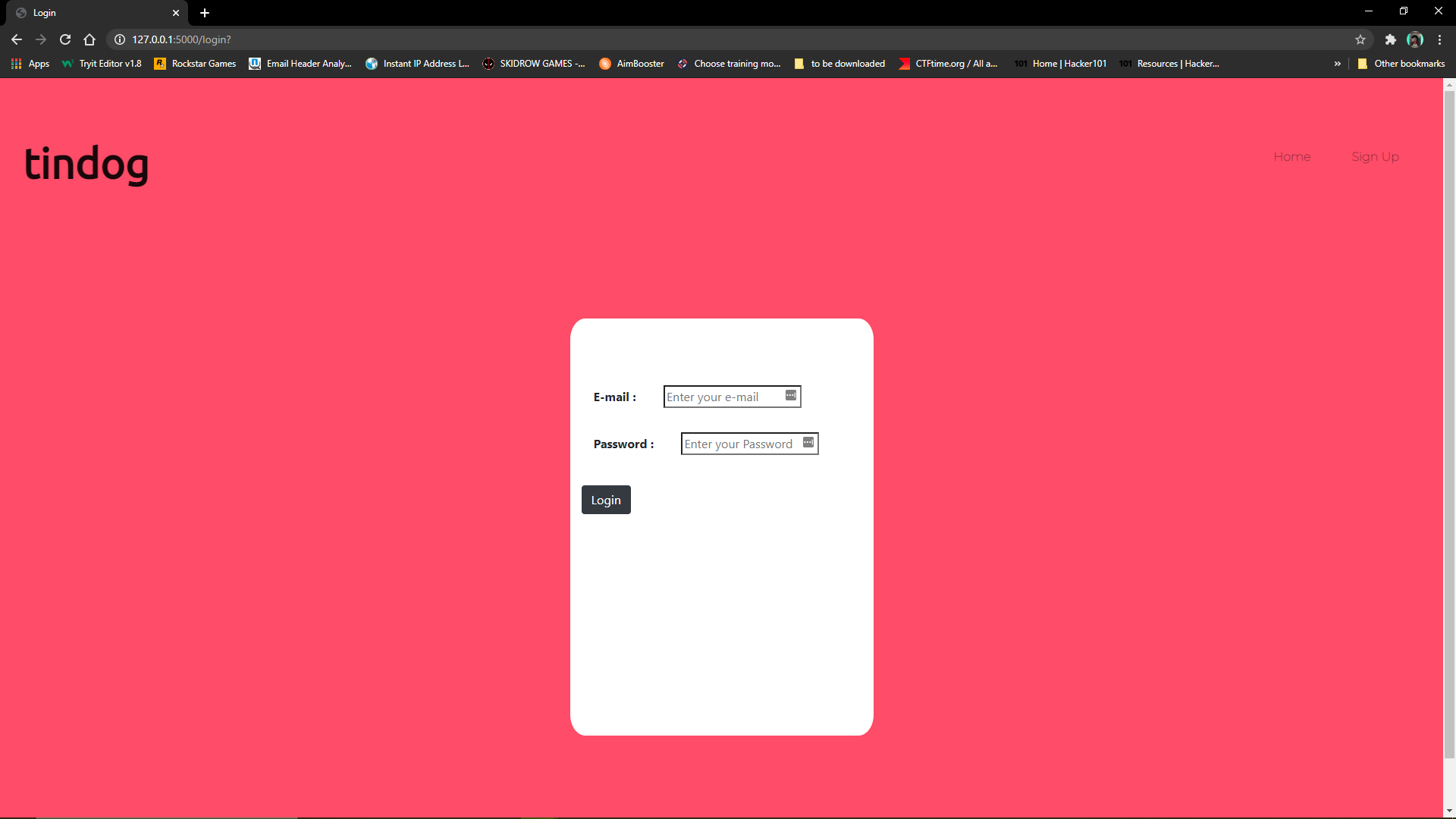
Tindog® – A popular dating app for your dogs, making it the place to help your dog can meet new friends and also to help you to meet new people too 😉.

**Screenshots:**

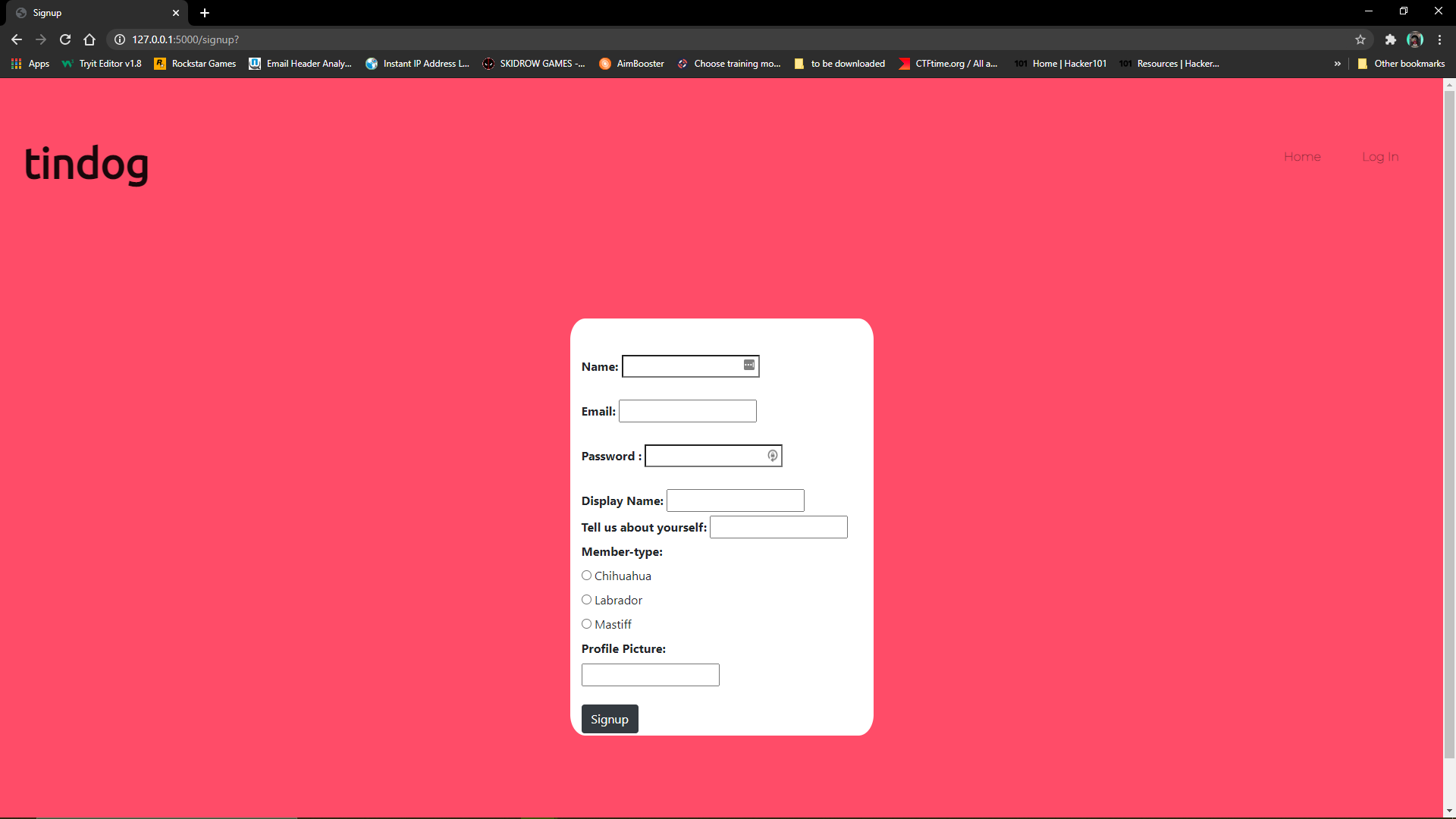
Home Page

****

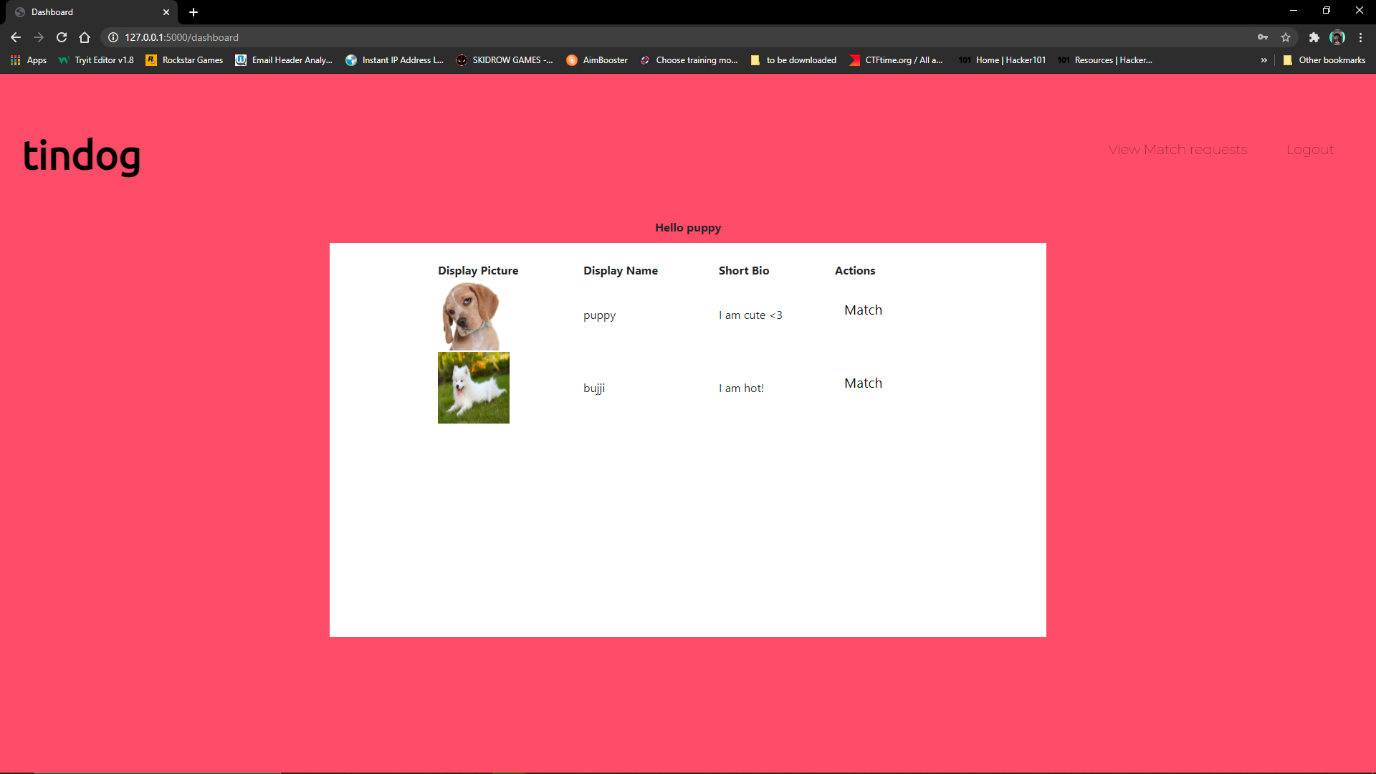
Login



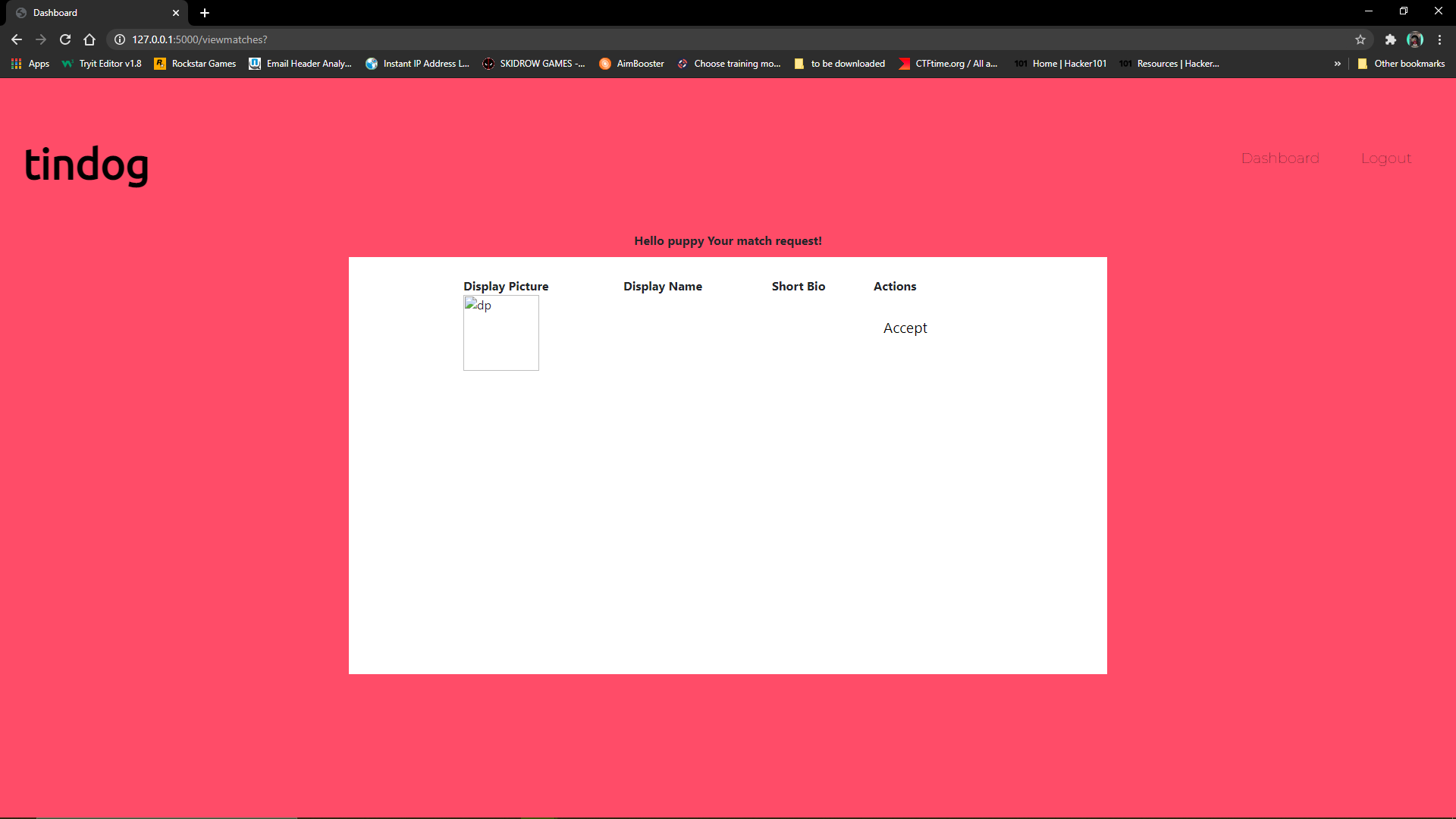
Signup



Dashboard



View Match requests



**Each route and it’s functionality:**

1. ‘/’

This is the home page. Users can view some details about our website, pricing, plans, user reviews (one from our special dogs too 😊). It also has links to Log in and Sign up.

1. ‘/login’

This is the login page for the website. When the user enters his/her details and presses submit button, it verifies the details of the user by checking the value with the Database Database (user login details db) and redirects to ‘/dashboard’

1. ‘/signup’

This is the login page for the website. When user enters his/her details and presses submit button, it checks whether a user with the email exists, if so, throws an error on the HTML page else updates new details in Database Database and redirects to’/’

1. ‘/dashboard’

Once a user verifies his/her credentials, they will be redirected here. Here they can browse for new matches (where data will be sent from backend by querying the Database Database) or they can ‘vie match request’ which will redirect to ‘/viewmatches’

1. ‘/logout’

This will empty the session and redirect to ‘/’

1. ‘/match’

This will be invoked when the user presses ‘Match’ button near the profile they are browsing.

1. ‘/viewmatches’

This will take to the page where we can see the match request that others gave us.

1. ‘/acceptmatch’

This will be invoked when the user presses ‘Accept’ in the view match page and it will reflect the accepted change in the Database MatchDB

1. Database Database

This contains the user details of our customer, **Dogs.**

1. Database MatchDB

This contains the match request of users with their accepted status(yes/no)

**Note before running the program:**

**Required dependencies:**

1. Python3.
2. Flask.
3. Pytz in Python.
4. Datetime in Python.
5. SQLAlchemy.
6. Browser that supports HTML5 and CSS3 with internet connection to get the required fonts and BootStrap.

**Commands to run:**

A virtual environment has been created named ‘env’ and it is advised to run under it.

1. If login.db file is missing, please type ‘python’, ‘from app import db’, ‘db.create\_all()’. Each commands in ‘’ are to run on separate lines.
2. Go to the source folder.
3. Open CMD/Powershell.
4. Run ‘.\env\Scripts\activate.bat’.
5. Run ‘Python ./app.py’.
6. Default host is @127.0.0.1:5000 .
7. You are good to go! 😊.

**Server side code:**

from flask import \*

from flask\_sqlalchemy import SQLAlchemy

from datetime import datetime

import pytz

email=None

app = Flask(\_\_name\_\_, template\_folder='template')

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///login.db'

db = SQLAlchemy(app)

class Database(db.Model):

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

    name = db.Column(db.String(50), nullable=False)

    email = db.Column(db.String(50), nullable=False)

    password = db.Column(db.String(50), nullable=False)

    display\_name = db.Column(db.String(50), nullable=False)

    short\_bio = db.Column(db.String(300), nullable=False)

    member\_type = db.Column(db.String(50), nullable=False)

    profile\_pic = db.Column(db.String(500), nullable=False)

    date\_created = db.Column(db.DateTime, default = datetime.now(pytz.timezone('Asia/Kolkata')))

    def \_\_repr\_\_(self):

        return '<Account %r>' %self.id

class MatchDB(db.Model):

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

    sender = db.Column(db.String(50), nullable=False)

    reciever = db.Column(db.String(50), nullable=False)

    date\_created = db.Column(db.DateTime, default = datetime.now(pytz.timezone('Asia/Kolkata')))

    accept = db.Column(db.Boolean, default=False)

    def \_\_repr\_\_(self):

        return '<Match %r>' %self.id

@app.route('/', methods=['GET'])

def index():

    return render\_template('index.html')

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

def login():

    error = None

    global email

    if request.method == 'POST':

        if request.form['email'] and request.form['password']:

            query = Database.query.filter\_by(email = request.form['email']).first()

            if query is None:

                error = 'Invalid Credentials. Please try again.'

            elif request.form['email'] == query.email and request.form['password'] == query.password:

                email=query.email

                return redirect('/dashboard')

            else:

                error = 'Invalid Credentials. Please try again.'

        else:

            error = 'Enter your E-mail and Password'

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

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

def signup():

    error = None

    if request.method == 'POST':

        query = Database.query.filter\_by(email = request.form['email']).first()

        if query is None:

            temp\_name=request.form['name']

            temp\_email=request.form['email']

            temp\_password=request.form['password']

            temp\_disp=request.form['display\_name']

            temp\_shortbio=request.form['shortbio']

            temp\_mt=request.form['member\_type']

            temp\_profilepic=request.form['profile\_pic']

            new\_acc = Database(name=temp\_name,  email=temp\_email, password=temp\_password, display\_name=temp\_disp, short\_bio=temp\_shortbio, member\_type=temp\_mt, profile\_pic=temp\_profilepic)

            try:

                db.session.add(new\_acc)

                db.session.commit()

                return redirect('/')

            except:

                error='Cannot sign up!'

        else:

            error = 'Account already exists!'

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

@app.route('/dashboard')

def dashboard():

    global email

    query = Database.query.filter\_by(email = email).first()

    if query is not None:

        tasks = Database.query.order\_by(Database.id).all()

        return render\_template('dashboard.html', email=query.email, disp = query.display\_name, items=tasks)

    else:

        return 'Illegal Access!'

@app.route('/logout')

def logout():

    email=None

    return redirect('/')

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

def match():

    global email

    query = Database.query.filter\_by(email = email).first()

    tasks = Database.query.order\_by(Database.id).all()

    error=None

    match\_from=email

    match\_to=request.form['match']

    new\_match=MatchDB(sender=match\_from, reciever=match\_to)

    try:

        db.session.add(new\_match)

        db.session.commit()

    except:

        error='Cannot Match!'

    return render\_template('dashboard.html',error=error, disp = query.display\_name, items=tasks)

@app.route('/viewmatches')

def viewmatch():

    global email

    error=None

    query1 = MatchDB.query.filter\_by(reciever = email).all()

    query = Database.query.filter\_by(email = email).first()

    if query1 is not None:

        error=None

    else:

        error='No matches found! :('

    return render\_template('matches.html', disp = query.display\_name, items=query1, error=error)

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

def acceptmatch():

    global email

    error=None

    query = Database.query.filter\_by(email = email).first()

    query1 = MatchDB.query.filter\_by(reciever = email).all()

    tasks = Database.query.order\_by(Database.id).all()

    if request.method == 'POST':

        t\_id=request.form['match']

        task = MatchDB.query.filter\_by(id=t\_id).first()

        if task is not None:

            task.accept = True

            try:

                db.session.commit()

            except:

                error="Cant update! Try again!"

    return render\_template('dashboard.html', disp = query.display\_name, items=tasks, error=error)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)