1. Student Details

- **Name:** Veera Venkata Gopi Naga Manikanta Sai Paladugu
- **Roll Number:** 241U1R3023
- **Gender:** Male
- **Department:** CSE - Data Science
- **University:** Aurora University
- **Email:** veeravenkatagopinagamanikantasai.paladugu@aurora.edu.in
- **Contact:** 9346912398

2. Title of the Project

Blog Website — A Blogging Platform

3. Topic of the Project

Web Application Development using Python Flask Framework

4. Purpose of Implementation

- Design and implement a full-stack blogging website.
- Learn backend development using Python Flask.
- Implement User Registration, Login, Logout functionalities.
- Apply CRUD operations (Create, Read, Update, Delete) on blog posts.
- Integrate SQLite database for data persistence.
- Apply secure Password Hashing using werkzeug.security.
- Host and deploy a fully functional web application using Render Cloud Platform.

5. Code Used in the Project

Technologies Used:

- **Frontend:** HTML5, CSS3, Bootstrap 5
- **Backend:** Python 3.11, Flask Framework
- **Database:** SQLite3
- **Deployment:** Render.com (Cloud Hosting)
- **Version Control:** Git, GitHub

Libraries Used:

- Flask

- Flask-SQLAlchemy
- Flask-WTF
- Flask-Login
- Werkzeug

6. How to Run This Project

Step 1: Clone or Download the Project

git clone https://github.com/yourusername/blog_project.git

cd blog_project

Step 2: Create Virtual Environment

python -m venv venv

source venv/bin/activate # (Mac/Linux)

venv\Scripts\activate # (Windows)

Step 3: Install Dependencies

pip install -r requirements.txt

Step 4: Run the Application

python app.py

Step 5: Open Web Browser

Visit: http://127.0.0.1:5000

7. Code of the Program

```
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, TextAreaField
from wtforms.validators import DataRequired, Length, EqualTo
from werkzeug.security import generate_password_hash, check_password_hash


app = Flask(__name__)
```

```
app.config['SECRET_KEY'] = 'secretkey'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///blog.db'
db = SQLAlchemy(app)

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True, nullable=False)
    password = db.Column(db.String(150), nullable=False)

class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(150), nullable=False)
    content = db.Column(db.Text, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

class RegisterForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=3,
max=20)])
    password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])
    confirm_password = PasswordField('Confirm Password', validators=[DataRequired(),
EqualTo('password')])
    submit = SubmitField('Register')

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired()])
```

```
    submit = SubmitField('Login')

class PostForm(FlaskForm):
    title = StringField('Title', validators=[DataRequired()])
    content = TextAreaField('Content', validators=[DataRequired()])
    submit = SubmitField('Submit')

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

@app.route('/')
def index():
    posts = Post.query.all()
    return render_template('index.html', posts=posts)

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegisterForm()
    if form.validate_on_submit():
        hashed_pw = generate_password_hash(form.password.data)
        user = User(username=form.username.data, password=hashed_pw)
        db.session.add(user)
        db.session.commit()
        flash('Account created successfully!')
        return redirect(url_for('login'))
    return render_template('register.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 and check_password_hash(user.password, form.password.data):
            login_user(user)
            return redirect(url_for('index'))
        else:
            flash('Login Failed!')
    return render_template('login.html', form=form)

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('index'))

@app.route('/add', methods=['GET', 'POST'])
@login_required
def add():
    form = PostForm()
    if form.validate_on_submit():
        post = Post(title=form.title.data, content=form.content.data, user_id=current_user.id)
        db.session.add(post)
        db.session.commit()
        return redirect(url_for('index'))
    return render_template('add.html', form=form)

@app.route('/edit/<int:post_id>', methods=['GET', 'POST'])
```

```
@login_required
def edit(post_id):
    post = Post.query.get_or_404(post_id)
    if post.user_id != current_user.id:
        flash('Not authorized to edit this post.')
        return redirect(url_for('index'))
    form = PostForm(obj=post)
    if form.validate_on_submit():
        post.title = form.title.data
        post.content = form.content.data
        db.session.commit()
        return redirect(url_for('index'))
    return render_template('edit.html', form=form)

@app.route('/delete/<int:post_id>')
@login_required
def delete(post_id):
    post = Post.query.get_or_404(post_id)
    if post.user_id != current_user.id:
        flash('Not authorized to delete this post.')
        return redirect(url_for('index'))
    db.session.delete(post)
    db.session.commit()
    return redirect(url_for('index'))

if __name__ == '__main__':
    db.create_all()
    app.run(debug=True)
```