Internship Project Report: Flask Contact Form Web Application

1. Introduction

This report documents the development of a personal portfolio web application integrated with a contact form using Flask and SQLite. The primary goal of this project was to allow visitors to interact with the developer through a simple, elegant contact submission interface. The application emphasizes clean UI design, intuitive navigation, and a working backend that stores user-submitted messages in a local SQLite database.

2. Project Inspiration and Background

The inspiration for this project arose from the need to provide a functional portfolio website that not only showcases personal and academic achievements but also provides a communication bridge between the developer and potential collaborators, recruiters, or peers. Rather than using external tools like Google Forms, the aim was to handle form submission, validation, and data storage natively using Flask and SQLite.

3. Project Objective and Features

- Portfolio presentation for showcasing skills and projects.
- Contact form allowing users to submit categorized messages.
- Backend developed with Flask for routing and form handling.
- SQLite database to store and manage form data.
- Flash messages to provide user feedback.

4. Project Objective and Features

The frontend of the site is developed using HTML, CSS, and some embedded Jinja2 template logic. It contains a styled contact form with input fields for name, email, category, and message.

```
<form method="POST" action="/" class="form">
    <input type="text" name="name" required>
    <input type="email" name="email" required>
    <select name="category" required>
        <option value="Feedback">Feedback</option>
        <option value="Suggestion">Suggestion</option>
        <option value="Complaint">Complaint</option>
        <option value="Other">Other</option>
    </select>
    <textarea name="message" rows="5" required></textarea>
    <button type="submit">Send Message</button>
</form>
```

5. Backend: app.py

The Flask backend handles GET and POST requests. When the form is submitted (POST request), data is retrieved from the form fields and passed to the database layer.

```
from flask import Flask, request, render_template, redirect, flash
import sqlite3

app = Flask(__name__)
app.secret_key = 'secret_key'  # Replace with a secure key
```

A class `ContactDB` is created to manage database connections and queries. It ensures the message table is created if not present.

```
class ContactDB:
    def __init__(self, db_name='contact.db'):
        self.db_name = db_name
        self.create_table()

    def create_table(self):
        with sqlite3.connect(self.db_name) as conn:
            conn.execute('''
                CREATE TABLE IF NOT EXISTS messages (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    name TEXT NOT NULL,
                    email TEXT NOT NULL,
                    category TEXT NOT NULL,
                    message TEXT NOT NULL
                )
            ''')

    def save_message(self, name, email, category, message):
        with sqlite3.connect(self.db_name) as conn:
            conn.execute('''
                INSERT INTO messages (name, email, category, message)
                VALUES (?, ?, ?, ?)
            ''', (name, email, category, message))
```

The route `/` handles both GET and POST. It renders the homepage or stores the form input depending on the request type.

```
@app.route('/', methods=['GET', 'POST'])
def home():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        category = request.form['category']
        message = request.form['message']

        db.save_message(name, email, category, message)
        flash('Your message has been sent!')

    return render_template('index.html', title='Home')
```

6. Flash Messaging and Jinja2 Integration

To notify users after form submission, Flask's `flash()` is used. The message is displayed using Jinja2 templating in the HTML file.

```
{% with messages = get_flashed_messages() %}
  {% if messages %}
    <div class="flash-message">
    {% for message in messages %}
        <p>{{ message }}</p>
    {% endfor %}
    </div>
  {% endif %}
{% endwith %}
```

7. Design and Frontend Structure

The frontend is built with HTML5 and CSS3, using modern UI principles such as minimal design, responsiveness, and accessibility. The home page includes sections such as About, Projects, and Contact, styled using embedded stylesheets and enhanced with Google Fonts and Font Awesome icons.

8. Backend Architecture and Flask Routing

The application backend is built using the Flask web framework with routing logic defined in app.py. Data from the contact form is processed and stored using the SQLite database via Python's sqlite3 module.

9. Database Handling Using SQLite

SQLite is used as the backend database to store contact form messages. A ContactDB class was created to encapsulate all database operations.

10. Flash Messaging and User Feedback

To confirm successful message submission, Flask's flash() function is used. The feedback appears dynamically at the top of the contact form.

11. Contact Form Features

The form includes Email, Name, Category, and Message fields. Options include Feedback, Suggestion, Complaint, and Other. All fields are validated, and data is saved only on successful submission.

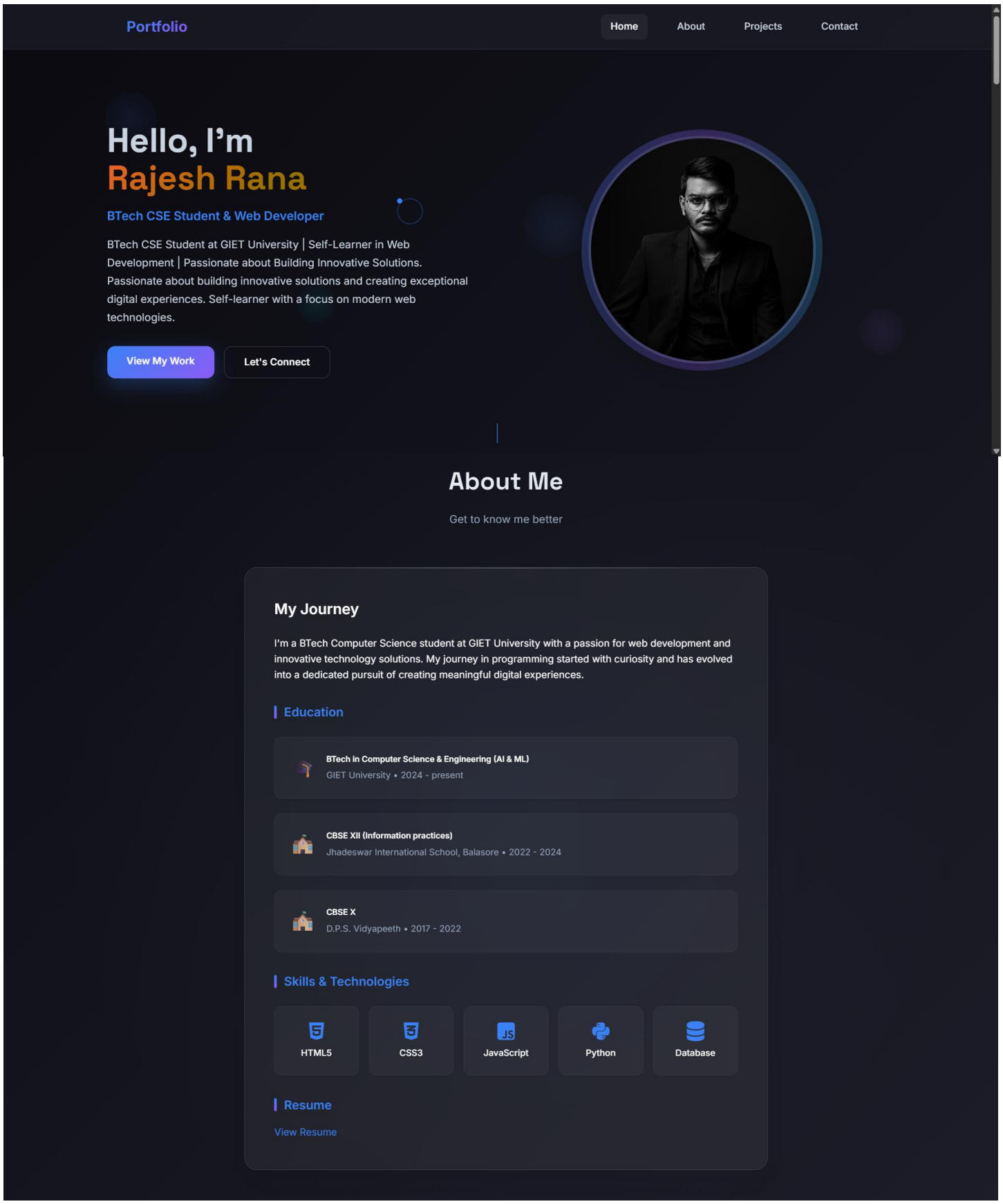
12. Technologies Used

- Frontend: HTML5, CSS3, JavaScript, Jinja2
- Backend: Python 3.x, Flask
- Database: SQLite
- Flash Messaging: Flask
- Hosting (Optional): PythonAnywhere

13. Project Structure

```
project/
    ├── app.py              # Flask backend logic
    ├── contact.db          # SQLite database
    ├── templates/
    │   └── index.html      # HTML frontend with form
    ├── static/
    │   └── style.css       # Optional CSS styles
```

14. Screenshots



Let's Connect

Ready to start your next project?

Get In Touch

I'm always open to discussing new opportunities and interesting projects. Let's create something amazing together!

Email
rr0656966@gmail.com

Phone
+91 79782 62400

Location
Bhadrak, Odisha, India

Your Name: Rajesh Rana

Your Email: rr0656966@gmail.com

Complaint

Your Message: Hii!

Send Message

Let's Connect

Ready to start your next project?

Get In Touch

I'm always open to discussing new opportunities and interesting projects. Let's create something amazing together!

Email
rr0656966@gmail.com

Phone
+91 79782 62400

Location
Bhadrak, Odisha, India

Your message has been sent!

Your Name

Your Email

Select a Category

Your Message

Send Message

15. Future Scope

- Add email notifications upon submission (via Flask-Mail).
- Integrate CAPTCHA to prevent spam.
- Store messages in a full-featured SQL database with admin panel.
- Allow authenticated users to view submission history.

16. Conclusion

This Flask-based contact form project showcases practical skills in Python, Flask, web templating, and data persistence using SQLite. With a clear separation between frontend and backend, the app is scalable, functional, and easy to expand into more complex applications.