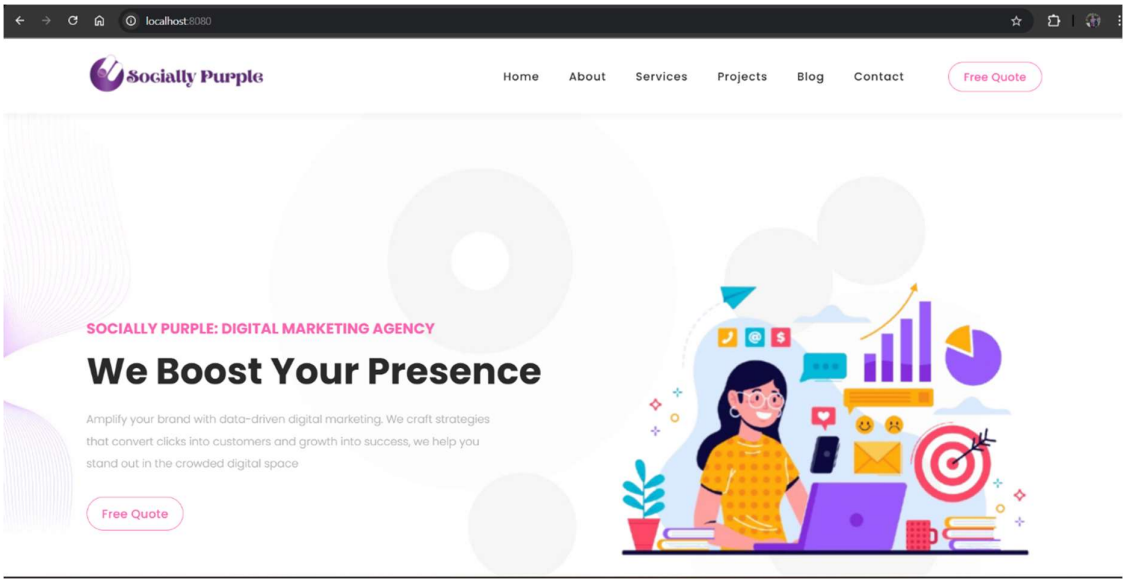
Documentation: Socially Purple Digital Marketing Website



Project Overview

Project Name: Socially Purple – Digital Marketing Website
Technology Stack: HTML, CSS, JavaScript, Thymeleaf (View Engine), Spring Boot (for optional backend integration)
Deployment Environment: Spring Boot (with embedded Tomcat), optional integration with email service and database

Socially Purple is a digital marketing agency's website built using a responsive HTML/CSS template. The project showcases dynamic content rendering with Spring Boot and Thymeleaf, while ultimately shifting to a frontend-only solution using Google Forms in place of server-side contact processing. This project was undertaken as a professional-grade, major project, for a marketing enterprise with detailed architecture planning, scalable design, and real-time UI/UX considerations, and this documentation is made with all due consideration to invite developers to contribute and suggest better aspect and functionalities as well as collaborate with the organization to upgrade the website in near future. Socially Purple and the associated developers allow professionals from around the world to use this project for their commercial purposes, however, Socially Purple strictly upholds the copyright policies and re-posting/ re-distributing this project/template without consent would be treated under legal matters.

Developers: Piyush Pal & Shivani Wakde

Key Features

- Responsive layout with professional design.
- Embedded Google Map for location visibility.
- Embedded Google Form for client inquiries.
- Contact icons for phone, email, and location.
- Smooth animations and scrolling effects.

Removed Backend Functionalities

Initially included:

- `ContactController` with POST `/contact`
- `EmailService.java` with Gmail SMTP integration
- Form binding using `th:object="${contact}"`

Currently removed due to Google Form replacement for simplicity.

Table of Contents

Project Structure

```
SPdigitalmarketing/
├── .idea/                          → IntelliJ project settings (auto-
generated)
├── out/                            → Compiled .class files (auto-
generated)
├── .mvn/                           → Maven wrapper files (optional)
├── pom.xml                         → Project Object Model file (manages
dependencies)
├── src/
│   └── main/
│       ├── java/
│       │   └── com/
│       │       └── sociallyPurple/
│       │           ├── SPdigitalmarketing/
│       │           │   └── DigitalMarketingApplication.java
│       │           └── model/
│       │               └── Contact.java
│       │
│       ├── resources/
│       │   ├── static/
│       │   │   ├── css/
│       │   │   │   ├── animated.css
│       │   │   │   ├── fontawesome.css
│       │   │   │   ├── owl.css
│       │   │   │   └── templatemo-digimedia-v1.css
│       │   │   ├── fonts/              → Web fonts used in template
│       │   │   ├── images/             → Static images/icons
│       │   │   ├── js/                 → JavaScript files
│       │   │   └── vendor/             → Third-party dependencies
│       │   │       ├── bootstrap/
│       │   │       │   ├── css/
│       │   │       │   └── js/
│       │   │       └── jquery/
│       │   ├── templates/
│       │   │   └── index.html          → Thymeleaf HTML template (main web)
│       │   └── application.properties → Spring Boot configurations
```

Frontend Architecture

- **Template Used:** DigiMedia by TemplateMo.
- **Language:** HTML5, CSS3, JS
- **Styling:** Bootstrap + custom CSS
- **Icons and Animations:** Font Awesome, WOW.js
- **Responsive Design:** Flexbox/Grid + Media Queries
- **Dynamic Elements:** Google Map iframe, form input replacements
- **Third-party Integration:** Google Forms

Backend Overview (Deprecated)

The backend was originally powered by Spring Boot. It managed contact form submissions by capturing form input using a POJO class (`Contact.java`), sending the information via SMTP mail using Spring's `JavaMailSender`, and redirecting the user back to the homepage with success flash messages.

Components included:

- `ContactController.java` with GET and POST endpoints
- `EmailService.java` for mail composition and dispatch
- Thymeleaf binding using `th:object`, `@ModelAttribute`, etc.

Reason for deprecation: Despite correct wiring, Thymeleaf template errors and frontend disruption caused by binding issues led to the strategic shift toward Google Forms for seamless UX.

Google Form Integration

The backend form was removed and replaced by a Google Form. A styled button (`<button>`) was placed in the exact structural spot to avoid UI distortion. Google Form allows users to submit inquiries, with collected responses accessible in a connected Google Sheet or emailed directly to the form owner.

Button Example

```
<a href="https://forms.gle/YOUR_FORM_LINK" target="_blank" class="main-
button">
  <button class="btn btn-primary" style="padding: 16px 36px; font-size:
18px; background-color: #ff589e; border: none; border-radius: 30px; font-
weight: 600;">
    Fill Out Our Google Form
  </button>
</a>
```

Placeholder for Removed Form

```
<textarea class="form-control" disabled style="pointer-events: none;">
  Please fill out the Google Form above & we'll reach out to you at the
earliest.
</textarea>
```

UI Preservation & Workarounds

- All original grid structures (`col-lg-*`) were maintained.
- `<form>` tags were removed, but the layout retained with equivalent `<div>` wrappers.

- Empty form areas were replaced with disabled fields or messages to preserve padding/margins.
- Google Map iframe retained and resized to match original design fluidity.

Optional Backend Integration: Mail & DB

For users wanting to reintroduce backend processing with Spring Boot, here's a complete guide.

1. Contact Model (POJO)

```
public class Contact {
    private String name;
    private String email;
    private String subject;
    private String message;
    // Getters and Setters
}
```

2. Email Service

```
@Service
public class EmailService {
    @Autowired private JavaMailSender mailSender;

    public void sendContactFormEmail(Contact contact) {
        SimpleMailMessage message = new SimpleMailMessage();
        message.setTo("your@email.com");
        message.setSubject("New Contact Submission");
        message.setText("Name: " + contact.getName() + "\n"
                + "Email: " + contact.getEmail() + "\n"
                + "Subject: " + contact.getSubject() + "\n"
                + "Message: " + contact.getMessage());
        mailSender.send(message);
    }
}
```

3. Controller

```
@Controller
public class ContactController {
    @Autowired private EmailService emailService;

    @GetMapping("/")
    public String home(Model model) {
        model.addAttribute("contact", new Contact());
        return "index";
    }

    @PostMapping("/contact")
    public String submit(@ModelAttribute Contact contact,
RedirectAttributes redirectAttributes) {
        emailService.sendContactFormEmail(contact);
```

```
        redirectAttributes.addFlashAttribute("successMessage", "Message
sent successfully!");
        return "redirect:/";
    }
}
```

4. application.properties

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=your@gmail.com
spring.mail.password=your-app-password
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

5. Optional DB Integration

To store contacts:

- Add Spring Data JPA
- Create `ContactRepository` extending `JpaRepository`
- Annotate `Contact` with `@Entity`, `@Id`, and persist in the controller

Build and Run Instructions

Prerequisites:

- JDK 17
- Maven 3.8+

Build Project:

```
mvn clean install or use mvn clean install -DskipTests (if facing error)
```

Run Spring Boot App:

```
mvn spring-boot:run
```

Visit: http://localhost:8080/ (strictly use this context if integrating backend)

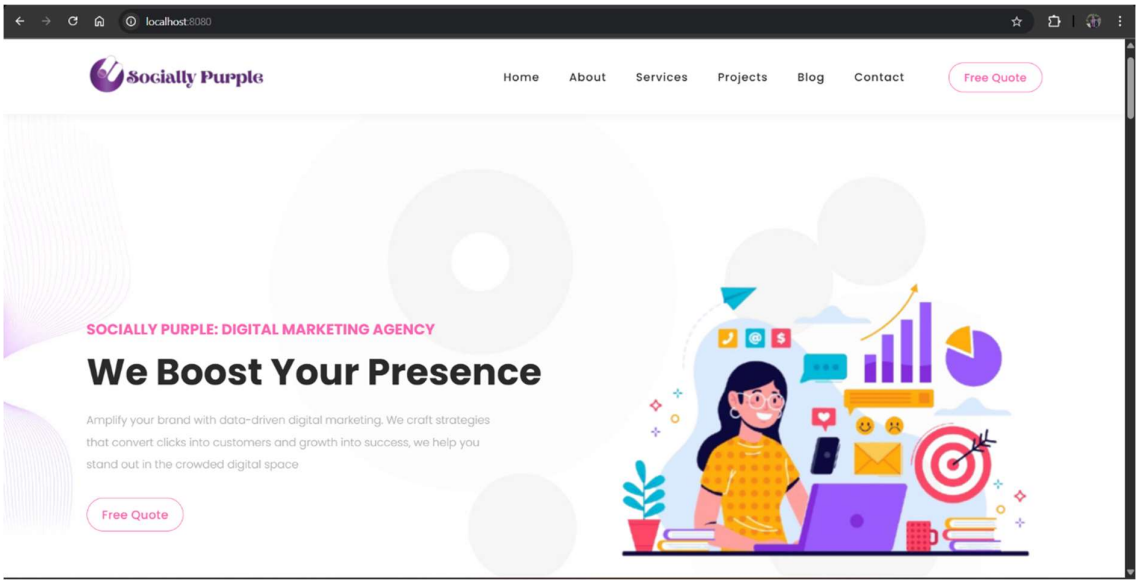
How to Deploy

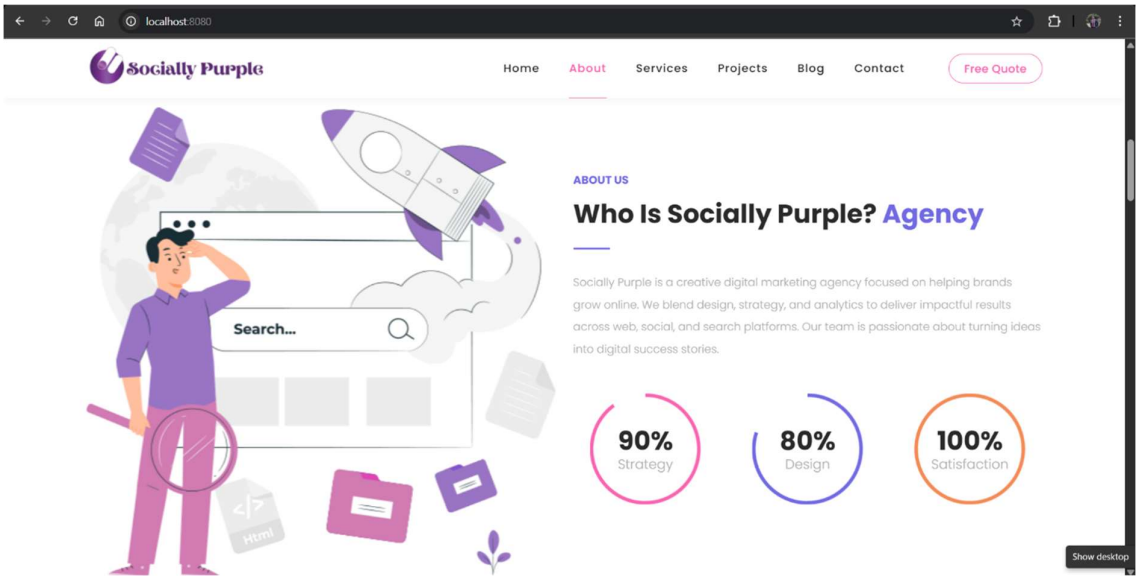
- For **static hosting** (Netlify, GitHub Pages, Vercel):
  - Remove backend entirely.
  - Use plain `.html` files.
- For **Spring Boot hosting**:
  - Keep the controller, but disable `/contact` POST mapping.
  - Serve templates from src/main/resources/templates/.

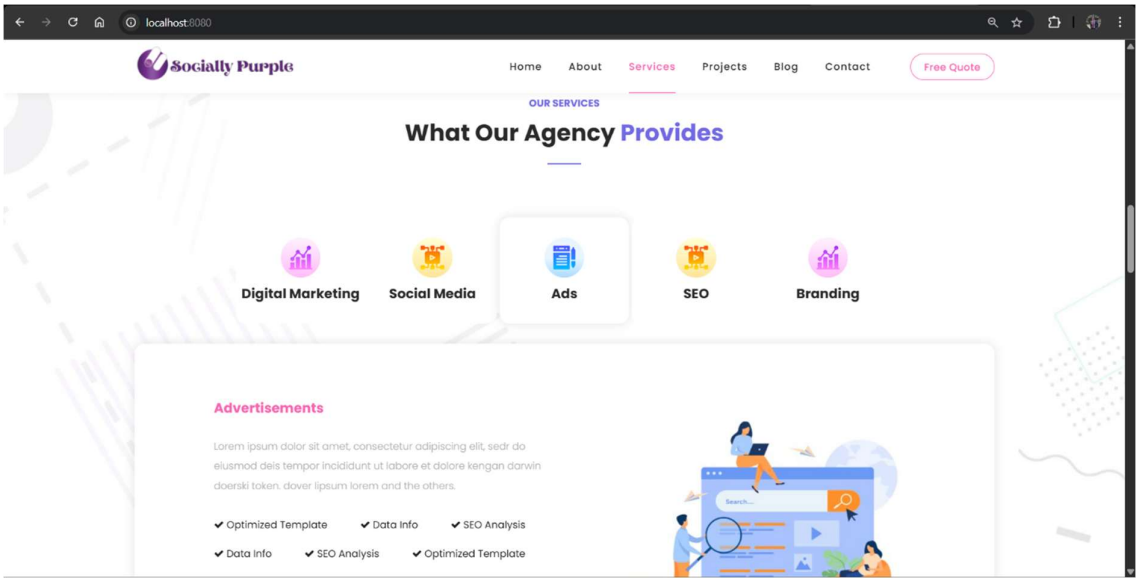
Suggestions for Future Additions

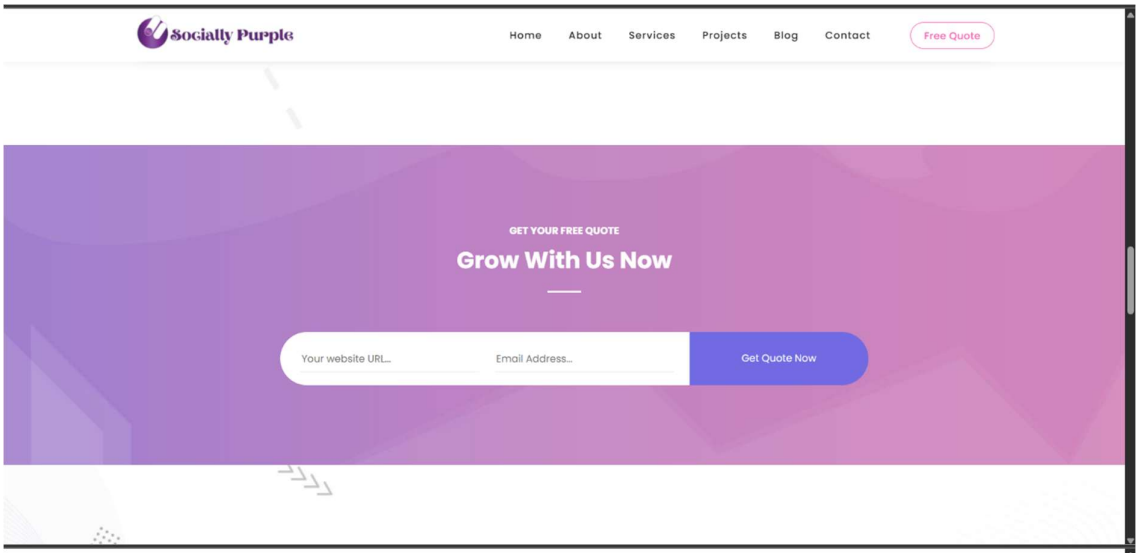
- Re-enable backend contact form with validations
- Add a blog/news section (Spring Data JPA)
- Analytics and SEO tools integration
- Testimonials carousel
- Admin dashboard to manage leads (if using a DB)

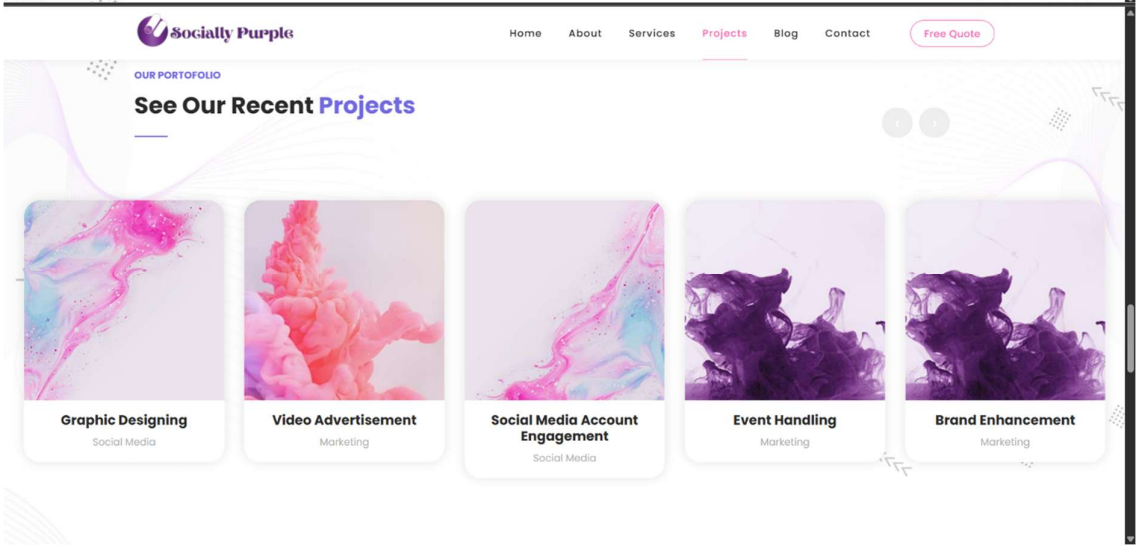
Screenshots:

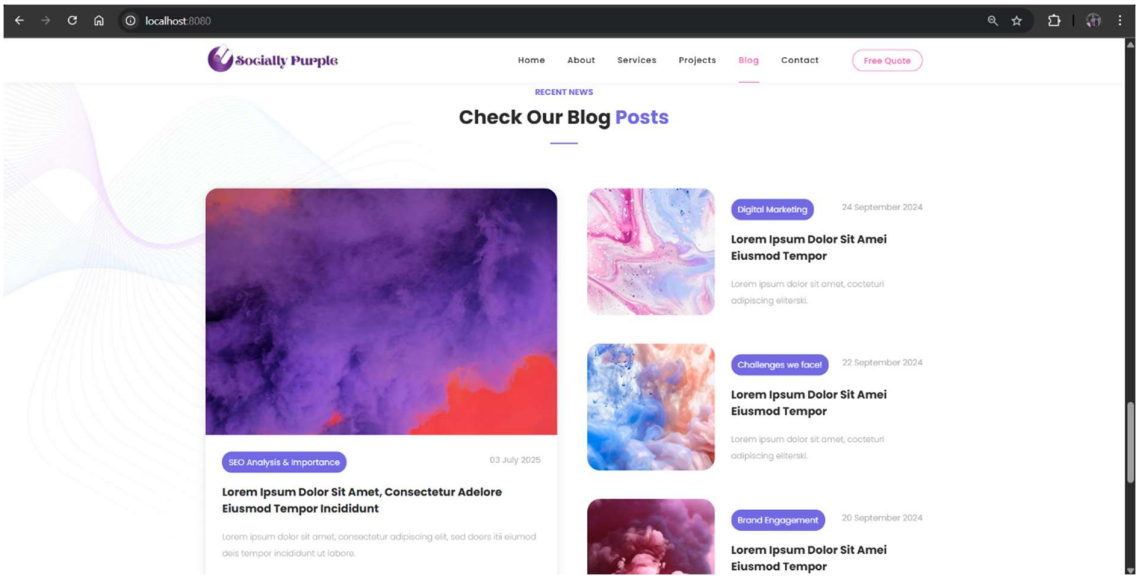


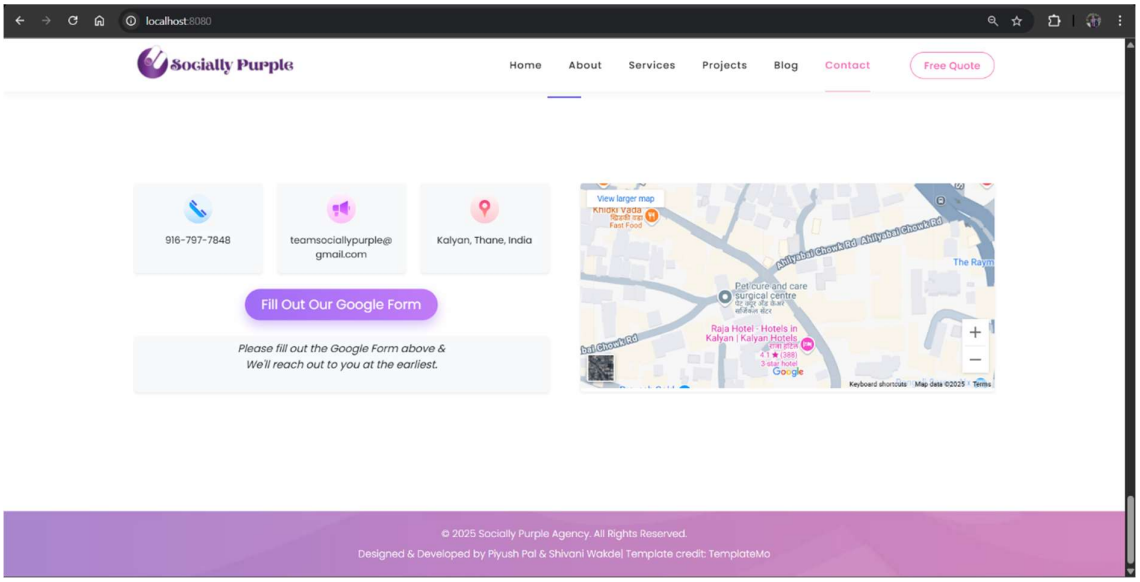



localhost:8080
Socially Purple
Home
About
Services
Projects
Blog
Contact
Free Quote
OUR SERVICES
What Our Agency Provides
Digital Marketing
Social Media
Ads
SEO
Branding
Advertisements
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sedr do eiusmod deis tempor incididunt ut labore et dolore kengan darwin doerski token. dover lipsum lorem and the others.
✔ Optimized Template
✔ Data Info
✔ SEO Analysis
✔ Data Info
✔ SEO Analysis
✔ Optimized Template


Socially Purple
Home
About
Services
Projects
Blog
Contact
Free Quote
GET YOUR FREE QUOTE
Grow With Us Now
Your website URL...
Email Address...
Get Quote Now


Socially Purple
Home
About
Services
Projects
Blog
Contact
Free Quote
OUR PORTOFOLIO
See Our Recent Projects
Graphic Designing
Social Media
Video Advertisement
Marketing
Social Media Account Engagement
Social Media
Event Handling
Marketing
Brand Enhancement
Marketing





Conclusion

The Socially Purple digital marketing website is an ideal blend of design precision, flexible integration options, and clean UI/UX workflow. By allowing optional backend support, it's suitable for both static informational pages and full-stack enterprise deployments.

This documentation provides complete guidance for developers, university reviewers, and clients who may wish to scale or reuse the system.