

INTEGRATION SUBSCRIPTION BILLING WITH STRIPE

A NAAN MUDHALVAN REPORT

SUBMITTED BY

SANMATHI S 912421106305

SAROJADEVI P 912421106306

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING



SHANMUGANATHAN ENGINEERING COLLEGE

ARASAMPATTI, PUDUKKOTTAI 622 507

YEAR & SEMESTER : IV & VII

SUBJECT CODE :NM1050

COURSE NAME :SaaS (SOFTWARE AS A SERVICE)



ANNA UNIVERSITY::CHENNAI 600 025

NOV/DEC 2024

BONAFIDE CERTIFICATE

Certified that this Naan Mudhalvan report of **“INTEGRATION SUBSCRIPTION BILLING WITH STRIPE”** is the bonafide work of **“SANMATHI S (912421106305)” & “SAROJADEVI P(912421106306)”** who carried out the project work under my guidance.

SIGNATURE

Mrs. D. LATHA M.E.,
NAAN MUDHALVAN COORDINATOR
ASSISTANT PROFESSOR,
Department of Electronics &
Communication Engineering,
Shanmuganathan Engineering College,
Arasampatti – 622 507

SIGNATURE

Dr. A.MUTHU MANICKAM M.E., Ph.D.,
HEAD OF THE DEPARTMENT
ASSISTANT PROFESSOR,
Department of Electronics &
Communication Engineering,
Shanmuganathan Engineering College,
Arasampatti – 622 507

Submitted for the Internship viva-voice on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGMENT

At this pleasing moment having successfully completed our internship report, we wish to convey our sincere thanks to our beloved chairperson **Mrs. PICHAPPA VALLIAMMAL**, correspondent **Dr. P. MANIKANDAN B.E**, director(Academic) Shri **M.SHANMUGANATHAN**, director(Administration) Shri **M. PICHAPPA** and honourable secretary **Mr. M. VISWANATHAN** for their extensive support.

I thankful to our principal **Dr. KL. MUTHURAMU M.E(W.R)., M.E(S.E)., Ph.D., FIE., M.I.S.T.E.**, Shanmuganathan engineering college, for providing the opportunity to conduct our project.

I extend our gratitude to **Dr. A.MUTHU MANICKAM M.E., Ph.D.**, the head of the department and our internship co-ordinator of Electronics and communication engineering for providing a valuable suggestion and supports given through the study.

Gratitude never fails. We are grateful to our dynamic and effective internal guide **Asst. Prof. Mrs. D. LATHA, AP/ECE, M.E.**, for his valuable innovative suggestion, constructive interactions, constant encouragement and valuable helps that have been provided us throughout the project.

I also express my heartfelt thanks to all other staff members of Electronics communication engineering Department for their support. Above all, we thank our parents, for affording us the valuable education till now.

ABSTRACT

When abstracting Stripe's subscription billing system, focus on creating modular components that separate concerns, allowing the rest of your application to interact with high-level billing concepts without dealing with Stripe's specific API calls. Key abstraction points include **Customer Management** for handling customer creation and updates, **Payment Method Handling** for managing customers' payment sources, **Subscription Lifecycle Management** to cover all stages of subscriptions (creation, updating, cancellation), and **Pricing and Plans** for dealing with products and pricing configurations. Additionally, **Invoice Management** enables efficient handling of billing cycles and payments. By defining a clear API at each of these abstraction points, you encapsulate Stripe's implementation details, making it easier to maintain, test, and adapt to future billing needs or alternative payment providers. This approach results in a billing layer that interacts seamlessly with your application, ensuring flexibility and modularity.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
1.	INTRODUCTION OF INTEGRATION SUBSCRIPTION BILLING WITH STRIPE	
	1.1 INTRODUCTION TO INTEGRATION	
	SUBSCRIPTIONBILLING WITH STRIPE	6
	1.2 USES OF TO INTEGRATION SUBSCRIPTION	
	BILLINGWITH STRIPE	6
2.	TOOLS TO INTEGRATION SUBSCRIPTION BILLING WITH STRIPE	
	2.1 TOOLS TO INTEGRATION SUBSCRIPTION	
	BILLINGWITH STRIPE	7
	2.2 FRONT-END PAYMENT COLLECTION	8
	2.3 BACK-END FRAME WORK LIBRARIES	8
	2.4 WEBHOOK MANAGEMENT	9
	2.5 BILLING AND SUBSCRIPTION MANAGEMENT	9
	2.6 ANALYTICS AND REPORTING	9
3.	PROJECT I MPLEMENTATION	
	3.1 HTML CODES FOR PROJECT	10
	IMPLEMENTATION	10
	3.2CSS CODES FOR PROJECT	
	IMPLEMENTATION	
4.	CONCLUSION	
	4.1 CONCLUSION	12

CHAPTER 1

INTRODUCTION OF INTEGRATE SUBSCRIPTION BILLING WITH STRIPE

1.1 INTRODUCTION TO INTEGRATION SUBSCRIPTION BILLING WITH STRIPE

Integrating subscription billing with Stripe enables you to manage recurring payments, customersubscriptions, and billing cycles in a streamlined way, with Stripe handling the backend complexity of payment processing. Stripe provides robust APIs for handling subscriptions, making it a popular choice for SaaS platforms, membership sites, and any service that relies on recurring revenue. The integration process typically involves creating and managing **customers** and **subscriptions** and setting up **payment methods**. You also have to handle **invoices**, **payments**, and **webhooks** to keep your application informed about events like successful payments, subscription renewals, or failures. An effective integration focuses on abstracting these features into modular components, allowing your application to interact with high-level billing functions without getting into the intricacies of Stripe's API. A well-abstracted Stripe integration allows you to handle everything from customer creation to complex billing requirements, such as proration, free trials, and tiered pricing, with minimal coding in your core application. By creating a dedicated billing module, you separate billing logic from other business logic, making your application more modular, maintainable, and scalable. This setup is particularly beneficial if you plan to extend your billing to support other payment providers or billingmodels in the future.

1.2 USES OF INTEGRATION SUBSCRIPTION BILLING WITH STRIPE:

Automated Recurring Billing: Stripe's subscription billing system allows businesses to automaticallycharge customers on a recurring basis, reducing the need for manual invoicing and payment collection. This is especially useful for businesses offering services on a monthly, quarterly, or annual basis, such as SaaS applications, content subscriptions, and memberships.

Flexible Pricing and Plans: With Stripe, you can set up various pricing plans tailored to different customer needs. Stripe supports multiple billing models, including flat-rate, per-seat, usage-based, andtiered pricing, enabling you to customize

offerings for individual customers or segments. This flexibility makes it easy to implement different subscription tiers or add-on services.

Customer and Payment Method Management: Stripe allows businesses to securely manage customer information, including payment methods, which helps streamline the billing process. Stripe supports a variety of payment methods such as credit cards, debit cards, and ACH, as well as international options like digital wallets, allowing you to cater to a global audience.

Handling Invoicing and Proration: Stripe provides tools for generating and managing invoices, which is essential for businesses that require detailed billing records. Additionally, Stripe can handle proration for mid-cycle plan changes, ensuring customers are billed accurately for upgrades or downgrades, and creating a seamless billing experience.

Comprehensive Webhook Support for Real-time Notifications: Stripe's webhook support enables your system to listen for important events, such as payment successes, subscription renewals, and payment failures. This allows you to respond in real-time, updating customer statuses, sending notifications, and retrying failed payments to reduce churn.

Revenue Insights and Analytics: Stripe's dashboard offers built-in reporting and analytics features, allowing businesses to gain insights into key metrics like MRR (Monthly Recurring Revenue), LTV (Customer Lifetime Value), and churn rate. This helps businesses monitor growth, make data-driven decisions, and optimize pricing or customer retention strategies.

Reduced Development Effort and Maintenance: Stripe handles the security, compliance, and complex logic behind subscription billing, which would otherwise require significant development resources to build and maintain. This enables businesses to focus on core offerings, reducing time-to-market and operational costs.

CHAPTER 2

TOOLS TO INTEGRATE SUBSCRIPTION BILLING WITH STRIPE

2.1 TOOLS TO DEVELOP INTEGRATE SUBSCRIPTION BILLING WITH STRIPE

To develop a subscription billing integration with Stripe, several tools and resources can facilitate development, improve security, and ensure a smooth, maintainable integration process. Here's a list of tools and resources essential for building a subscription billing system with Stripe:

1. Stripe Developer Tools

Stripe API: The core API that provides access to all of Stripe's subscription billing features, such as customer creation, plan management, payment processing, and invoices.

Stripe SDKs: Available in multiple programming languages (Python, JavaScript, PHP, Ruby, Node.js, Java, etc.), these SDKs simplify API calls and help handle complex logic like payment intents and authentication.

Stripe CLI: A command-line tool that simplifies development with Stripe by allowing you to create, manage, and delete test data, as well as to test Webhooks in your local environment.

2. Frontend Payment Collection

Stripe Elements: A set of customizable pre-built UI components to securely collect customer payment details, supporting responsive design and simplified PCI compliance. Stripe Elements works directly with the Stripe SDKs.

Stripe Checkout: A hosted, pre-built checkout page by Stripe that handles payment collection and compliance, offering a faster implementation for those who want to avoid building their own UI.

3. Backend Frameworks and Libraries

Web Frameworks: Popular backend frameworks like **Django** (Python), **Express** (Node.js), **Laravel**

Task Scheduling Tools: Tools like **Celery** (Python) or **Sidekiq** (Ruby) help schedule tasks, such as retrying failed payments or sending customer notifications for failed charges.

4. Testing and Simulation

Stripe's Test Mode: Use Stripe's test mode for running and verifying subscription flows without real charges. Stripe offers numerous test card numbers to simulate various scenarios like successful charges, authentication failures, and more.

Stripe's Webhook Testing: Simulate webhook events using the Stripe CLI, which allows you to test how your system reacts to events like payment failures, subscription renewals, or customer cancellations.

5. Webhook Management

Webhook Management Tools: Use services like **ngrok** for testing webhooks locally, forwarding Stripe's webhook requests to your local environment.

Retry and Logging Libraries: Libraries like **Sentry** for error tracking and **BullMQ** (for Node.js) or Celery (for Python) can help set up retry mechanisms for handling webhook failures, ensuring resilience.

6. Security and Compliance

Environment Management

Tools like **dotenv** for managing sensitive data (e.g., Stripe secret keys) securely in environment files.

Secure Payment Collection: Stripe Elements and Stripe Checkout automatically handle PCI compliance for you, but if you're collecting and transmitting card data, ensure compliance by using Stripe's prebuilt components and following security guidelines.

7. Billing and Subscription Management Libraries

Stripe's Billing Portal: A pre-built Stripe portal where customers can manage their subscriptions (change plans, cancel, update payment methods), reducing the amount of custom code you need to write.

Subscription Management Libraries: If you're using a popular framework, check for available subscription management libraries that may streamline Stripe integration, like **dj-stripe** for Django or **Cashier** for Laravel.

8. Analytics and Reporting

Stripe Dashboard: Stripe's built-in dashboard offers insights into billing, MRR, churn rate, and other financial metrics, providing instant analytics without extra development.

Additional Analytics Platforms: Tools like **Baremetrics** or **ProfitWell** integrate directly with Stripe to offer more in-depth subscription analytics and insights.

9. Documentation and Resources

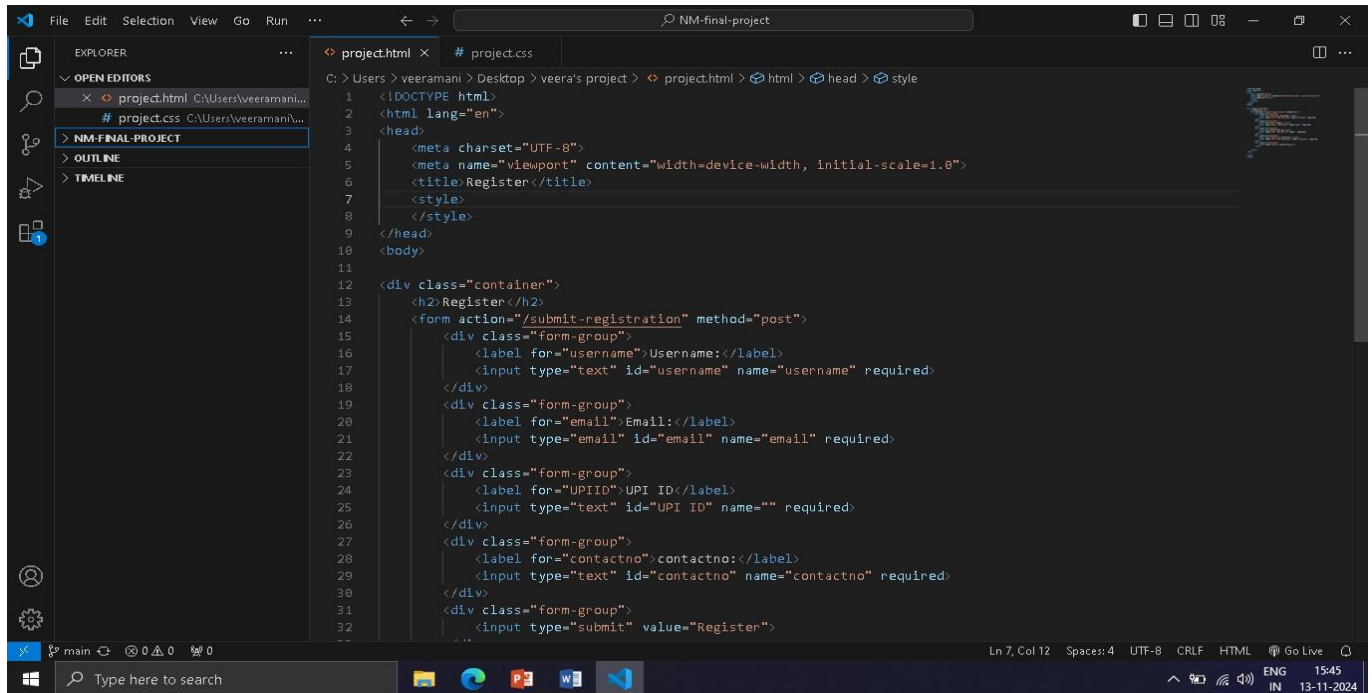
Stripe Documentation: Stripe offers extensive documentation with guides, examples, and detailed API references.

Stripe Samples: Sample code repositories from Stripe, like **sample-code** on GitHub, provide pre-built examples for common subscription use cases.

CHAPTER 3

PROJECT IMPLEMENTATION

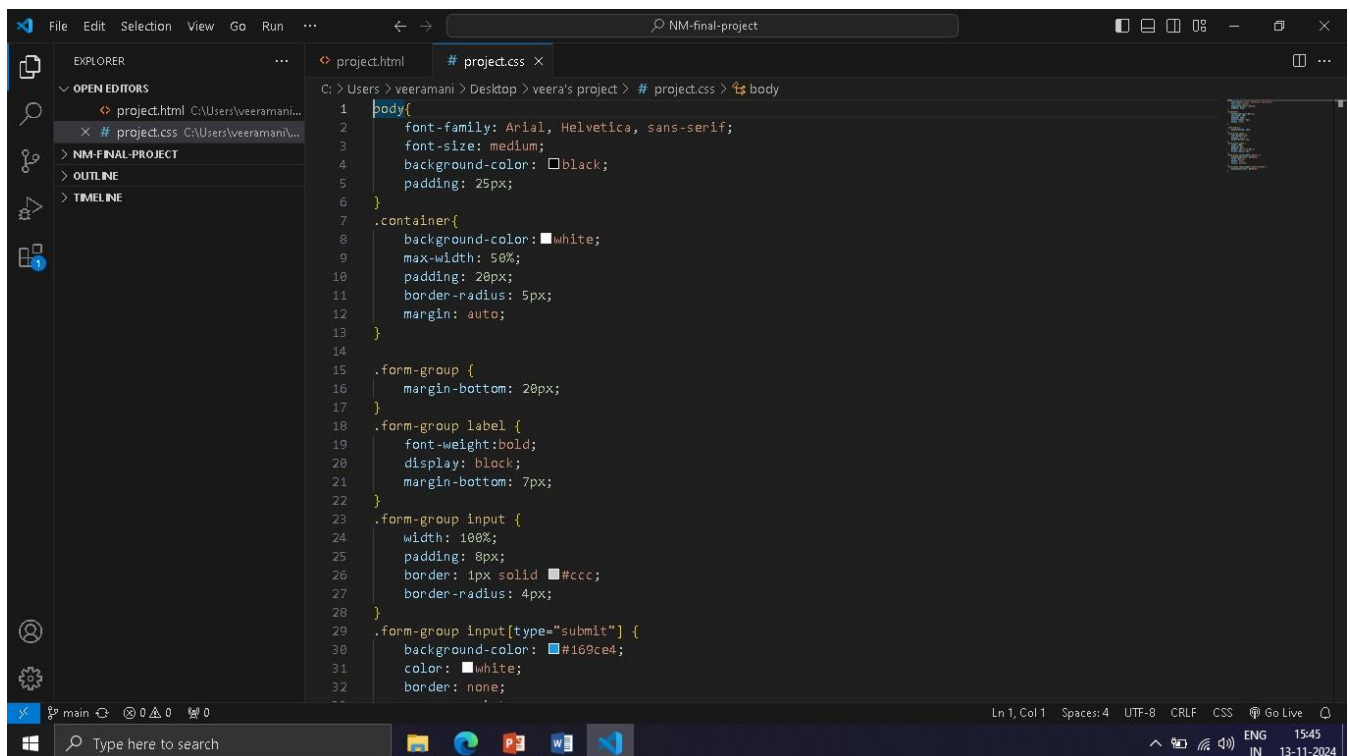
3.1 HTML CODES FOR PROJECT IMPLEMENTATION:



The screenshot shows the Visual Studio Code editor with a project named 'NM-final-project'. The Explorer panel on the left shows the file structure with 'project.html' and 'project.css'. The main editor area displays the HTML code for 'project.html'. The code includes a DOCTYPE declaration, a meta charset of 'UTF-8', a viewport meta tag, a title 'Register', and a form with five input fields: Username, Email, UPI ID, Contact Number, and a Submit button. The form is styled with a container class and form-group classes.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Register</title>
7   <style>
8   </style>
9 </head>
10 <body>
11
12 <div class="container">
13   <h2>Register</h2>
14   <form action="/submit-registration" method="post">
15     <div class="form-group">
16       <label for="username">Username:</label>
17       <input type="text" id="username" name="username" required>
18     </div>
19     <div class="form-group">
20       <label for="email">Email:</label>
21       <input type="email" id="email" name="email" required>
22     </div>
23     <div class="form-group">
24       <label for="UPIID">UPI ID</label>
25       <input type="text" id="UPI ID" name="" required>
26     </div>
27     <div class="form-group">
28       <label for="contactno">contactno:</label>
29       <input type="text" id="contactno" name="contactno" required>
30     </div>
31     <div class="form-group">
32       <input type="submit" value="Register">
33     </div>
34   </form>
35 </div>
```

3.2 CSS CODES FOR PROJECT IMPLEMENTATION:



The screenshot shows the Visual Studio Code editor with the same project. The Explorer panel shows 'project.html' and 'project.css'. The main editor area displays the CSS code for 'project.css'. The code defines styles for the body, container, form-group, form-group label, form-group input, and form-group input[type="submit"].

```
1 body{
2   font-family: Arial, Helvetica, sans-serif;
3   font-size: medium;
4   background-color: black;
5   padding: 25px;
6 }
7 .container{
8   background-color: white;
9   max-width: 50%;
10  padding: 20px;
11  border-radius: 5px;
12  margin: auto;
13 }
14
15 .form-group {
16   margin-bottom: 20px;
17 }
18 .form-group label {
19   font-weight: bold;
20   display: block;
21   margin-bottom: 7px;
22 }
23 .form-group input {
24   width: 100%;
25   padding: 8px;
26   border: 1px solid #ccc;
27   border-radius: 4px;
28 }
29 .form-group input[type="submit"] {
30   background-color: #169ce4;
31   color: white;
32   border: none;
33 }
```


OUTPUT:



POWDUR

Pure set

\$65.00



Shipping information

Email

Shipping address

Home

United States

Address

Payment method

Card

Cash App

Apple Pay

Card information

1234 1234 1234 1234

MM / YY

CVC

Pay

Powered by stripe

Legal

Refunds

11

The Output that we get in the project implementation is the figures ,codes and the HTML ,CSS and relevant Front-end and the Back-end that should be obtain from Visual Studio code.

CHAPTER 4

CONCLUSION

CONCLUSION:

In conclusion, integrating subscription billing with Stripe offers a powerful solution for businesses that rely on recurring revenue. Stripe's comprehensive API, along with its developer-friendly SDKs, CLI, and tools like Stripe Elements and Stripe Checkout, simplifies the process of creating a robust and secure billing system. By leveraging abstraction points—such as customer management, payment methods, subscription lifecycle, and invoicing—you can design a modular, maintainable billing layer that easily adapts to evolving business needs. Using additional tools like webhooks for real-time updates, task schedulers for managing retries, and analytics platforms for insights into billing metrics, you gain greater control over the billing experience and better visibility into key financial data. Stripe's ecosystem supports rapid development, minimizes the risk of handling sensitive financial data, and automates compliance, allowing you to focus more on core business activities rather than billing logistics. Ultimately, a well-implemented Stripe integration can improve customer experience, streamline billing operations, and drive sustainable growth by ensuring reliable, automated revenue generation.