Development of a feature-rich, Book Store

A Project Report for Industrial Training & Internship

Submitted by

SUBHOJIT CHAKRABORTY

DEBANSU GHOSH

In partial fulfilment for the award of the degree of

BCA

TMSL

At

Euphoria GenX

Euphoria GenX

BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

“Development of a feature-rich, practical Book Store “is the Bonafide work of

Name of the student: SUBHOJIT CHAKRABORTY **Signature:**

Name of the student: DEBANSU GHOSH **Signature:**

SIGNATURE

Name : Saumitra Das

PROJECT MENTOR

Acknowledgement

I take this opportunity to express my deep gratitude and sincerest thank to my project mentor, Mr. Saumitra Das for giving most valuable suggestion, helpful guidance and encouragement in the execution of this project work.

I will like to give a special mention to my colleagues. Last but not the least I am grateful to all the faculty members of Euphoria GenX or their support.

Table of Contents	Page No
1. Book Store (DEBSU LIT)	2
2. Introduction and Objectives of the Project	2
3. Project Category (RDBMS/OOPS/Networking/Multimedia/Artificial Intelligence/Expert Systems etc.)	3
4. Tools/Platform, Hardware and Software Requirement specifications	3
5. Goals of Implementation	3 - 4
6. SDLC Process Applied	4 - 5
7. Data Model	5
8. Functional Requirements (Use Case Diagram)	6 - 10
9. Non-functional Requirements	10 – 15
10. Feasibility Study	16
11. Project Planning	17
12. Project Scheduling	17
13. Software Engineering Paradigm applied a. Data Flow Diagram (DFD)	18 – 20
14. Schema/Database Design (project)	21 – 22
15. User Interface Design	23 – 35
16. Coding	36 – 44
17. Testing	45 – 50
18. System Security measures (Implementation of security for the project developed)	51
19. Database/Data security	51
20. Creation of User profiles and access rights	51
21. Cost Estimation of the Project along with Cost Estimation Model	51 – 52
22. Future scope and further enhancement of the Project	52
23. Bibliography	53 – 54

1.Title of the Project

Development of a feature-rich, Book Store .

2. Introduction and Objectives of the Project

This project is aimed at developing an that provides a platform for book enthusiasts to buy and sell books. The system (BOOK STORE) allows registered users to browse through a curated collection of books based on categories, authors, or titles. Users can access details about books, such as descriptions and pricing, and make purchases using the integrated payment system. Admin can log in to add book for listings and sharing information such as the title, price,etc.
To ensure the quality and accuracy of information, the system includes Super user manage the book details and also approve the orders for their customers.

1. **A user should be able to:**
    - Homepage Features:
        - Users should be able to browse from the homepage.
        - The homepage should display a variety of books by categorized.
    - User Authentication:
        - Users must be able to log in to the system through the homepage of the application.
        - A signup feature should allow new users to create an account, providing essential information such as name, email, and password.
    - User Profile Management:
        - Users can view and update their profile, including details such as the number of books purchased.
        - Transaction Management:
            - The platform should include functionality for secure payment processing for book purchases.

2. An admin login should be present who can read, approve as well as remove any uploads.

3.Project Category

Web Application.

4.Tools/Platform, Hardware and Software Requirement specifications.

Tools

1. VS Code, Browser ,Ms Office

Platform

1. Microsoft Windows 10/11

Hardware Requirement Specification

Client Machine		Server Machine	
HDD	200 MB	**HDD**	320 GB
Processor	Pentium 4 or newer processor that supports SSE2	**Processor**	Dual Core or newer processor
Memory	512 MB	**Memory**	2 GB

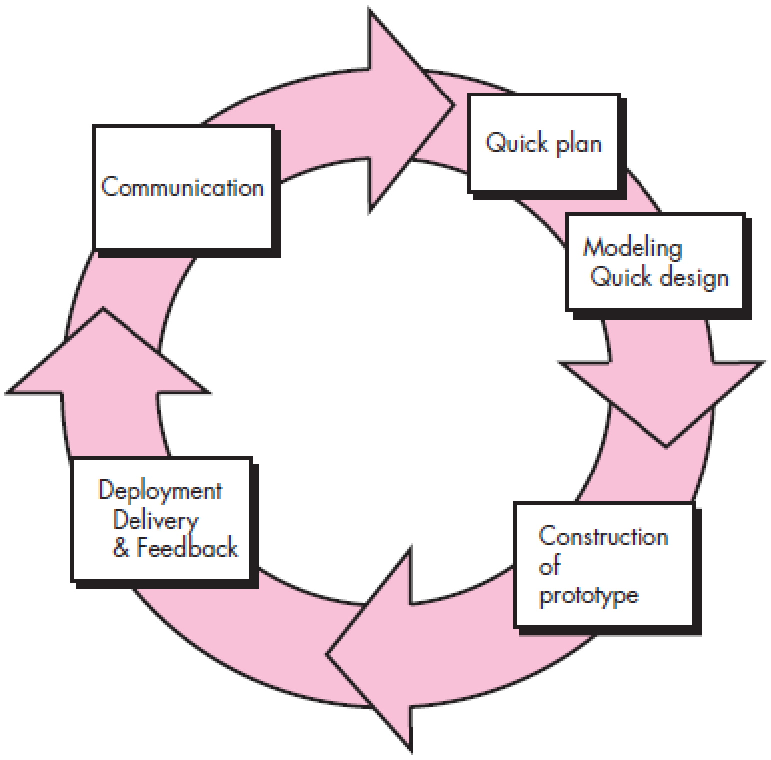
Software Requirement Specification

Client Machine		Server Machine	
Browser	Any standard browser with Javascript interpreter	**Software**	Apache
Client side mark up / scripting languages	HTML, Javascript	**Database Management System Software**	MySQL
		Specification	MySQL 4.1

5. Goals of Implementation

To offer seamless access to a wide variety of physical books, creating a convenient and delightful shopping experience for book lovers everywhere.

6. SDLC Process Applied



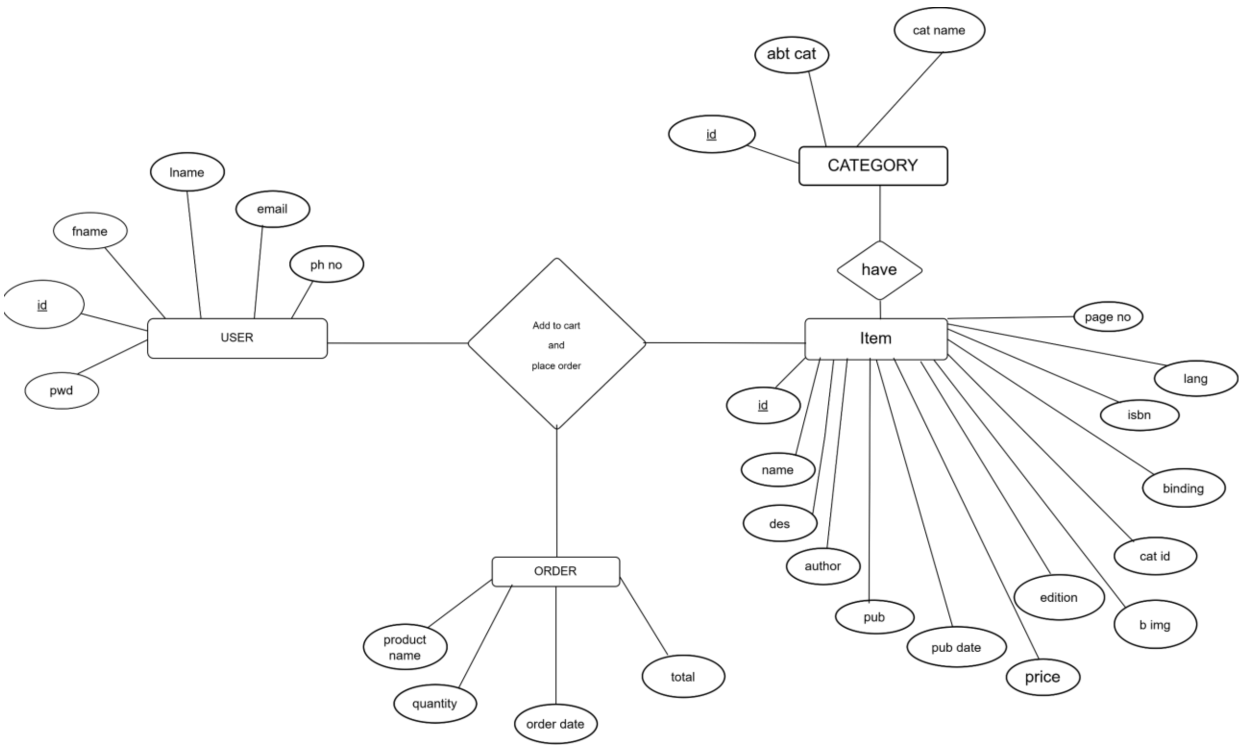
Often, a customer defines a set of general objectives for software but does not identify detailed input, processing, or output requirements. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take. In these, and many other situations, a prototyping paradigm may offer the best approach.

The prototyping paradigm begins with **requirements gathering**. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. A **"quick design"** then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats). The quick design leads to the construction of a prototype. The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

Ideally, the prototype serves as a mechanism for identifying software requirements. If a working prototype is built, the developer attempts to use existing program fragments or applies tools (e.g., report generators, window managers) that enable working programs to be generated quickly.

7. Data Model

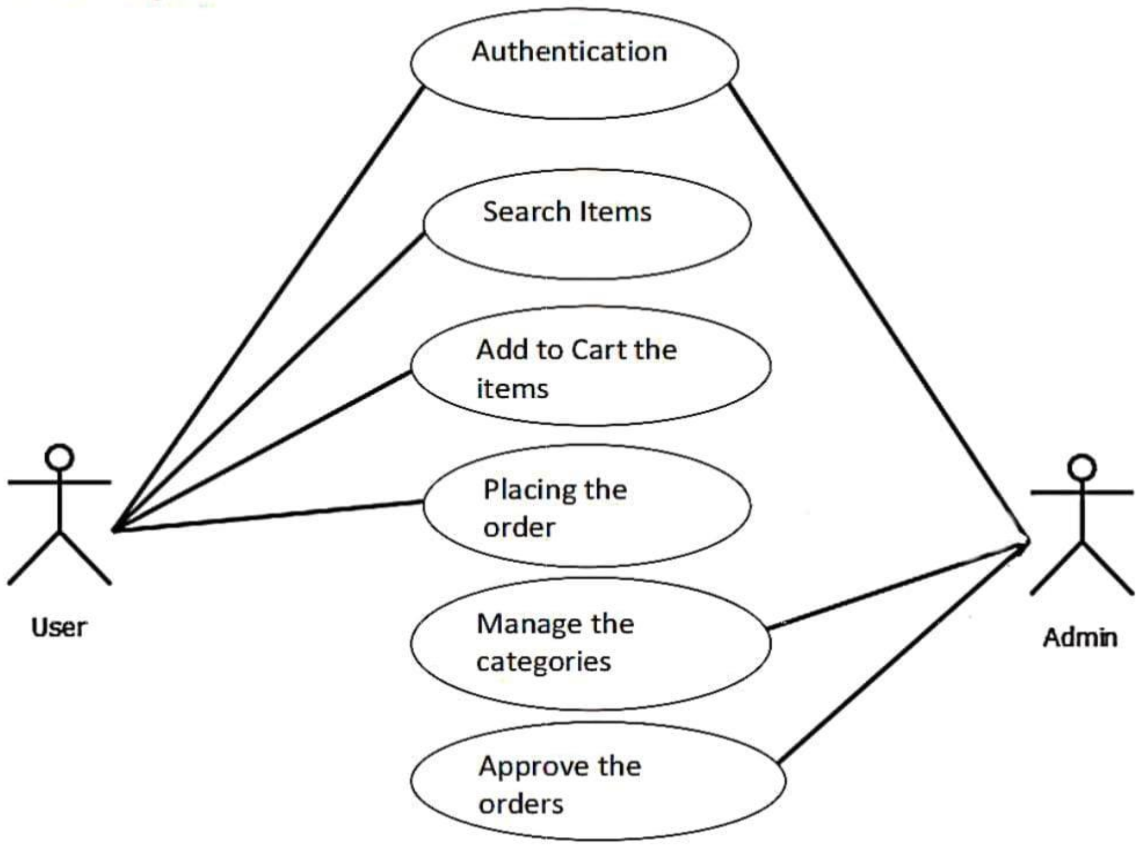
ER Diagram



8. Functional Requirements

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Nonfunctional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

Use Case Diagram



Use Case Descriptions

Use Case Name:	Authentication
Priority	Essential
Trigger	Menu selection
Precondition	User is connected to the Internet and on the DEBSU LIT home page
Basic Path	1. User enters username and password. 2. The username and password is matched with the record in the database. 3. If the authentication parameters are correct the user is directed to the user's main page, otherwise an error message is displayed.
Alternate Path	NA
Post Condition	The user is on the User Home Page
Exception Path	If there is a connection failure the server returns to the wait state

Use Case Name:	Purchase a book
Priority	Essential
Trigger	Menu selection
Precondition	User is connected to the Internet and on the user's main page
Basic Path	1. User browses or searches for a desired book. 2. User selects the book they want to purchase. 3. User clicks the "Add to Cart" button. 4. The system processes the purchase request and directs the user to the payment gateway. 5. User completes the payment process.

	6. The system confirms the purchase and order confirmation.The server side program receives the document and saves it in the server system's filesystem.
Alternate Path	NA
Post Condition	The book is successfully purchased and marked as sold.
Exception Path	If there is a payment failure, the user is notified, and the system prompts them to retry or use another payment method.

Use Case Name:	Place Order
Priority	Essential
Trigger	Menu selection
Precondition	User is connected to the Internet and on the user's main page
Basic Path	1. User reviews the items in their cart. 2. User clicks the "Proceed to Checkout" button. 3. User enters or confirms their delivery address. 4. User selects Razorpay as the payment method. 5. The system redirects the user to the Razorpay payment gateway. 6. User completes the payment process on Razorpay. 7. The system confirms the order and generates an order confirmation for the user.
Alternate Path	NA
Post Condition	The order is successfully placed, and payment is confirmed.
Exception Path	If there is a payment failure, the user is redirected to the payment retry page with an error notification.

Use Case Name:	Manage Categories

Priority	Essential
Trigger	Menu selection
Precondition	Admin is connected to the Internet and on the admin’s main page
Basic Path	1. Admin selects the "Manage Categories" link from the menu. 2. The server-side program retrieves the list of existing categories from the MySQL database. 3. Admin can add, edit, or delete categories:- To add a new category: Admin enters the category name and description, then clicks "Add." 4. To edit a category: Admin selects the category, modifies details, and clicks "Save." 5. To delete a category: Admin selects the category and clicks "Delete." 6. The system updates the MySQL database with the changes.
Alternate Path	NA
Post Condition	The category list is updated in the database as per the admin’s actions.
Exception Path	If there is a connection failure, the server returns to a wait state, and changes are not saved.

Use Case Name:	Approving Order
Priority	Essential
Trigger	Menu selection
Precondition	User is connected to the Internet and on the user’s main page
Basic Path	1. User selects an order from the list. 2. A confirmation dialog appears with the order details for review. 3. User reviews the details and clicks "Confirm" in the dialog. 4. The server-side program processes the order and updates its status to "Approved."

	5. User receives a confirmation message or notification about the successful approval.
Alternate Path	NA
Post Condition	The order is approved and reflected in the "Approved Orders" section.
Exception Path	If a connection failure occurs, the system displays a notification and retries automatically or waits for user intervention.
Alternate Path	NA
Post Condition	The admin suspends a document.
Exception Path	If there is a connection failure the server returns to the wait state

9.Non-Functional Requirements

In addition to the features and functionality, these "Quality Attributes" will ensure the system's robustness and usability. These attributes are essential characteristics, rather than individual features, and will guide the overall development and deployment of the system.

Performance Requirements

- The application will restrict the document (e.g., book previews or uploads) size to 5 MB for optimal performance.
- The system should be able to handle up to 100 concurrent user transactions with a response time of less than 2 seconds.

Operating Constraints

- The application must run without any manual intervention.
- Automated periodic backups must occur every 24 hours to prevent data loss.

Platform Constraints

- As the application is developed in Django, it will be platform-independent and deployable on any environment supporting Python (e.g., Linux, Windows, or macOS).
- It should be compatible with modern web browsers, such as Chrome, Firefox, Safari, and Edge.

Modifiability

- Requirement: The system should accommodate future changes with minimal effort. This includes updating book categories, revising pricing strategies, or introducing new promotional offers.

- - Measurement: Changes should not require more than minimal personnel effort (person-months) to implement. The architecture should support modular updates for swift adaptability without causing disruptions to other functionalities.
- - Example: A book publisher's addition or removal from the database should reflect instantly without requiring code modifications.

Portability

- Requirement: The platform should easily migrate to different operating systems, devices, or environments (e.g., transitioning from a local host to a cloud platform or switching from Android to iOS).

- Measurement: Minimal effort (person-months) or limited module changes required for portability.

- Example: Moving from a web-based system to include mobile compatibility with the same backend infrastructure.

Reliability

- Requirement: Ensure system downtime is minimal, specifying "mean time between failures" (MTBF) as a metric. Failures could include server crashes, disrupted user sessions, or payment errors.

- Strategy: Implement robust logging mechanisms, redundancy setups, and fail-safe modules to minimize the impact of failures and to quickly recover from them.

- Example Consequences: If the payment gateway fails, users should be provided with alternative options, and incomplete orders should remain saved for future completion.

Security

- Requirement: Implement strict access control, including user authentication and session management.- Only registered users should access features like book purchases or user dashboards.

- Public access should be limited to generic browsing and search functionalities.

- Measurement: Use penetration testing to assess vulnerabilities, measuring the level of effort or skill needed for unauthorized access.

- Examples:- Enforce HTTPS for all website interactions.

- Prevent direct URL access to internal or restricted pages without authentication.

Usability

- Requirement: Provide an intuitive and user-friendly experience for both customers and administrators. Focus on streamlining navigation, minimizing the number of clicks to complete actions like purchases, and offering comprehensive user manuals.

- Measurement: Learning time for first-time users should be minimal (e.g., proficiency achieved within 30-60 minutes).

- Example: Implement tooltips, guided tours, and search auto-suggestions to make navigation seamless.

Legal

- Requirement: Adhere to all applicable laws, including:- Data privacy regulations: Safeguard customer data under rules like GDPR or CCPA.

- Intellectual property rights: Ensure all listed books have proper authorization for sale and distribution.

- Export restrictions: Manage digital rights management (DRM) for electronic books based on geographic locations.

- Strategy: Periodically review legal compliance with evolving laws to ensure adherence.

Feasibility Study

Product

- Description: A robust, interactive platform enabling users to browse, purchase, and review books dynamically. Sellers can manage their inventory, and buyers benefit from personalized recommendations, seamless checkout experiences, and multiple payment options.

- Business Value: This system aligns with organizational goals to modernize book retail, expand market reach, and improve customer engagement

Technical Feasibility

- Proposal: Utilize modern technologies such as:- Backend: Frameworks like Django or Laravel.

- Frontend: React.js or Angular for a responsive user experience.

- Database: MySQL or MongoDB to store user data, book inventories, and transaction logs.

- Hosting: Cloud-based platforms like AWS or Azure for scalability and reliability.

- Adaptability: Ensure that the architecture is modular to allow future enhancements without system-wide overhauls.

Social Feasibility

- Impact on Workforce:- Minimal retraining required for administrators transitioning from traditional systems to the new platform.

- Backend operators and data managers will find intuitive admin panels for efficient data entry and management.

- No relocation needed, and roles like customer service remain unaffected.

- Strategy: Foster user cooperation by conducting interactive workshops and providing 24/7 support during the transition.

Economic Feasibility

- Costs Incurred:- Development Costs: Programming, design, and deployment.

- Maintenance Costs: Hosting, database management, and customer support.

- Marketing Costs: Campaigns to attract customers and promote books.

- Benefits Achieved:- Direct Benefits: Increased sales volume, reduced operational costs, and streamlined inventory management.

- Indirect Benefits: Enhanced user satisfaction through personalized recommendations and a seamless buying experience.

- Analysis: Use cost-benefit models like the payback period method to demonstrate profitability.

Market Research

- Needs Analysis: Growing demand for niche genres, indie authors, and regional language books among students, avid readers, and collectors.

- Competitive Analysis: Evaluate competitors like Amazon or Flipkart, emphasizing unique selling propositions such as curated book recommendations or exclusive offers.

- Customer Base: Students, educators, avid readers, and collectors across urban and rural areas.

- Strategy: Leverage differentiators like:- Faster delivery for physical books.

- Exclusive digital content for e-books.

- High-quality customer service.

- Interactive features like book previews and reviews.

Alternative Solutions

- Solution A: Static Website- Benefits: Low development cost and quick deployment.

- Drawbacks: Requires manual updates for inventory and offers, lacks scalability.

- Solution B: Mobile-Only Application- Benefits: High convenience and optimized for mobile users.

- Drawbacks: Excludes desktop and tablet users, limited screen space for features

- Proposed Solution: Dynamic Responsive Website- Benefits: Combines cross-platform compatibility, scalability, and ease of use.

- Drawbacks: Higher development cost compared to static websites.

- Justification: A dynamic, responsive website is the most effective solution to ensure scalability, adaptability, and user satisfaction.

Feasibility Study	
System: DEBSU LIT	Date: 11/02/2025
Author: Subhojit Chakraborty	Page: 1
Product	
The project requires a web application to be developed that will enable seamless online book browsing, purchasing, and sharing reviews.	
Technical Feasibility	
The web application will be developed using Django and MySQL. The development team is skilled and experienced in these technologies, ensuring the application is built efficiently and reliably.	
Social Feasibility	
Some training for users and administrators may be required, but since all users are IT-literate, the learning curve is expected to be minimal, allowing for quick adoption.	
Market Research	
Market research indicates that the application would be beneficial to book enthusiasts and sellers, as it will provide a convenient and user-friendly platform to browse, purchase, and review books online.	
Economic Feasibility	
The application can be developed within the allocated budget, making it a financially viable project.	
Alternate Solution	
An offline desktop application is possible but would limit book browsing and purchasing to local systems, reducing accessibility and scalability.	

11. Project Planning

Project planning is concerned with identifying the following for every project:

- Activities
- Milestones
- Deliverables.
- The project plan will guide the development of the book-selling website. It will include clear objectives, timelines, and steps like specification, design, implementation, testing, and delivery. The initial plan will evolve based on feedback and challenges to ensure the system meets user needs efficiently.

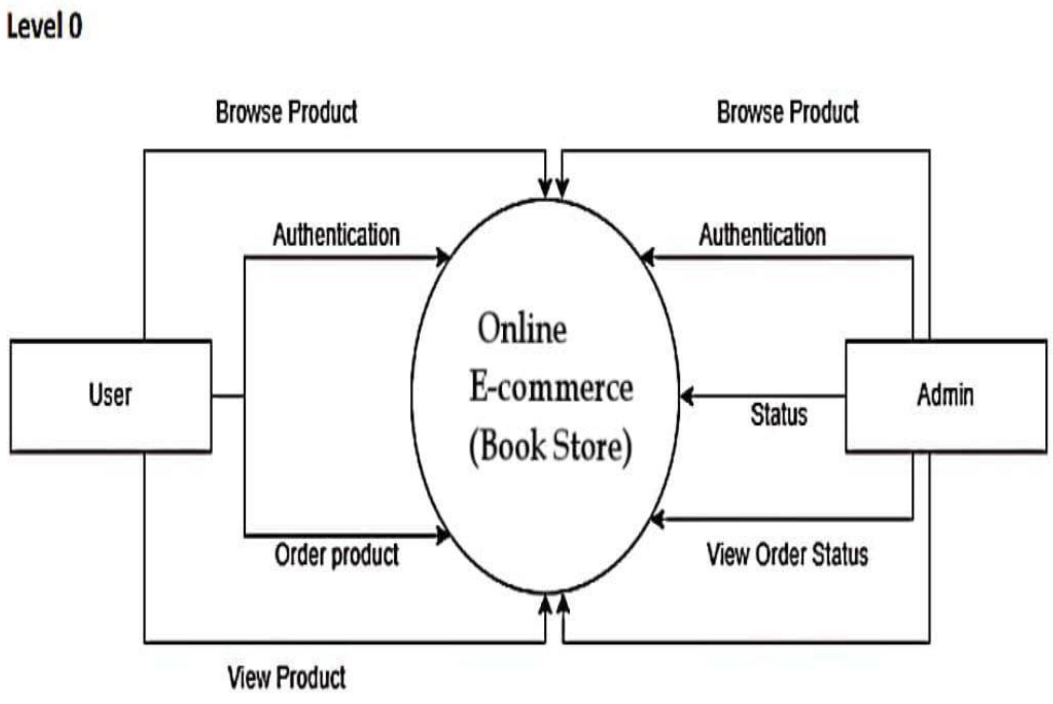
12. Project Scheduling

GANTT chart

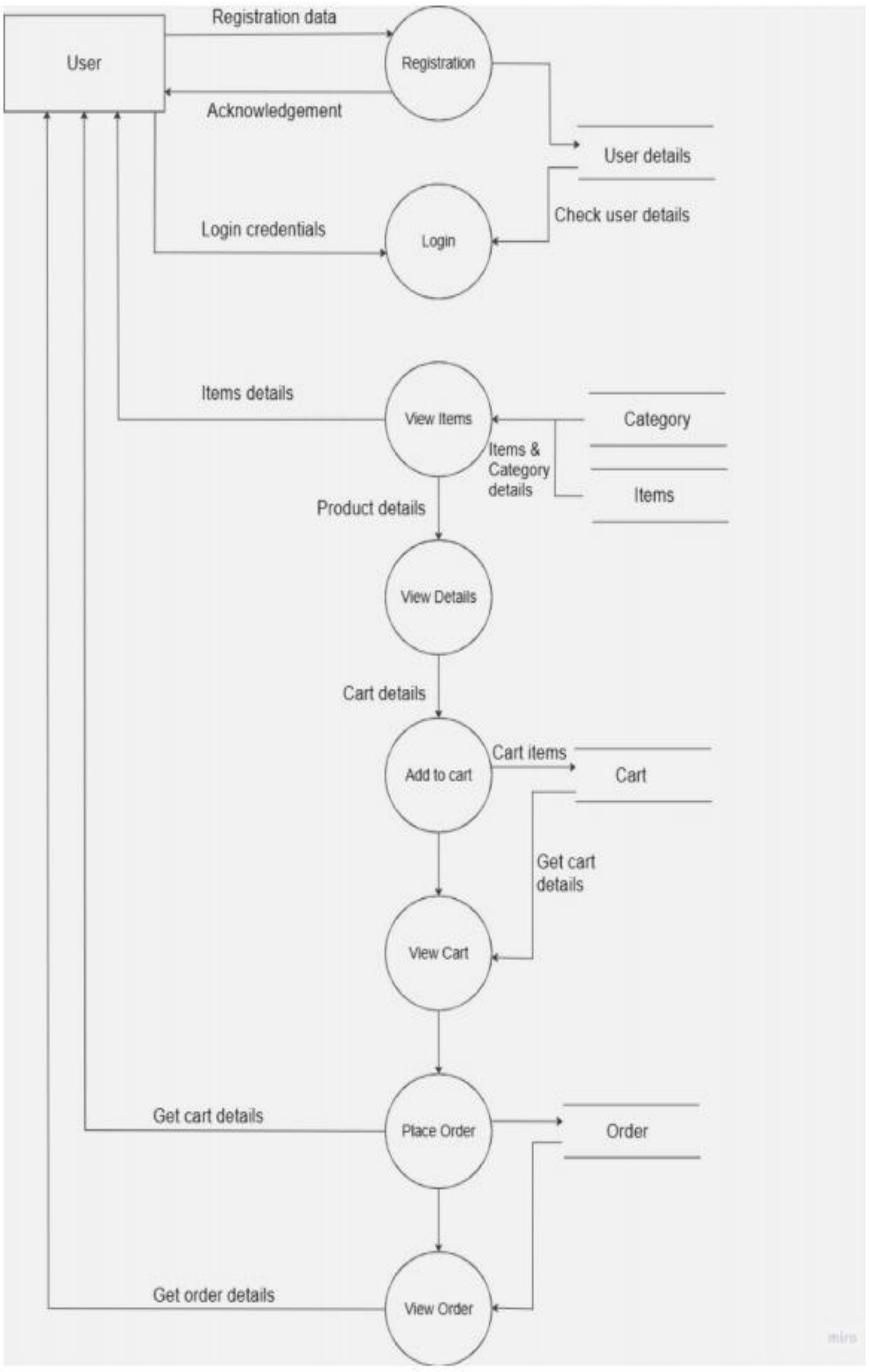
Task	Person(s) Responsible	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Communication		▨					
Quick Plan		▨					
Modeling Quick Design			▨	▨			
Construction of Prototype				▨	▨	▨	▨
Deployment, Delivery and Feedback						▨	▨

13. Software Engineering Paradigm Applied

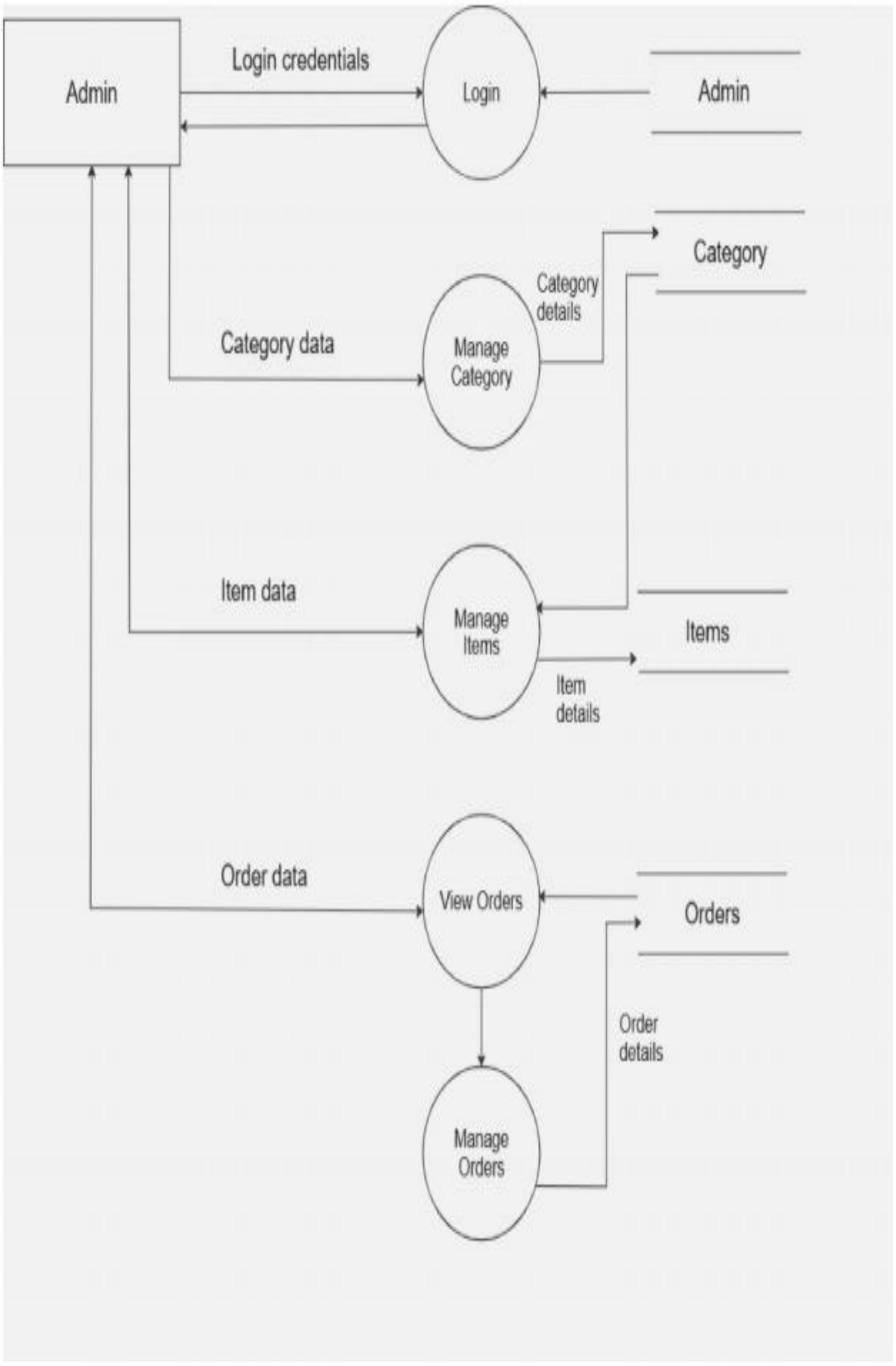
Data Flow Diagrams



LEVEL1 for USER



LEVEL1 for ADMIN



14. Schema/Database Design (project)

Table Name customuser:-

Name of Type Attribute	Data Type	Key	Description
id	Bigint(20)	Primary Key	Unique identifier
password	Varchar(128)		Authentication key
username	Varchar(150)		Login name
first_name	Varchar(150)		User name
last_name	Varchar(150)		User last name
email	Varchar(254)		Contact Adress
mobile	Varchar(10)		Phone number

Table Name category:-

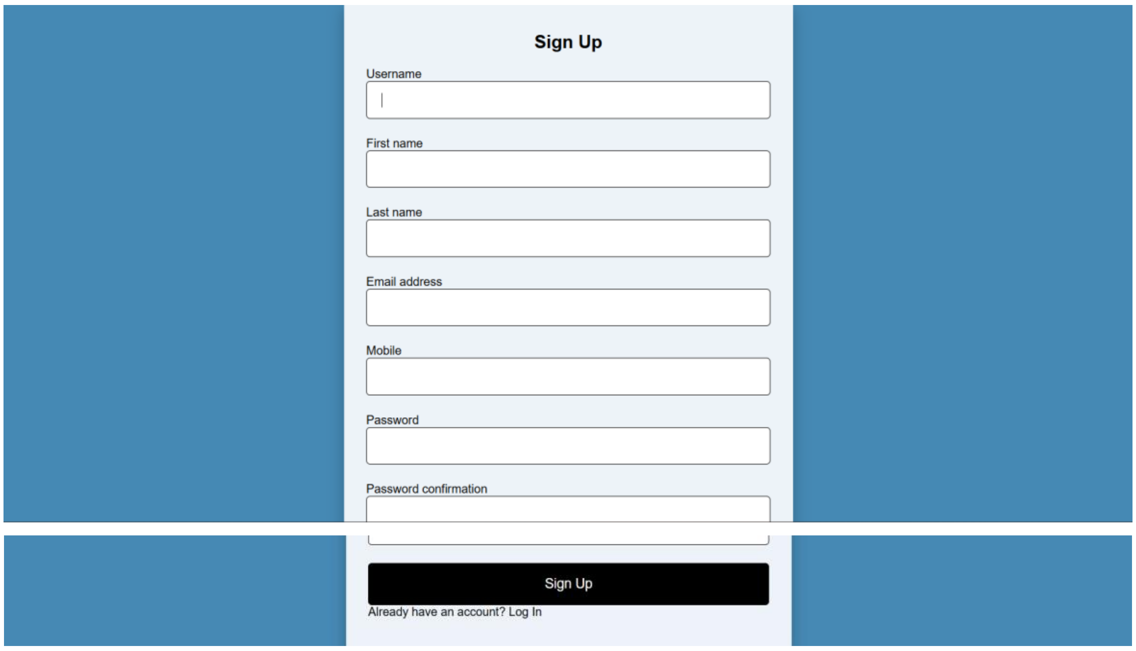
Name of Type Attribute	Data Type	Key	Description
id	BigInt(20)	Primary Key	Item id
abt_cat	longtext		About the Item
cat_name	Varchar(255)		Item Name

Table Name item:-

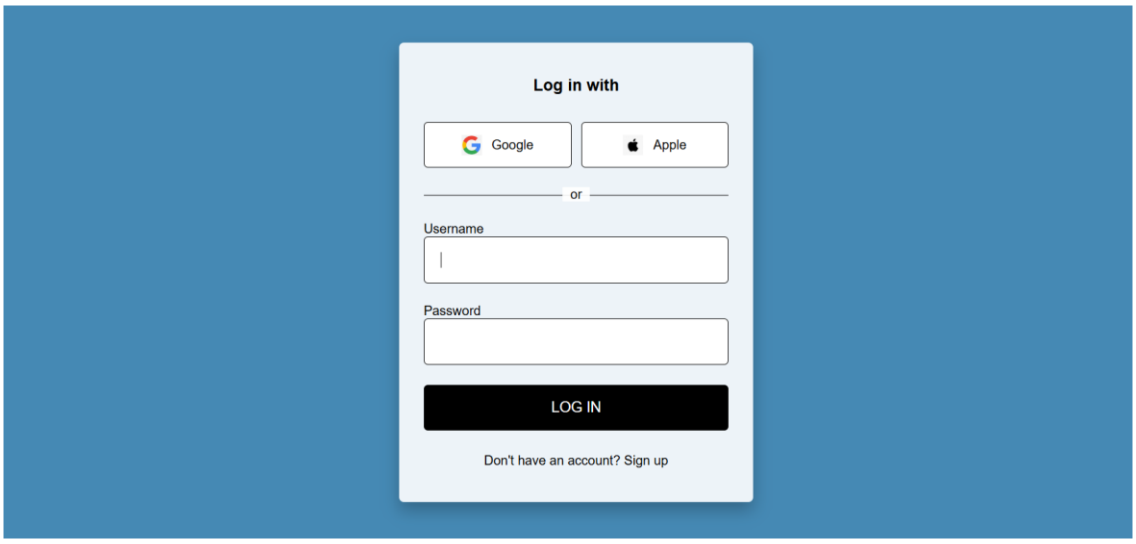
Name of Type Attribute	**Data Type**	**Key**	**Description**
id	Bigint(20)	Primary Key	Item Id
name	Varchar(255)		Item Name
description	Longtext		Item Description
author	Varchar(255)		Author Of The Item
publisher	Varchar(255)		Publisher Of The Item
pub_date	Varchar(10)		Item Publishing Date
price	Varchar(255)		Item Price
edition	Varchar(100)		Item Edition
b_img	Varchar(100)		Image Of The Items
category_id	bigint(20)	Primary Key	Items Id
binding	Varchar(50)		Item Binding Back

15. User Interface Design

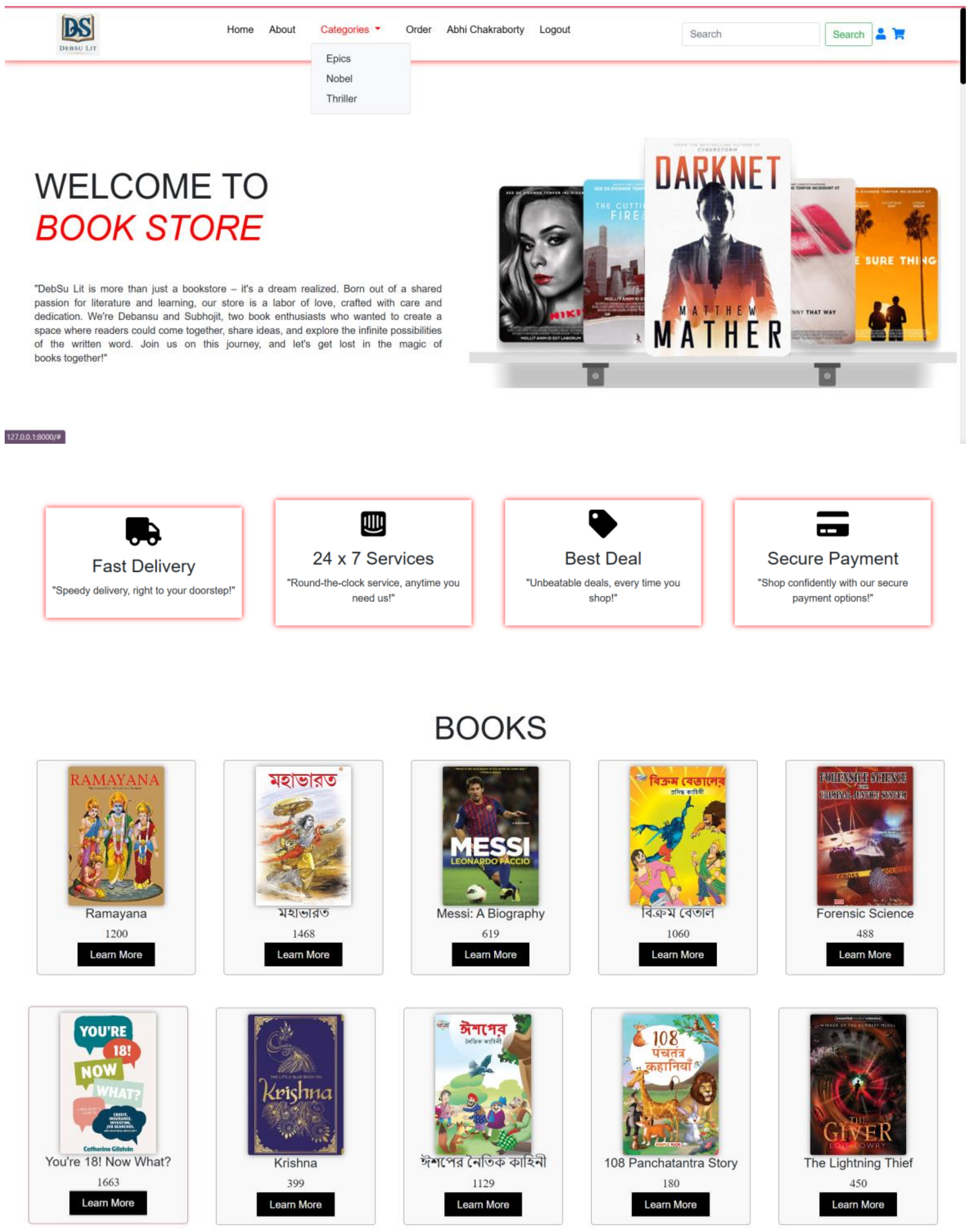
- **Sign up page**



- **Login page:**



- **Home Page:**



WELCOME TO
BOOK STORE

"DebSu Lit is more than just a bookstore – it's a dream realized. Born out of a shared passion for literature and learning, our store is a labor of love, crafted with care and dedication. We're Debansu and Subhojit, two book enthusiasts who wanted to create a space where readers could come together, share ideas, and explore the infinite possibilities of the written word. Join us on this journey, and let's get lost in the magic of books together!"

Fast Delivery
"Speedy delivery, right to your doorstep!"

24 x 7 Services
"Round-the-clock service, anytime you need us!"

Best Deal
"Unbeatable deals, every time you shop!"

Secure Payment
"Shop confidently with our secure payment options!"

BOOKS

Ramayana
1200
Learn More

মহাভারত
1468
Learn More

Messi: A Biography
619
Learn More

বিক্রম বেতাল
1060
Learn More

Forensic Science
488
Learn More

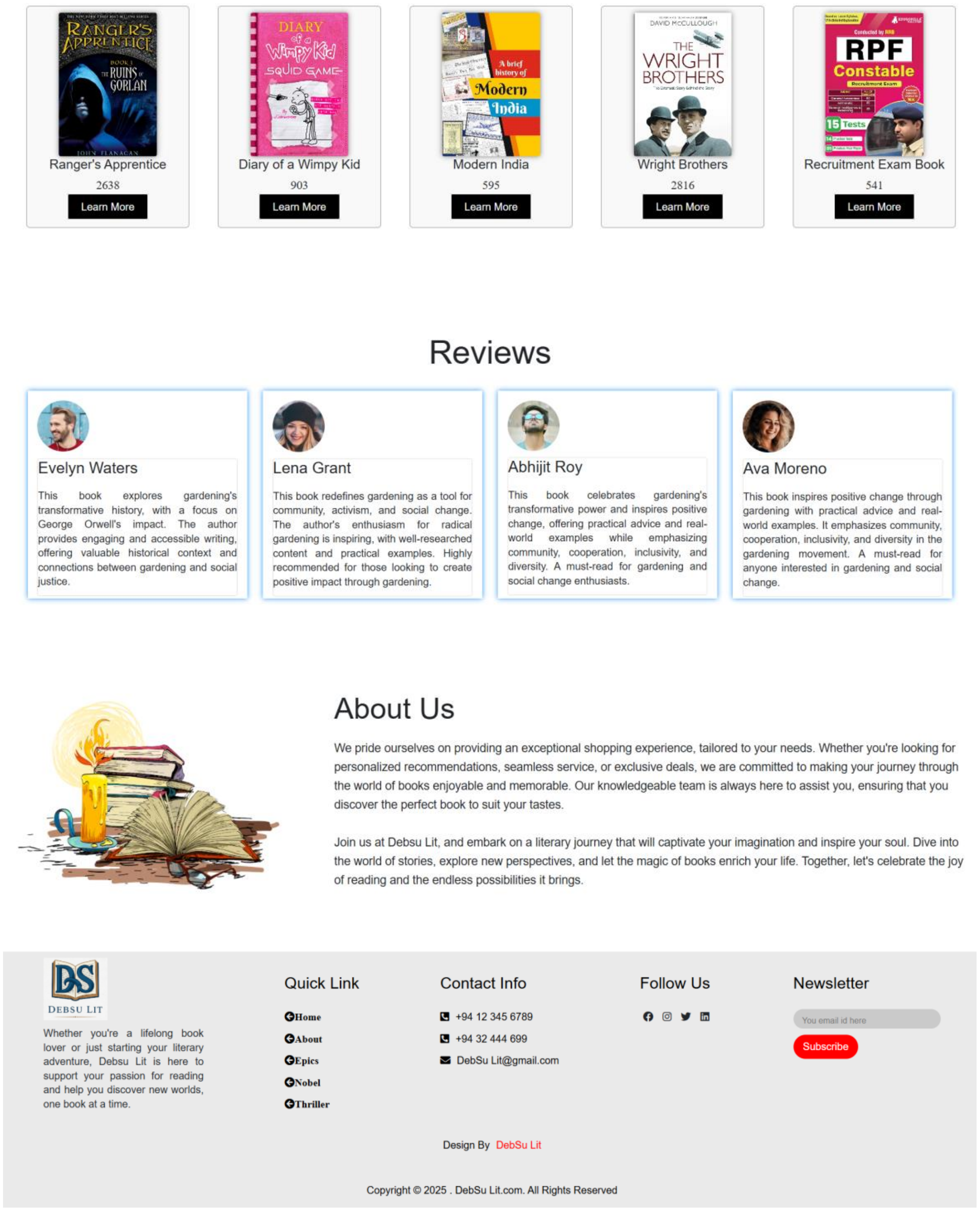
You're 18! Now What?
1663
Learn More

Krishna
399
Learn More

ঈশপের নৈতিক কাহিনী
1129
Learn More

108 Panchatantra Story
180
Learn More

The Lightning Thief
450
Learn More



Ranger's Apprentice
2638
Learn More

Diary of a Wimpy Kid
903
Learn More

Modern India
595
Learn More

Wright Brothers
2816
Learn More

Recruitment Exam Book
541
Learn More

Reviews

Evelyn Waters

This book explores gardening's transformative history, with a focus on George Orwell's impact. The author provides engaging and accessible writing, offering valuable historical context and connections between gardening and social justice.

Lena Grant

This book redefines gardening as a tool for community, activism, and social change. The author's enthusiasm for radical gardening is inspiring, with well-researched content and practical examples. Highly recommended for those looking to create positive impact through gardening.

Abhijit Roy

This book celebrates gardening's transformative power and inspires positive change, offering practical advice and real-world examples while emphasizing community, cooperation, inclusivity, and diversity. A must-read for gardening and social change enthusiasts.

Ava Moreno

This book inspires positive change through gardening with practical advice and real-world examples. It emphasizes community, cooperation, inclusivity, and diversity in the gardening movement. A must-read for anyone interested in gardening and social change.

About Us

We pride ourselves on providing an exceptional shopping experience, tailored to your needs. Whether you're looking for personalized recommendations, seamless service, or exclusive deals, we are committed to making your journey through the world of books enjoyable and memorable. Our knowledgeable team is always here to assist you, ensuring that you discover the perfect book to suit your tastes.

Join us at Debsu Lit, and embark on a literary journey that will captivate your imagination and inspire your soul. Dive into the world of stories, explore new perspectives, and let the magic of books enrich your life. Together, let's celebrate the joy of reading and the endless possibilities it brings.

DEBSU LIT

Whether you're a lifelong book lover or just starting your literary adventure, Debsu Lit is here to support your passion for reading and help you discover new worlds, one book at a time.

Quick Link

Home
About
Epics
Nobel
Thriller

Contact Info

+94 12 345 6789
+94 32 444 699
DebSu Lit@gmail.com

Follow Us

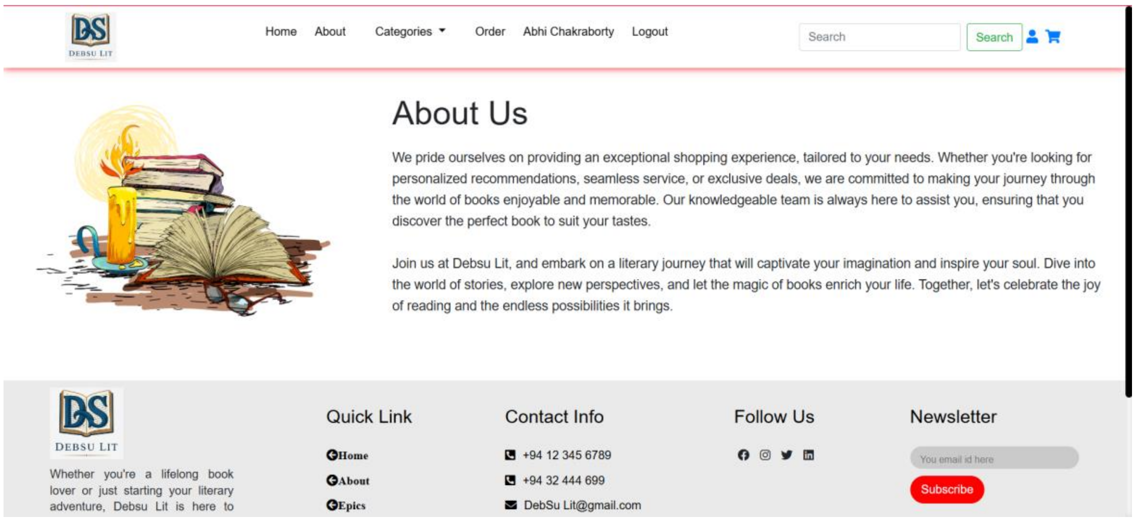
Newsletter

You email id here
Subscribe

Design By DebSu Lit

Copyright © 2025 . DebSu Lit.com. All Rights Reserved

- **About Page:**



Home About Categories Order Abhi Chakraborty Logout Search Search

About Us

We pride ourselves on providing an exceptional shopping experience, tailored to your needs. Whether you're looking for personalized recommendations, seamless service, or exclusive deals, we are committed to making your journey through the world of books enjoyable and memorable. Our knowledgeable team is always here to assist you, ensuring that you discover the perfect book to suit your tastes.

Join us at Debsu Lit, and embark on a literary journey that will captivate your imagination and inspire your soul. Dive into the world of stories, explore new perspectives, and let the magic of books enrich your life. Together, let's celebrate the joy of reading and the endless possibilities it brings.

DEBSU LIT

Whether you're a lifelong book lover or just starting your literary adventure, Debsu Lit is here to

Quick Link: Home, About, Epics

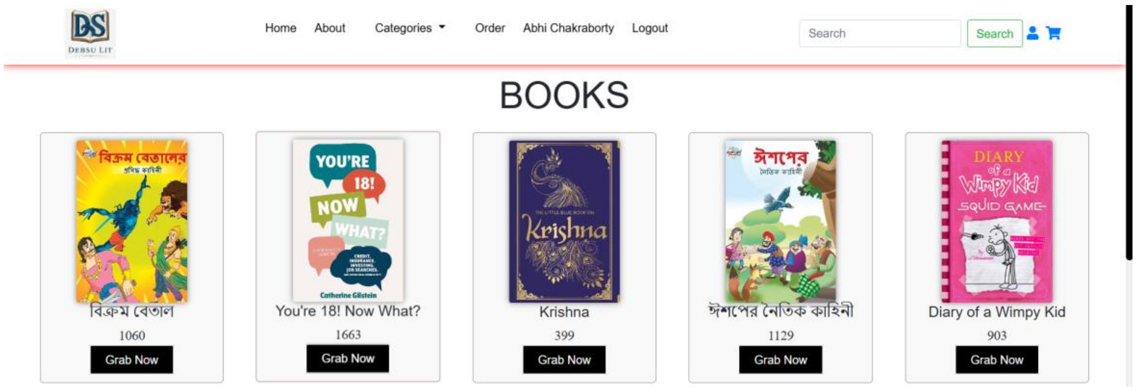
Contact Info: +94 12 345 6789, +94 32 444 699, DebSu Lit@gmail.com

Follow Us

Newsletter: You email id here, Subscribe

- <u>**Categories**</u>

- **Epics:**

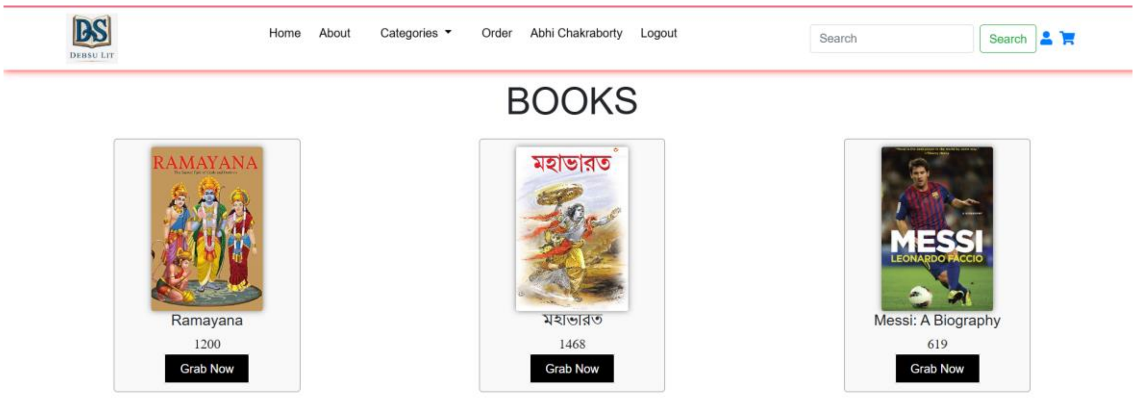


Home About Categories Order Abhi Chakraborty Logout Search Search

BOOKS

বিক্রম বেতাল	You're 18! Now What?	Krishna	ঈশপের নৈতিক কাহিনী	Diary of a Wimpy Kid
1060	1663	399	1129	903
Grab Now	Grab Now	Grab Now	Grab Now	Grab Now

- **Nobel:**



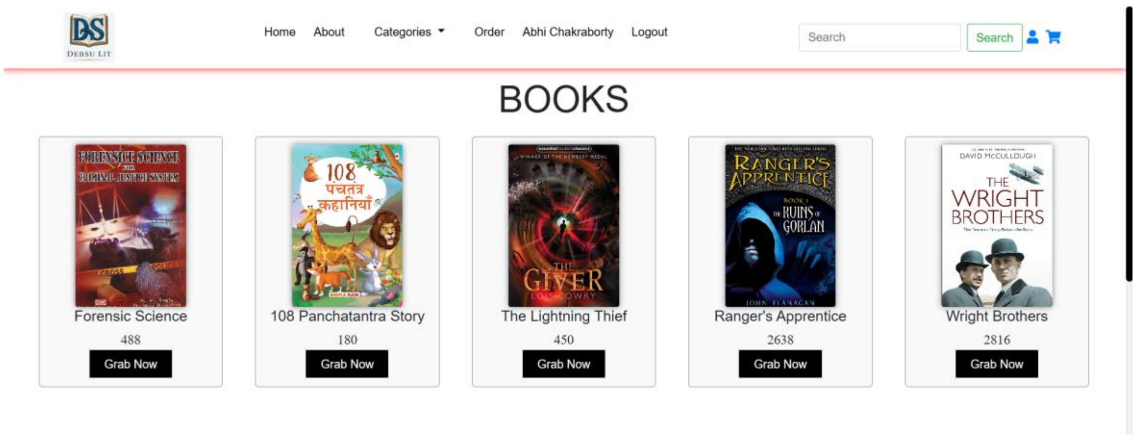
BOOKS

Ramayana
1200
Grab Now

মহাভারত
1468
Grab Now

Messi: A Biography
619
Grab Now

- **Thriller:**



BOOKS

Forensic Science
488
Grab Now

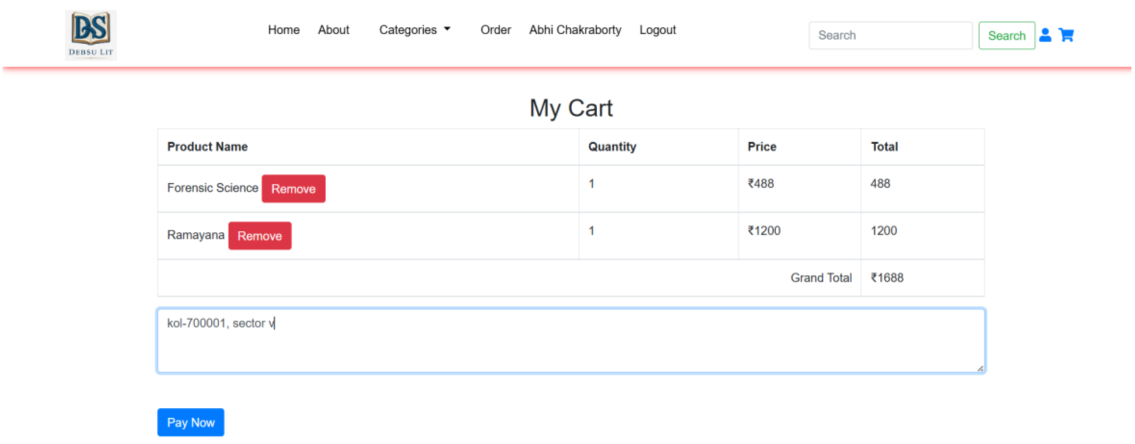
108 Panchatantra Story
180
Grab Now

The Lightning Thief
450
Grab Now

Ranger's Apprentice
2638
Grab Now

Wright Brothers
2816
Grab Now

- **Cart**



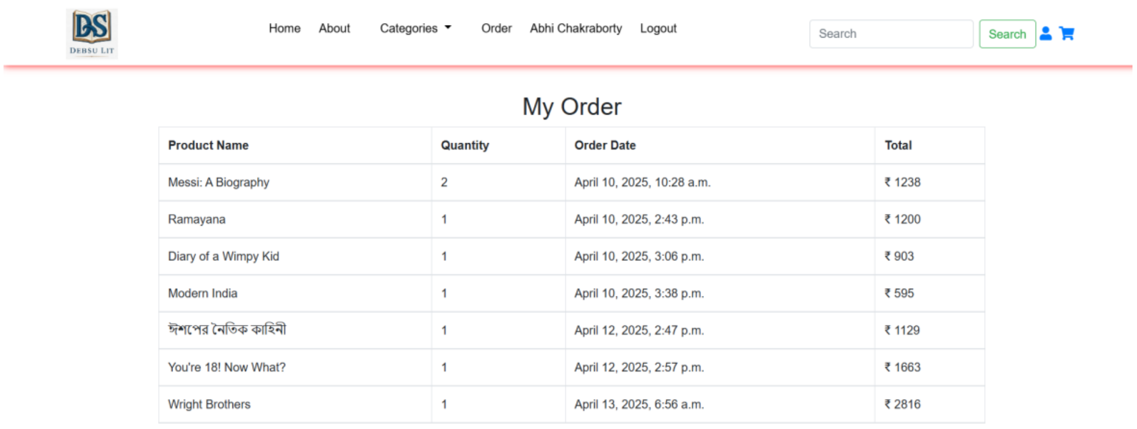
My Cart

Product Name	Quantity	Price	Total
Forensic Science Remove	1	₹488	488
Ramayana Remove	1	₹1200	1200
		Grand Total	₹1688

kol-700001, sector v

Pay Now

- **Order:**

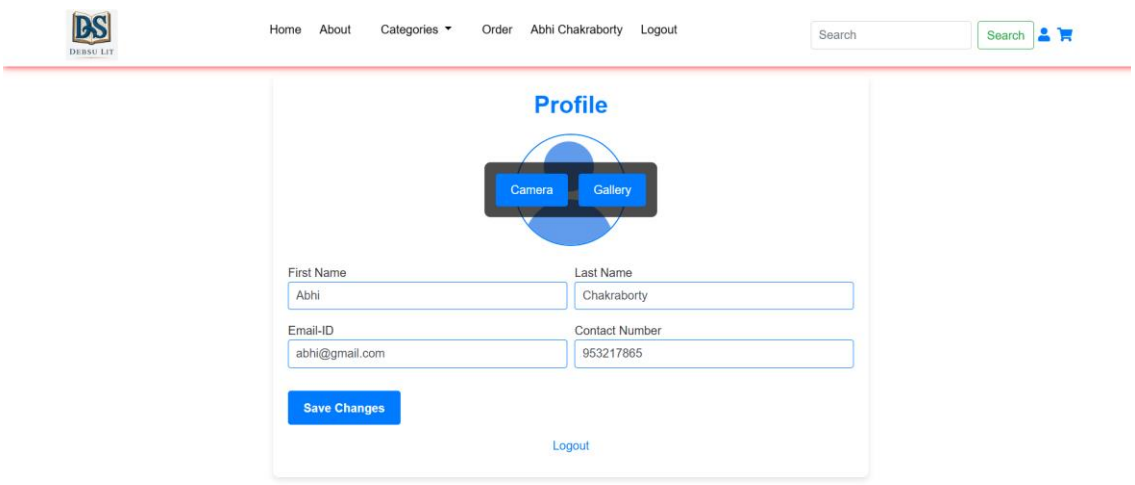


My Order

Product Name	Quantity	Order Date	Total
Messi: A Biography	2	April 10, 2025, 10:28 a.m.	₹ 1238
Ramayana	1	April 10, 2025, 2:43 p.m.	₹ 1200
Diary of a Wimpy Kid	1	April 10, 2025, 3:06 p.m.	₹ 903
Modern India	1	April 10, 2025, 3:38 p.m.	₹ 595
ঈশপের নৈতিক কাহিনী	1	April 12, 2025, 2:47 p.m.	₹ 1129
You're 18! Now What?	1	April 12, 2025, 2:57 p.m.	₹ 1663
Wright Brothers	1	April 13, 2025, 6:56 a.m.	₹ 2816

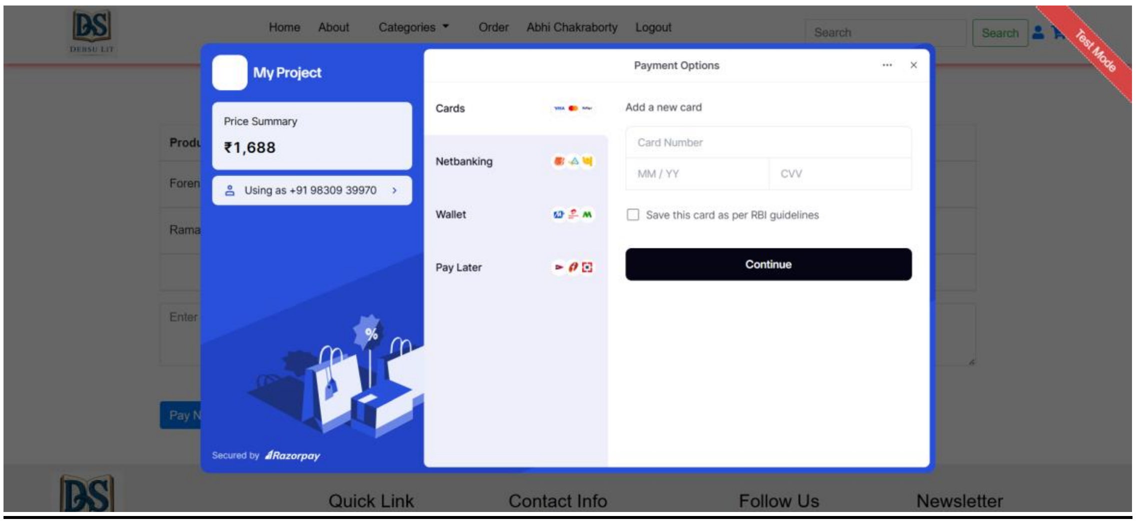
- **User account**

Abhi Chakraborty:



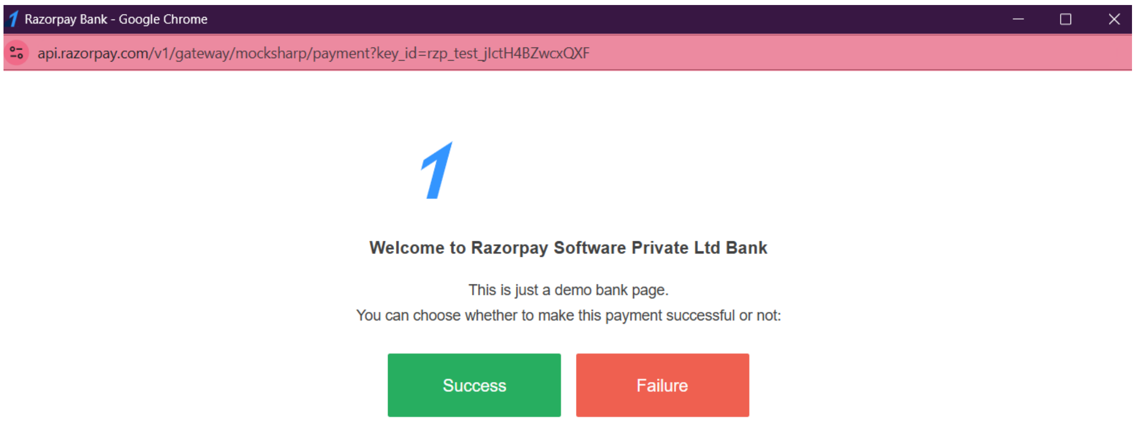
First Name: Abhi
Last Name: Chakraborty
Email-ID: abhi@gmail.com
Contact Number: 953217865

- **Payment page**



Price Summary
₹1,688
Using as +91 98309 39970

- **Payment success or failure:**

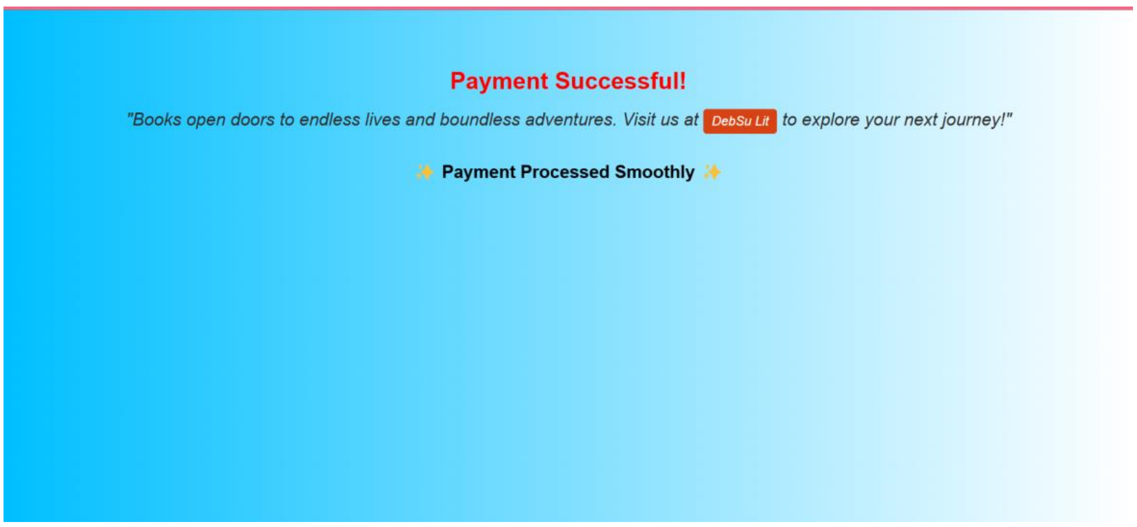

Razorpay Bank - Google Chrome

api.razorpay.com/v1/gateway/mocksharp/payment?key_id=rzp_test_jIctH4BZwcxQXF

Welcome to Razorpay Software Private Ltd Bank

This is just a demo bank page.
You can choose whether to make this payment successful or not:

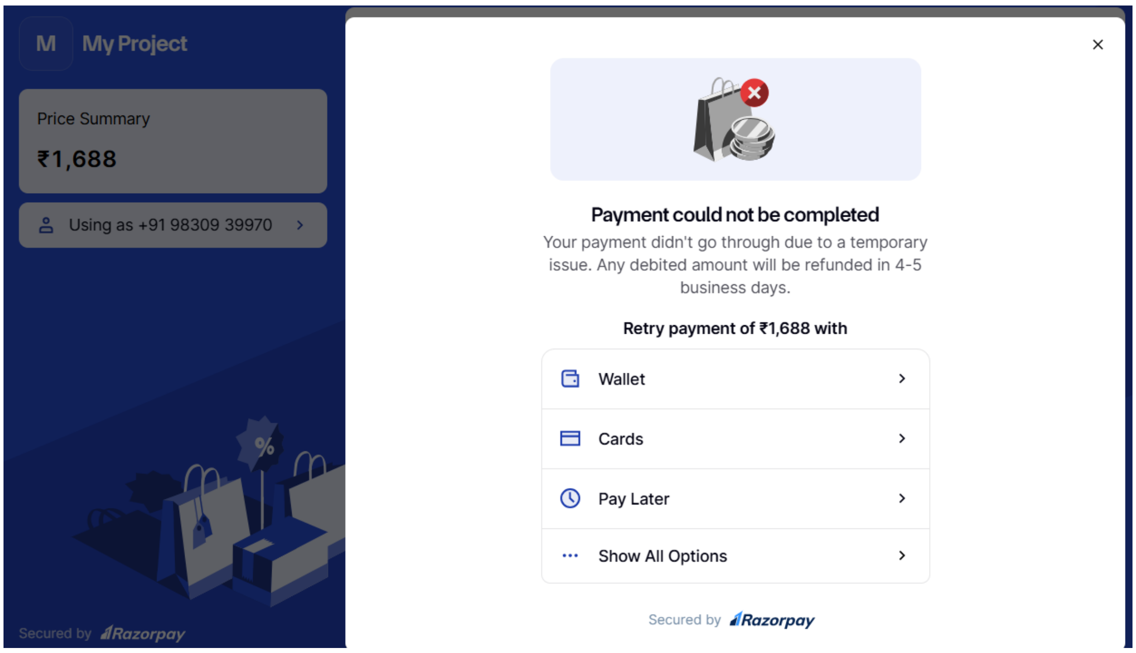
- **Payment success:**


Payment Successful!

"Books open doors to endless lives and boundless adventures. Visit us at DebSu Lit to explore your next journey!"

✨ Payment Processed Smoothly ✨

- **Payment failure:**



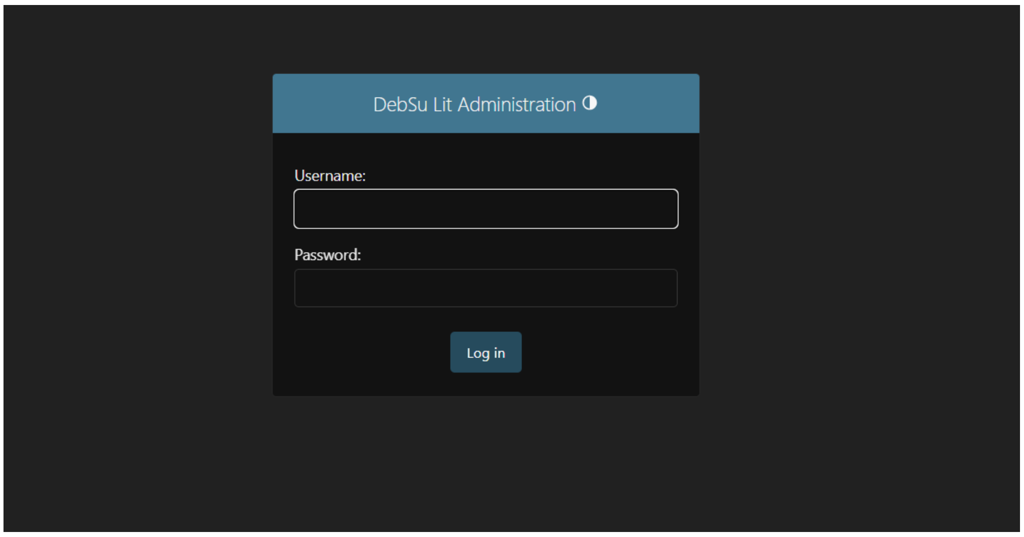
Payment could not be completed

Your payment didn't go through due to a temporary issue. Any debited amount will be refunded in 4-5 business days.

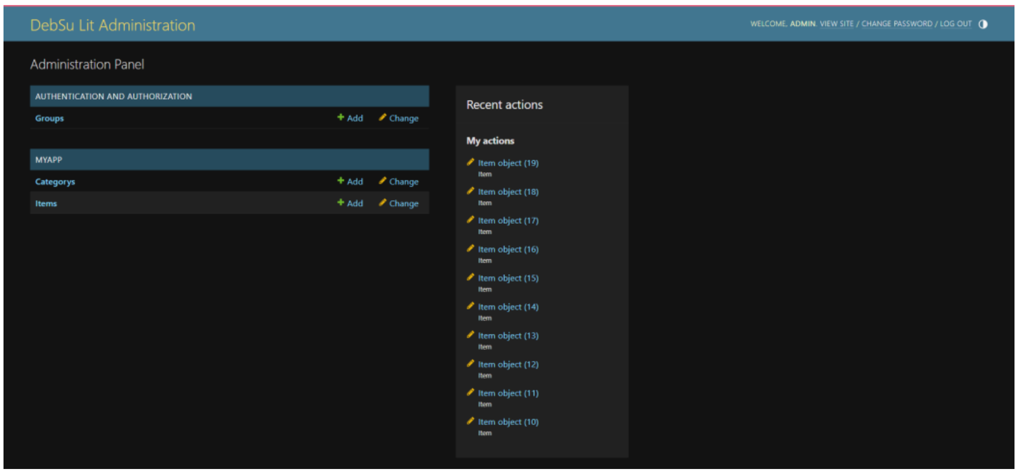
Retry payment of ₹1,688 with

- **Admin page**

Login page:



- **Welcome admin page:**



AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change

MYAPP

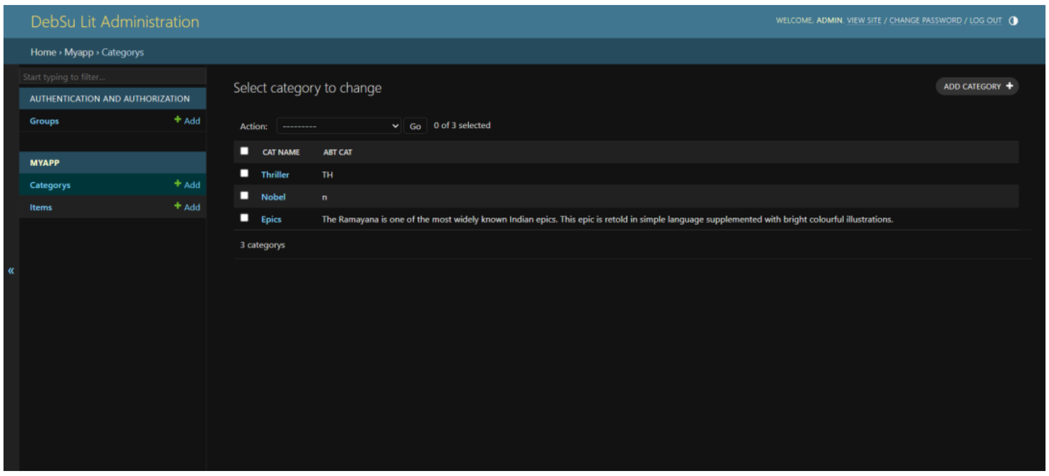
Categorys	+ Add	Change
Items	+ Add	Change

Recent actions

My actions

- Item object (19) Item
- Item object (18) Item
- Item object (17) Item
- Item object (16) Item
- Item object (15) Item
- Item object (14) Item
- Item object (13) Item
- Item object (12) Item
- Item object (11) Item
- Item object (10) Item

- **Select category to change:**



DebSu Lit Administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Myapp › Categorys

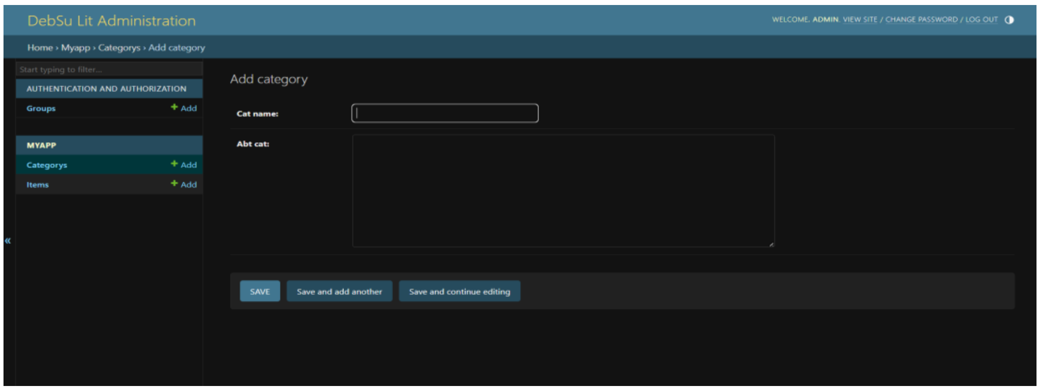
Select category to change

Action: --------- Go 0 of 3 selected

CAT NAME	ABT CAT
Thriller	TH
Nobel	n
Epics	The Ramayana is one of the most widely known Indian epics. This epic is retold in simple language supplemented with bright colourful illustrations.

3 categorys

- **Add category:**



DebSu Lit Administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Myapp › Categorys › Add category

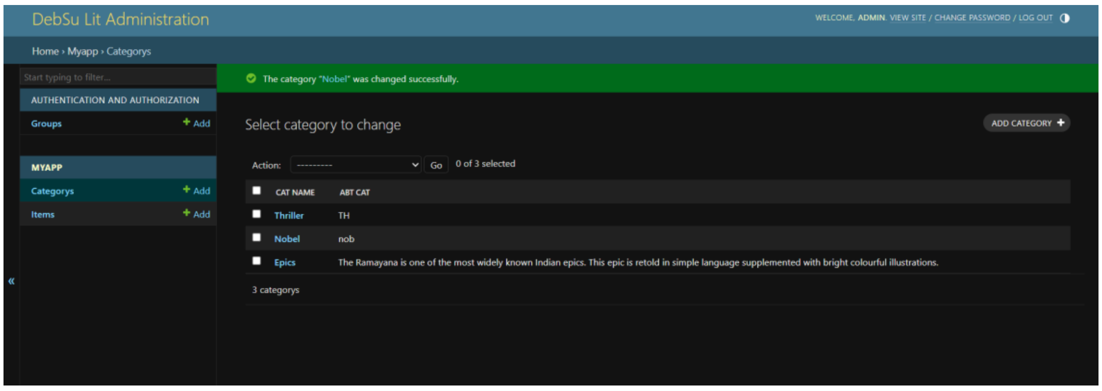
Add category

Cat name:

Abt cat:

Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
MYAPP
Categorys + Add
Items + Add
SAVE
Save and add another
Save and continue editing

- **Update category:**



DebSu Lit Administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Myapp › Categorys

The category "Nobel" was changed successfully.

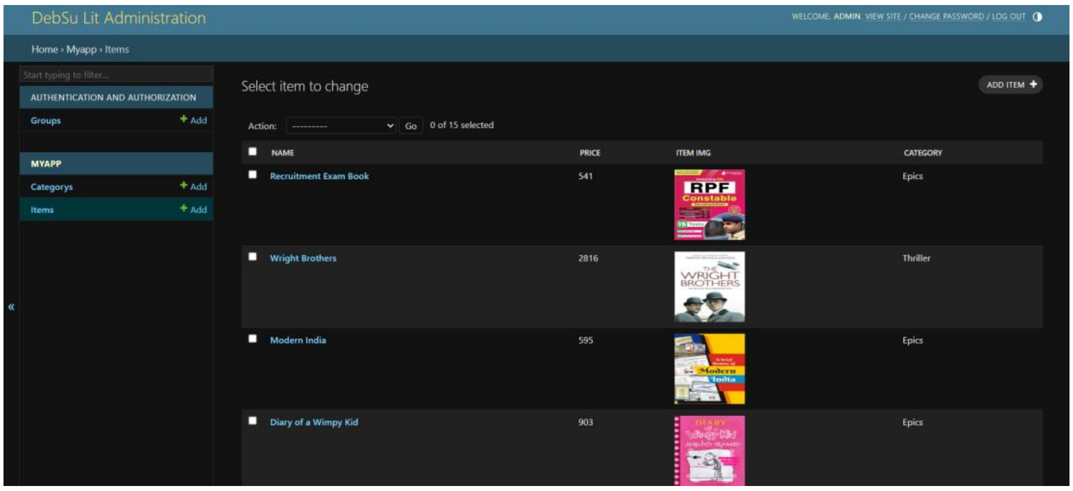
Select category to change

Action: --------- Go 0 of 3 selected

CAT NAME	ABT CAT
Thriller	TH
Nobel	nob
Epics	The Ramayana is one of the most widely known Indian epics. This epic is retold in simple language supplemented with bright colourful illustrations.

3 categorys

- **Select item to change:**



DebSu Lit Administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Myapp › Items

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

MYAPP

Categorys + Add

Items + Add

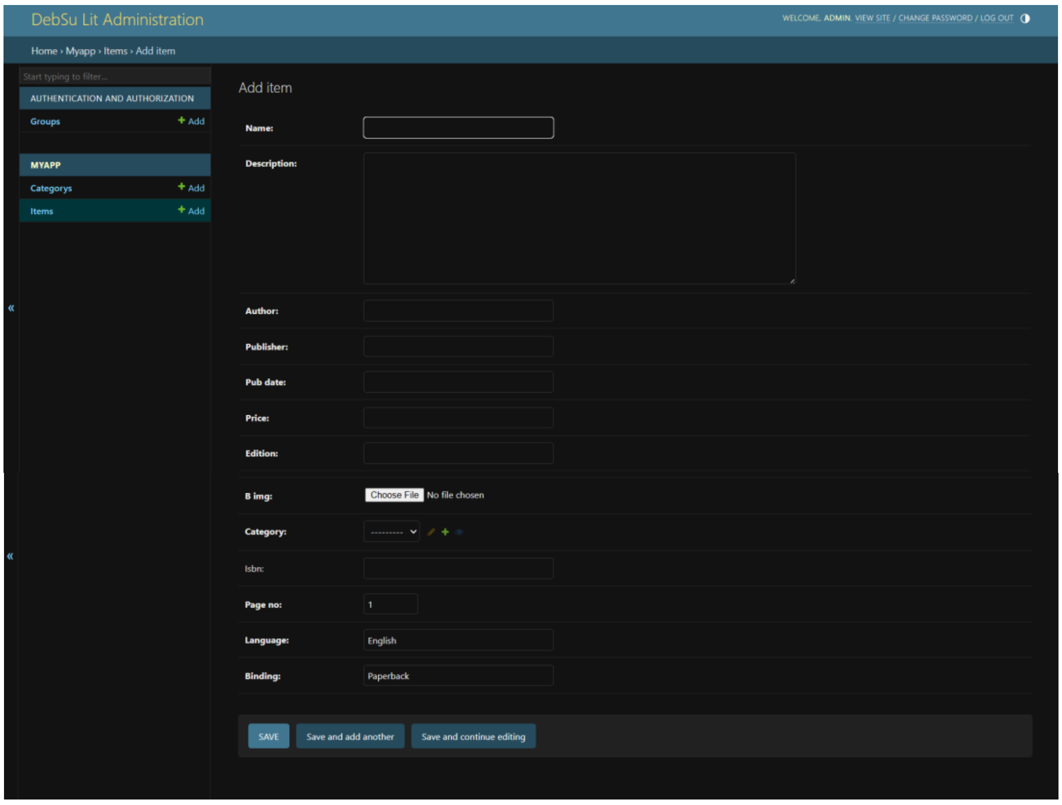
Select item to change

ADD ITEM +

Action: --------- Go 0 of 15 selected

NAME	PRICE	ITEM IMG	CATEGORY
Recruitment Exam Book	541		Epics
Wright Brothers	2816		Thriller
Modern India	595		Epics
Diary of a Wimpy Kid	903		Epics

- **Add item:**



DebSu Lit Administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Myapp › Items › Add item

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

MYAPP

Categorys + Add

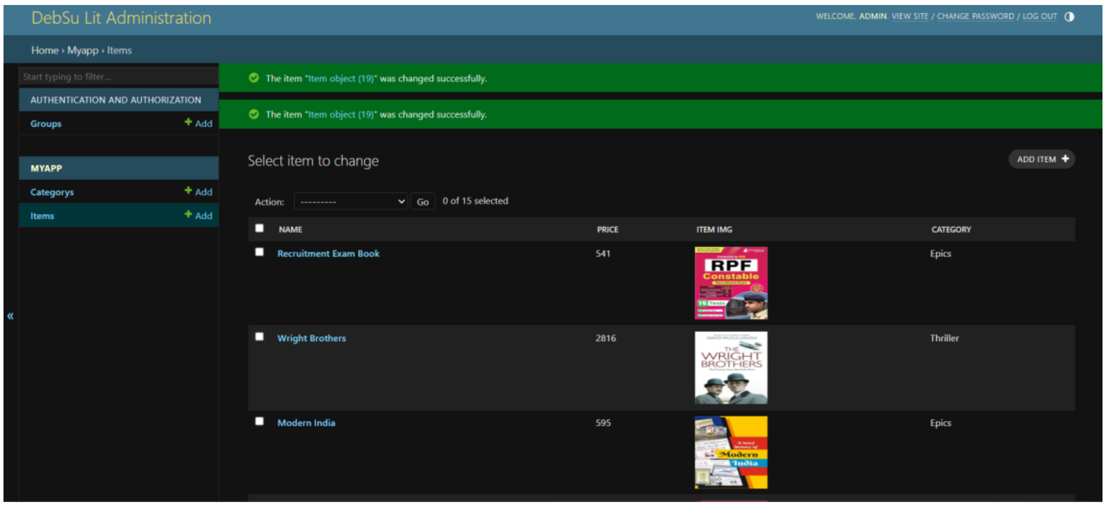
Items + Add

Add item

Name:	
Description:	
Author:	
Publisher:	
Pub date:	
Price:	
Edition:	
B img:	Choose File No file chosen
Category:	---------
Isbn:	
Page no:	1
Language:	English
Binding:	Paperback

SAVE | Save and add another | Save and continue editing

- **Update item:**



DebSu Lit Administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Myapp › Items

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

MYAPP

Categorys + Add

Items + Add

The item "Item object (19)" was changed successfully.

The item "Item object (19)" was changed successfully.

Select item to change

ADD ITEM +

Action: --------- Go 0 of 15 selected

NAME	PRICE	ITEM IMG	CATEGORY
Recruitment Exam Book	541		Epics
Wright Brothers	2816		Thriller
Modern India	595		Epics

16. Coding

- **models.py:**

```
from django.db import models
from django.contrib.auth.models import AbstractUser

# Create your models here.
class CustomUser(AbstractUser):
   mobile=models.CharField(max_length=10)

class Student(models.Model):
   name=models.CharField(max_length=200)
   email=models.CharField(max_length=255, unique=True)
   mobile=models.CharField(max_length=10)

class Category(models.Model):
   cat_name=models.CharField(max_length=255)
   abt_cat=models.TextField()
   def __str__(self):
      return self.cat_name

class Item(models.Model):
   name=models.CharField(max_length=255)
   description=models.TextField()
   author=models.CharField(max_length=255)
   publisher=models.CharField(max_length=255)
   pub_date=models.CharField(max_length=10)
   price=models.CharField(max_length=255)
   edition=models.CharField(max_length=100)
   b_img=models.ImageField(upload_to='items/')
   category=models.ForeignKey(Category, on_delete=models.CASCADE)
   isbn = models.CharField(max_length=13, unique=True, null=True,
blank=True)# ISBN is typically 13 characters
   page_no = models.IntegerField(default=1)
```

```
    language = models.CharField(max_length=50, default="English")  #
Change the default as needed
    binding = models.CharField(max_length=50, default="Paperback")


class CartItem(models.Model):
    item = models.ForeignKey(Item, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=0)
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    date_added = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f'{self.quantity} x {self.item.name}'


class Order(models.Model):
    item = models.ForeignKey(Item, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=0)
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    date_ordered = models.DateTimeField(auto_now_add=True)
    payment_status=models.CharField(max_length=255)
    payment_id=models.CharField(max_length=255)
    address=models.TextField()
```

- **Views.py:**

```
import razorpay
from django.conf import settings
from django.http import JsonResponse
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth import authenticate, login, logout
from . forms import MyRegFrm, StdRegFrm, MyChngFrm
from . models import Student, Category, Item, CartItem, Order
from django.http import HttpResponse
```

```
# Create your views here.
def home(request):
    categories = Category.objects.all().order_by('cat_name')
    allItem=Item.objects.all()
    return render(request, 'myapp/home.html', {'allItem':allItem,
'categories':categories})

def about(request):
    categories = Category.objects.all().order_by('cat_name')
    allItem=Item.objects.all()
    return render(request, 'myapp/about.html', {'allItem':allItem,
'categories':categories})

def useraccount(request):
    categories = Category.objects.all().order_by('cat_name')
    allItem=Item.objects.all()
    if request.user.is_authenticated:
        if request.POST:
            form = MyChngFrm(request.POST, instance=request.user)
            if form.is_valid():
                try:
                    form.save()
                    messages.success(request,'Profile Update
successfully')
                except Exception as e:
                    messages.error(request, 'Profile Could Not Update
successfully')
        else:
            form=MyChngFrm(instance=request.user)
        return render(request, 'myapp/useraccount.html',
{'form':form, 'allItem':allItem, 'categories':categories})
    else:
        return redirect('/login')

def signIn(request):
    if request.user.is_authenticated:
        return redirect('/useraccount')
    else:
        if request.POST:
          form=AuthenticationForm(request=request,
data=request.POST)
          if form.is_valid():
             uname=form.cleaned_data['username']
             upass=form.cleaned_data['password']
             user=authenticate(username=uname, password=upass)
             if user is not None:
                login(request, user)
                return redirect('/useraccount')
       else:
          form=AuthenticationForm()
       context={"form":form}
       return render(request, 'myapp/login.html', context)

def signOut(request):
   logout(request)
   return redirect('/login')

def signup(request):
   if request.POST:
      form=MyRegFrm(data=request.POST)
      if form.is_valid:
         try:
            form.save()
            messages.success(request, 'Your registration is
succesfull')
         except Exception as e:
            messages.error(request, 'Your registration is not
successfull')
   else:
      form=MyRegFrm()
   context={'frm':form}
   return render(request, 'myapp/signup.html', context)

def forgotpassword(request):
   return render(request, 'myapp/forgotpassword.html')

def orders(request):
   return render(request, 'myapp/orders.html')
```

```
def shoppingcart(request):
    return render(request, 'myapp/shoppingcart.html')

def featured(request):
    return render(request, 'myapp/featured.html')

def arrivals(request, id):
    categories = Category.objects.all().order_by('cat_name')
    allItem=Item.objects.filter(category=id)
    return render(request, 'myapp/arrivals.html',
{'allItem':allItem, 'categories':categories})

def stdReg(request):
    if request.POST:
        form=StdRegFrm(data=request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, 'Registartion is Successfull')
        else:
            messages.error(request, 'Registration is Unsuccessfull')
    else:
         form=StdRegFrm()
    context={'form':form}
    return render(request, 'myapp/stdReg.html', context)

def stdList(request):
    allstd=Student.objects.all()
    return render(request, 'myapp/stdShow.html', {'allstd':allstd})

def stdDel(request, id):
    std=Student.objects.get(id=id)
    if std:
        std.delete()
        messages.success(request, 'Student Details Remove
Successfully')
    else:
        messages.error(request, 'Student Details Not Remove
Successfully')
    return redirect('/stdlist')
```

```
def stdUpd(request,id):
   std=Student.objects.get(id=id)
   form=StdRegFrm(request.POST or None, instance=std)
   if form.is_valid():
      try:
         form.save()
         messages.success(request, 'Student Details Updated
Successfully')
      except Exception as e:
         messages.error(request, 'Student Details Not Updated
Successfully')
      return redirect('/stdlist')
   return render(request, 'myapp/stdUpd.html', {'form':form})

def stdAdmin(request):
   allfeatured=Category.objects.all()
   return render(request, 'myapp/stdAdmin.html',
{'allfeatured':allfeatured})

def featuresbook1(request,item_id):
   item=Item.objects.get(id=item_id)
   return render(request, 'myapp/featuresbook1.html',
{'item':item})

def cart(request):
   return render(request, 'myapp/cart.html')

def add_to_cart(request, item_id):
   if request.user.is_authenticated:
      item = Item.objects.get(id=item_id)
      cart_item, created =
CartItem.objects.get_or_create(item=item,
                                          user=request.user)
      cart_item.quantity += 1
      cart_item.save()
      return redirect('/cart')
   else:
      return redirect('/login')

def view_cart(request):
    if request.user.is_authenticated:
        cart_items = CartItem.objects.filter(user=request.user)
        total_price = sum(int(item.item.price) * int(item.quantity)
for item in cart_items)
        total_price=int(total_price)
        return render(request, 'myapp/cart.html', {'cart_items':
cart_items, 'total_price': total_price})
    else:
        return redirect('/login')



def remove_cart(request,id):
    if request.user.is_authenticated:
        cart_item = CartItem.objects.get(id=id, user=request.user)
        cart_item.delete()
        return redirect('/cart')
    else:
        return redirect('/login')

# def initiate_payment(request):
#     return HttpResponse('Hi')

def initiate_payment(request):
    if request.method == "POST":
        amount = int(request.POST["amount"]) * 100  # Amount in
paise
        address=request.POST['address']
        client = razorpay.Client(auth=(settings.RAZORPAY_API_KEY,
settings.RAZORPAY_API_SECRET))

        payment_data = {
            "amount": amount,
            "currency": "INR",
            "receipt": "order_receipt",
            "notes": {
                "email": "user_email@example.com",
            },
        }

        order = client.order.create(data=payment_data)
        # Include key, name, description, and image in the JSON
response
        response_data = {
            "id": order["id"],
            "amount": order["amount"],
            "currency": order["currency"],
            "key": settings.RAZORPAY_API_KEY,
            "name": "My Project",
            "description": "Payment for Your Product",
            "image": "https://yourwebsite.com/logo.png",  # Replace
with your logo URL
        }
        cart_items=CartItem.objects.filter(user=request.user)
        # payment_id=response_data.id
        for cart in cart_items:
            Order.objects.get_or_create(user=request.user, item=
cart.item, quantity=cart.quantity, payment_status='success',
address=address)

        CartItem.objects.filter(user=request.user).delete()

        return JsonResponse(response_data)
    return redirect('myapp:viewCart.html')

def payment_success(request):
    return render(request, "myapp/payment_success.html")

def payment_failed(request):
    return render(request, "myapp/payment_failed.html")

def myOrders(request):
    if request.user.is_authenticated:
        allord=Order.objects.filter(user=request.user)
        return render(request,
'myapp/viewOrders.html',{'orderItems':allord})
    else:
        return redirect('/login')
```

- **Urls.py:**

```
from django.urls import path
from . import views

urlpatterns=[
   path('', views.home, name='hmpage'),
   path('about', views.about, name='aboutpage'),
   path('useraccount', views.useraccount, name='useraccountpage'),
   path('login', views.signIn, name='loginpage'),
   path('signup', views.signup, name='signuppage'),
   path('forgotpassword', views.forgotpassword,
name='forgotpasswordpage'),
   path('orders', views.orders, name='orderspage'),
   path('shoppingcart', views.shoppingcart, name='shoppingcartpage'),
   path('featured', views.featured, name='featuredpage'),
   path('arrivals/<int:id>/', views.arrivals, name='arrivalspage'),
   path('logout', views.signOut, name='logout'),
   path('stdreg', views.stdReg, name='std-reg'),
   path('stdlist', views.stdList, name='std-list'),
   path('stddel/<int:id>', views.stdDel, name='std-del'),
   path('stdupd/<int:id>', views.stdUpd, name='std-upd'),
   path('stdAdmin', views.stdAdmin, name='std-Admin'),
   path('featuresbook1/<int:item_id>/', views.featuresbook1,
name='featuresbook1page'),
   path('add/<int:item_id>/', views.add_to_cart, name='add_to_cart'),
   path('cart', views.view_cart, name='view_cart'),
   path('remove/<int:id>', views.remove_cart, name='remove_cart'),
   path("initiate-payment/", views.initiate_payment,
name="initiate_payment"),
   path("payment-success/", views.payment_success,
name="payment_success"),
   path("payment-failed/", views.payment_failed,
name="payment_failed"),
   path("myorder/", views.myOrders, name="my-orders"),
]
```

17. Testing

Team Interaction

The following describes the level of team interaction necessary to have a successful product.

- The Test Team will work closely with the Development Team to achieve a high quality design and user interface specifications based on customer requirements. The Test Team is responsible for visualizing test cases and raising quality issues and concerns during meetings to address issues early enough in the development cycle.

- The Test Team will work closely with Development Team to determine whether or not the application meets standards for completeness. If an area is not acceptable for testing, the code complete date will be pushed out, giving the developers additional time to stabilize the area.

- Since the application interacts with a back-end system component, the Test Team will need to include a plan for integration testing. Integration testing must be executed successfully prior to system testing.

Test Objective

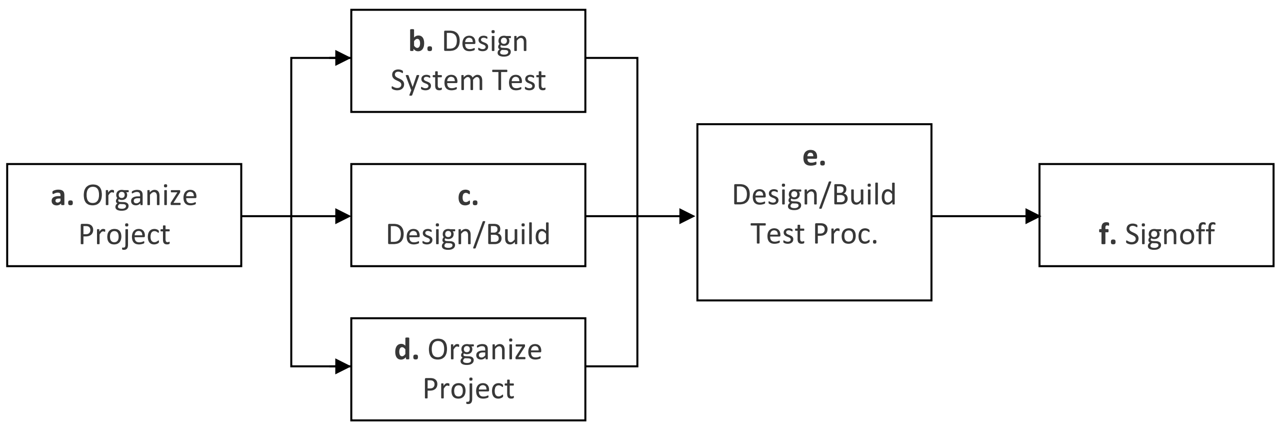
The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We will be testing a Binary Search Tree Application utilizing a pre-order traversal format. There will be eight key functions used to manage our application: load, store, clear, search, insert, delete, list in ascending order, and list in descending order. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

Testing Process



The diagram above outlines the Test Process approach that will be followed.

a. **Organize Project** involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.

b. **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.

c. **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.

d. **Build Test Environment** includes requesting/building hardware, software and data set-ups.

e. **Execute System Tests –** The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

f. **Signoff** - Signoff happens when all pre-defined exit criteria have been achieved.

Testing Strategy

The following outlines the types of testing that will be done for unit, integration, and system testing. While it includes what will be tested, the specific use cases that determine how the testing is done will be detailed in the Test Design Document. The test cases that will be used for designing use cases is shown in Figure 2.1 and onwards.

Test Cases

Tested By:	Subhojit Chakraborty	
Test Type	Unit Testing	
Test Case Number	1	
Test Case Name	User Identification	
Test Case Description	The user should enter his/ her accurate userid and password so that he/she can able to go for the further options. The test case will check the application for the same since a user can only login with the correct userid , password.	
Item(s) to be tested		
1	Verification of the userid and password with the record in the database.	
Specifications		
Input	**Expected Output/Result**	
1) Correct User id and password 2) Incorrect Id or Password	1) Successful login 2) Failure Message	

Tested By:	Debansu Ghosh
Test Type	Unit Testing
Test Case Number	2
Test Case Name	Order Process
Test Case Description	Verify that the order process functions correctly by ensuring the user follows all steps: login, selecting a product, adding it to the cart, entering the address, completing the payment, and successfully placing the order.

Item(s) to be tested	
1	Check whether the user id logged in.
2	Order Processes was Performing Perfectly or not

Specifications

Input	**Expected Output/Result**
1) Attempting to add a product to the cart or proceed with order directly without being logged in.	1) Redirect the user to the login page.
2) Logged in, but no product added to the cart.	2) In cart option it will show empty.
3) Logged in, product added to cart, but no address provided.	3) Display an error message asking the user to input an address.
4) Logged in, product in cart, address entered, but payment unsuccessful.	4) Show an error message for payment failure.
5) Logged in, product in cart, address entered, payment successful.	5) Display the product ordered in order section.

Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.
Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing

18. System Security measures (Implementation of security for the project developed)

- Only authorized users are allowed.
- Without signing in users are not allowed to go an intermediate page by typing an URL. For all such efforts, users will be redirected to the home page.

19. Database/Data security

- Database is present in remote machine.
- MySQL default securities are applied.

20. Creation of User profiles and access rights

- The admin must create users manually
- The admin can create more admins

21. Cost Estimation of the Project along with Cost Estimation Model

Analogous estimate of effort or cost

Used for Early Estimate or Individual Activity Estimate
Sample example shown below is for two major deliverables of a software project. You use a previous project as a benchmark for analogous estimation. Using your experience you will estimate a multiplier.

Multipliers:

1. Prototyping: 0.75.
2. Testing: 0.5
3. Deployment: 0.5

Finally, if you want to convert to cost, you would use current rates for the resource.

WBS ID	Previous Similar Project Activity	Previous Effort	Current Project Estimate	Multiplier	Effort (Previous Effort * 0.75)	Cost (Rs. 500/hr.)
1	Prototyping	40 Work-Hours	Prototyping	0.75	30 Work-hours	Rs. 15000/-
2	Testing	20 Work-Hours	Testing	0.50	10 Work-Hours	Rs. 5000/-
Total					**40 Work-Hours**	**Rs. 20000/-**

Note: Effort is also called Size and unit of estimation is called either Work-Hour, person-hours.

22. Future scope and further enhancement of the Project

DEBSU LIT could evolve into a captivating platform that seamlessly categorizes books for an enhanced browsing experience, allowing users to explore by genres, authors, or publication dates. Interactive elements such as virtual book clubs, engaging discussion forums, and live Q&A sessions with authors would foster a thriving community of readers. Authors could upload exclusive content, including video tutorials or live readings, to strengthen connections with their audience. Features like secure email verification during sign-up would ensure account authenticity, while integrated reviews and ratings would help users share insights and discover popular works. A predictive search bar would offer easy exploration, while customizable user profiles with options to add and update profile pictures would provide a personalized touch. Together, these elements would transform DEBSU LIT into a dynamic and immersive hub for literature enthusiasts to connect, share, and grow in FUTURE.

23. Bibliography

1. Roger S. Pressman. Software Engineering: A Practioner's Approach (Sixth Edition, International Edition). McGraw-Hill, 2005.
2. Ian Sommerville. Software Engineering (Seventh Edition). Addison-Wesley, 2004.
3. Frederick P. Brooks. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition. Addison-Wesley Pub Co; 1st edition (August 2, 1995).
4. Introducing HTML5 by Bruce Lawson, Remy Sharp.
5. HTML5 for Web Designer by Jeremy Keith.
6. HTML for World Wide Web (Visual Quick Start Guide) by Elizabeth Castro.
7. HTML5 Up And Running by Mark Pilgrim.
8. The Definitive Guide To HTML5 by Adam Freeman.
9. Pro HTML5 Programming: Powerful APIs for Rich Internet Application Development by Peter Lubbers.
10. Dynamic HTML: The Definitive Reference by Danny Goodman.
11. CSS: Definitive Guide by Eric A. Meyer.
12. CSS Cook Book by Christopher Schmitt.
13. CSS: Missing Manual by David Sawyer McFarland.
14. CSS Mastery: Advanced Web Standard Solution by Andy Budd, Cameron Moll, Simon Collison.
15. CSS Anthology by Rachel Andrew.

16. Handcrafted CSS: More Bulletproof Web Design/Bulletproof Essentials by Dan Cederholm, Ethan Marcotte.
17. Programming Python by Mark Lutz.
18. Python Cookbook by David Beazley and Brian K. Jones.
19. Python Security by Himanshu Sharma.
20. Professional Python Programming by Luke Sneeringer.
21. Python Object-Oriented Programming by Steven F. Lott.
22. Database System Concepts by Abraham Silberschatz, Henry F. Korth, S. Sudarshan.