Software Design Description

Ghadeer Mokhtar, Reham Adel, Mariam Ashraf, Ismail Nasser
Supervised by: Dr. Mohamed Shalbey, Eng. TA Ahmed Talaat

December 30, 2024

Table 1: Document version history

Version	Date	Reason for Change
1.0		SDD first version's description are defined.
1.1		Added Sequence Diagram.
1.3		Requirement Matrix updated.

GitHub: https://github.com/mariammhassann/Smart-Calendar-28-.git

Contents

Abstract

Calendok is a web-based platform designed to streamline personal scheduling, task management, and planner purchasing for users with busy lives. This application enables users to efficiently organize their daily, weekly, and monthly schedules while managing tasks and receiving timely reminders for upcoming events. Additionally, Calendok features an e-commerce component that allows users to purchase physical planners, enhancing their organizational tools.

For administrators, the platform offers a comprehensive dashboard for monitoring user schedules and managing product listings, ensuring a coordinated environment for task management and sales. This Software Design Description (SDD) details the architecture, functional and non-functional requirements, user interface design, and data management strategies necessary for building a reliable and user-friendly application. By focusing on productivity and effective task management, Calendok significantly enhances personal time management, providing users with both digital and physical tools to improve their organizational efficiency.

1 Introduction

1.1 Purpose

This Software Design Description (SDD) document outlines the design framework for Calendok, a web-based scheduling and task management application. It serves as a reference for developers, stakeholders, and project managers throughout the development life cycle.

1.2 Scope

The document details the design architecture, and components necessary to implement Calendok, focusing on user account management, scheduling, task tracking, notifications, and e-commerce functionality for planner purchases.

1.3 Overview

Calendok provides an integrated platform for managing personal schedules and tasks while allowing users to buy physical planners. This document is structured to cover system architecture, design constraints, user interfaces, and data design.

1.4 Intended audience

Project Stakeholders: Individuals or groups with an interest in the project, including sponsors, managers, and potential users.
Development Team: Software engineers, developers, and designers responsible for the implementation of the application.
Quality Assurance Team: Testers who will validate the functionality and performance of the application.
System Administrators: Personnel responsible for maintaining the application and managing user accounts.

Business Analysts: Professionals who bridge the gap between stakeholders and the technical team, ensuring requirements are met.
Documentation Specialists: Individuals responsible for creating user manuals and training materials based on the SDD.

1.5 Reference Material

software requirements specifications (SRS),SWE2_SDD_DOC.

1.6 Definitions and Acronyms

Term	Definition
Calendok	The name of the web-based scheduling and task management application described in this SDD.
API	Application Programming Interface. A set of rules and protocols for building and interacting with software applications.
E-commerce	Electronic commerce. The buying and selling of goods and services over the internet.
UI	User Interface. The space where interactions between users and the application occur, including buttons, menus, and other elements.
UX	User Experience. The overall experience of a person using the application, focusing on usability and satisfaction.
SRS	Software Requirements Specification. A document that captures the functional and non-functional requirements of the software.
UAT	User Acceptance Testing. A phase in software development where end-users test the application to ensure it meets their needs.
Database	An organized collection of data that can be easily accessed, managed, and updated.
Notification System	A component of the application that sends alerts and reminders to users about tasks and events.
Task Management	The process of managing a task through its lifecycle, including planning, execution, and tracking.
User Management	The administration of user accounts, including registration, login, and profile management.
Frontend	The part of the application that users interact with directly, typically developed using HTML, CSS,php and JavaScript.
Backend	The server-side of the application that processes requests, manages the database, and handles business logic.
Scalability	The capability of the application to grow and manage increased demand without compromising performance.

2 System Overview

The system described in this document is an e-Commerce and scheduling platform named Calendok, designed to help users manage their schedules, tasks, and purchase physical planners.
Key Features:

- **Schedule Management**: Users can create, edit, and organize daily, weekly, and monthly schedules.
- **Task Management**: Users can add, prioritize, and track tasks with reminders.
- **E-Commerce**: Customers can browse and purchase physical planners directly from the platform.
- **Secure Payment**: The system supports secure payment methods such as credit cards and PayPal for planner purchases.
- **Order Tracking**: Users can track the status of their planner orders from purchase to delivery.
- **Admin Dashboard**: Admins can manage users, product listings, and orders.

Target Users:

- **Customers**: Individuals looking to manage their schedules and purchase planners.
- **Admins**: Individuals responsible for managing product listings, user accounts, and order processing.
- **Owners of local brand**s: Partners who provide planners and manage their inventory and listings.

2.1 System Scope

Calendok is designed to serve as a comprehensive platform for personal scheduling, task management, and planner purchasing. It aims to address the needs of people with busy, multifaceted schedules by providing a user-friendly interface to organize daily, weekly, and monthly activities. The system integrates e-commerce functionality to allow users to purchase physical planners, bridging the gap between digital and traditional planning tools. Additionally, an admin dashboard facilitates the efficient management of user activities, product listings, and orders, ensuring a cohesive operational environment.

2.2 System objectives

The main objectives of Calendok are:

- To enable users to manage their schedules effectively with features like task prioritization and reminders.
- Provide an integrated e-commerce solution for purchasing physical planners, ensuring a seamless shopping experience.
- To achieve a System Usability Scale (SUS) score of 80 or higher, reflecting high user satisfaction.
- To ensure robust performance, with a target response time of less than 1 second for core functionalities under normal load.
- Implement role-based access control (RBAC) for secure user-admin interactions.

2.3 System Timeline

In figure 1: Timeline

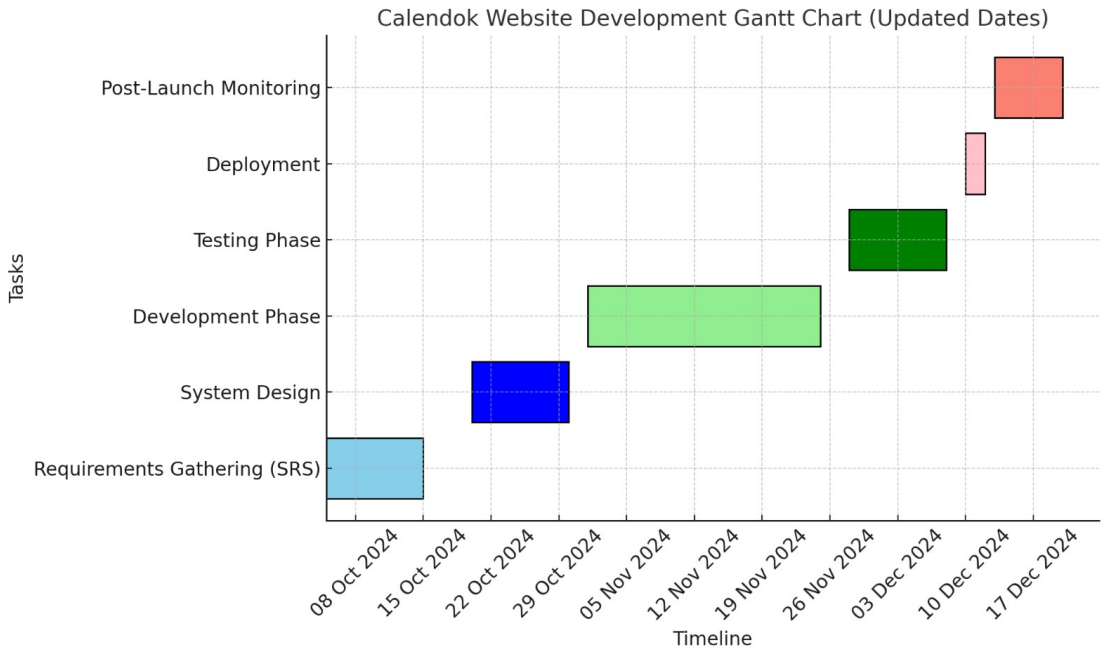


Figure 1: timeline

3 Design viewpoints

3.1 Context viewpoint

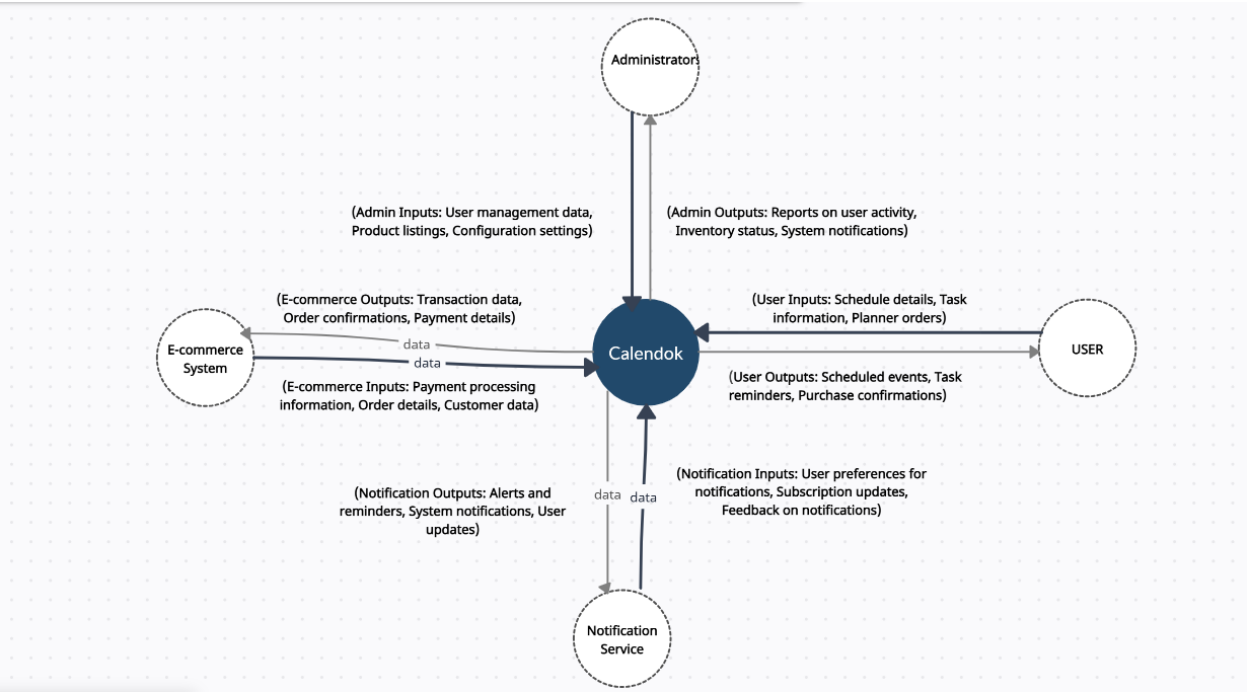


Figure 2: Context Digram

3.2 Composition viewpoint

3.2.1 Design Rationale

The architecture diagram illustrates the system's layered structure,
designed to ensure separation of concerns and modularity.

- Presentation Layer: This layer consists of two components—Web Application and Mobile Application—which serve as the interfaces for users to interact with the system. By separating the presentation from the business logic, the design allows for flexibility in user experience across different platforms.
- Business Logic Layer: Central to the architecture, this layer includes Controllers and Services. Controllers manage incoming requests, delegating tasks to the appropriate services. The Notification Service is specifically highlighted, showcasing its role in handling notifications within the system, ensuring that user engagement is maintained through timely updates.
- Data Access Layer: This layer abstracts the complexities of data handling through Repositories and Data Models. Repositories facilitate interaction with the database, while Data Models define the structure of the data being used, promoting a clear understanding of data relationships.
- Database Layer: At the foundation, the Relational Database is employed to store persistent data. This choice ensures data integrity and supports complex queries, making it suitable for the application's needs.
- External Services: The architecture also includes connections to Payment Gateway and Email Service, which are essential for processing transactions and managing communications. This modular approach allows for easier integration and scalability of external functionalities.

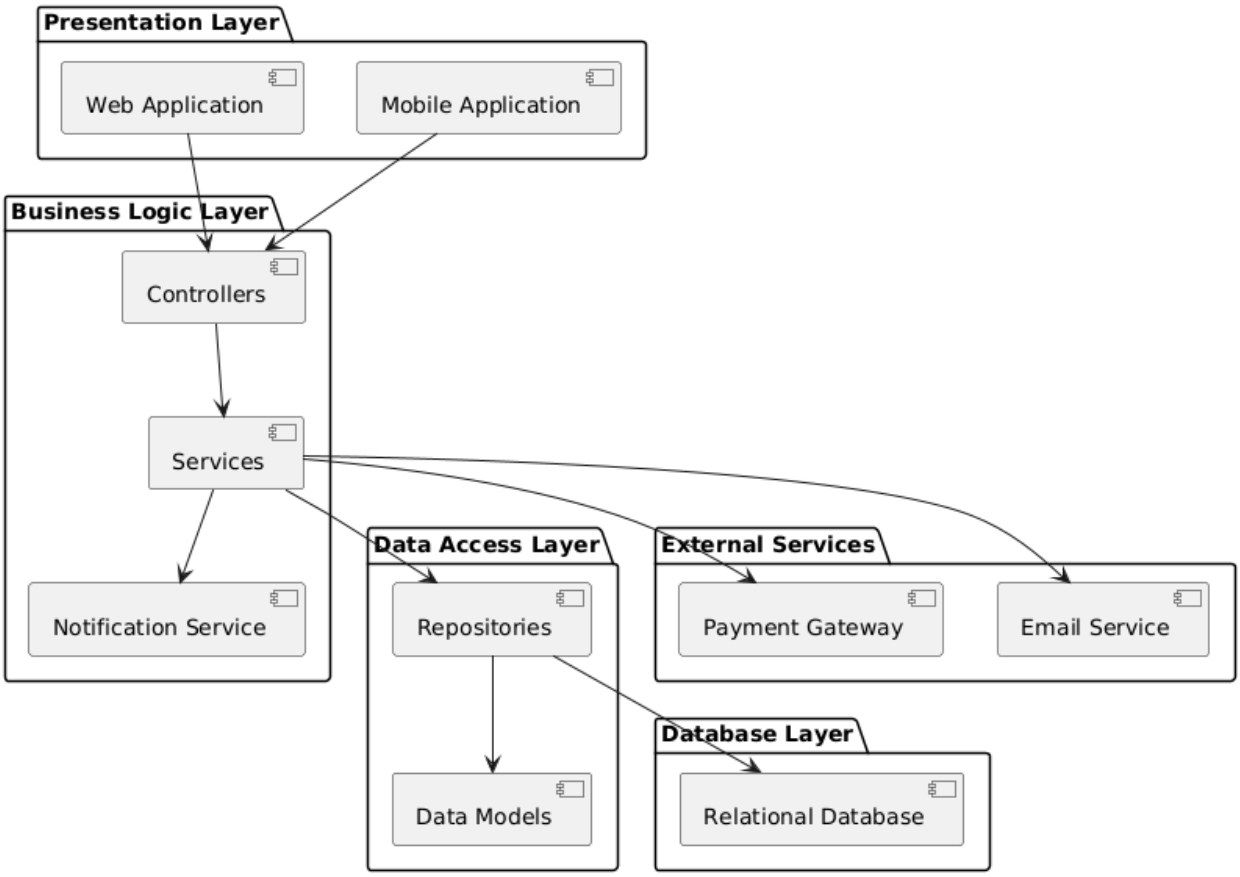


Figure 3: architecture diagram

3.3 Logical viewpoint

An Initial class diagram of the system is found in the figure

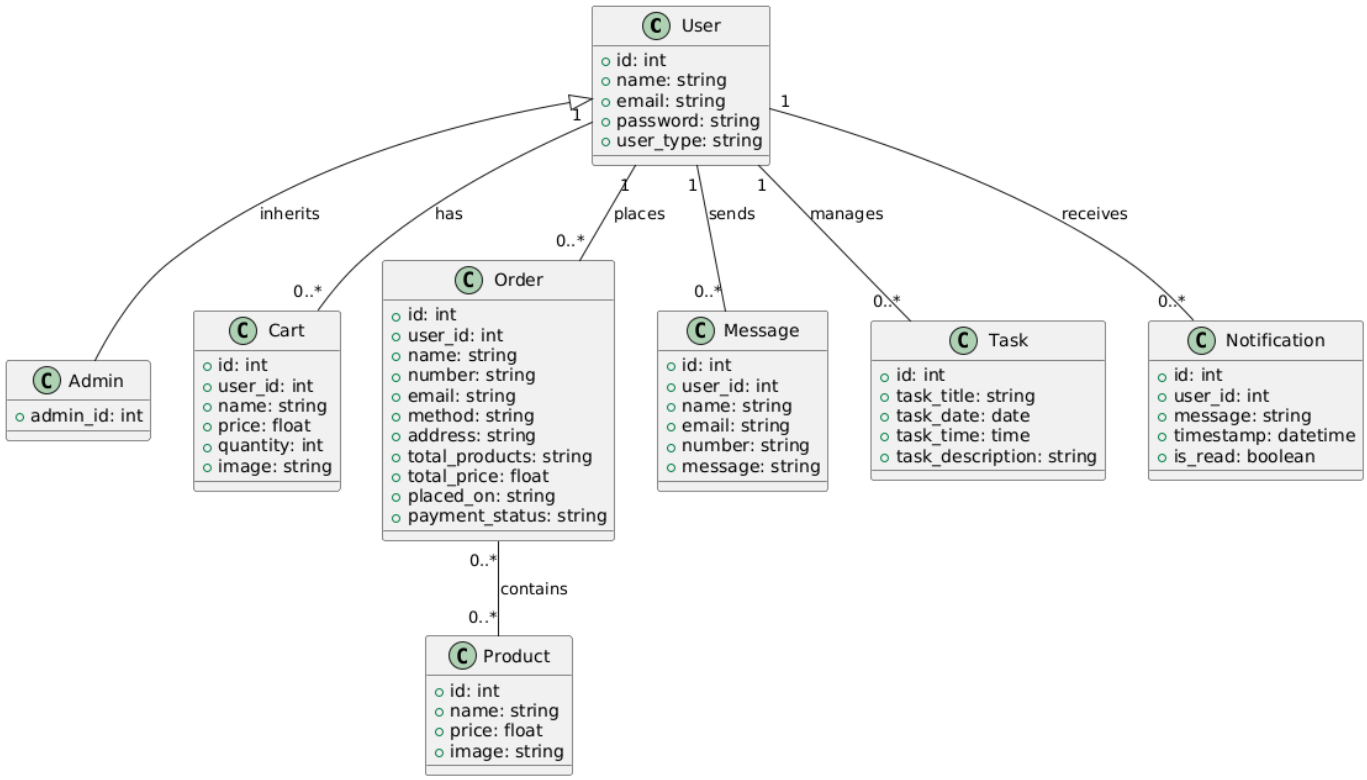


Figure 4: UML class diagram

Attribute	**Description**
Abstract or Concrete:	Concrete
Superclasses:	None
Subclasses:	Admin
Purpose:	Represents a user in the system, including both regular users and admins.
Collaborations:	none
Attributes:	• id: int • name: string • email: string • password: string • user_type: string
Operations:	• getUser(): user -> Retrieves the current user's information as a User object. • login(): void -> Authenticates the user with provided credentials and starts a user session. • logout(): void -> Ends the user's session and logs them out of the system. • updateProfile(): void -> Updates the user's profile information with the provided data.

Table 2: User Class

Attribute	Description
Abstract or Concrete:	Concrete
Superclasses:	User
Subclasses:	None
Purpose:	Represents an administrative user with elevated privileges.
Collaborations:	Collaborates with User class.
Attributes:	• admin_id: int
Operations:	• deleteUser(userId): void -> Removes a user from the system based on the provided user ID. • manageProducts(): void -> Facilitates the addition, modification, or removal of products in the system. • viewReports(): void -> displays reports related to system performance, user activity, or product sales.

Table 3: Admin Class

Attribute	Description
Abstract or Concrete:	Concrete
Superclasses:	None
Subclasses:	None
Purpose:	Represents a plan available in the shop.
Collaborations:	Collaborates with Order and Cart classes.
Attributes:	• id: int • name: string • price: float • image: string
Operations:	• updatePrice(newPrice): void -> Updates the price of a product to the specified new price. • getDetails(): String -> Retrieves a detailed description of the product as a string.

Table 4: Product Class

Attribute	Description
Abstract or Concrete:	Concrete
Superclasses:	None
Subclasses:	None
Purpose:	Represents a shopping cart for a user.
Collaborations:	Collaborates with User and Product classes.
Attributes:	• id: int • user_id: int • name: string • price: float • quantity: int • created_at: datetime
Operations:	• addProduct() ->Adds a product to the cart with a specified quantity. • removeProduct()-> Removes a product from the cart. • updateQuantity()->Updates the quantity of a specific product in the cart. • calculateTotal()->Calculates the total price of all products in the cart. • clearCart() -> Empties all products from the cart.

Table 5: Cart Class

Attribute	Description
Abstract or Concrete:	Concrete
Superclasses:	None
Subclasses:	None
Purpose:	Represents an plan order placed by a user.
Collaborations:	Collaborates with User and Product classes.
Attributes:	• order_Id: int • user_id: int • name: string • number: string • email: string • method: string • address: string • total_products: string • total_price: float • placed_on: string • payment_status: string
Operations:	• updateStatus(newStatus): void -> Updates the status of an order to the specified new status. • getOrderDetails(): String -> Retrieves and returns the details of the order as a string.

Table 6: Order Class

Attribute	**Description**
Abstract or Concrete:	Concrete
Superclasses:	None
Subclasses:	None
Purpose:	Represents a message sent by a user.
Collaborations:	Collaborates with User class.
Attributes:	• id: int • user_id: int • name: string • email: string • number: string • message: string
Operations:	• deleteMessage(): void ->Deletes the current message, removing it from the system. • getMessageDetails(): String ->Retrieves the details of the current message and returns them as a string.

Table 7: Message Class

Attribute	**Description**
Abstract or Concrete:	Concrete
Superclasses:	None
Subclasses:	None
Purpose:	Represents a task assigned to a user.
Collaborations:	Collaborates with User class.
Attributes:	• id: int • task_title: string • task_date: date • task_time: time • task_description: string
Operations:	• addEvent(event: Event) -> Adds a new event to the schedule using the provided event object. • removeEvent(eventId) -> Removes an event from the schedule based on the specified event ID.

Table 8: Task Class

Attribute	**Description**
Abstract or Concrete:	Concrete
Superclasses:	None
Subclasses:	None
Purpose:	Represents notifications sent to users regarding tasks, messages, or system updates.
Collaborations:	Collaborates with User and Task classes.
Attributes:	• id: int-> • user_id:int • message: string • timestamp :datetime • is_read :boolean
Operations:	• markAsRead() void ->Updates the is read status to true. • getNotification() void->Retrieves the notification details. • sendNotification()void ->Sends a notification to the specified user.

Table 9: notifacation Class

3.4 Patterns use viewpoint

The system follows the Model-View-Controller (MVC) pattern to separate the user interface, data management, and control logic. This structure improves maintainability and scalability by allowing independent updates to each part without affecting the others.

- **Model**: Manages data for tasks, schedules, and e-commerce products. All data operations interact with the model.
- **View**: Represents the user interface, showing calendars, tasks, and planners for purchase.
- **Controller**: Handles user actions such as creating tasks, updating schedules, and processing orders. Manages communication between the model and the view.

3.5 Algorithm viewpoint

3.6 Interaction viewpoint

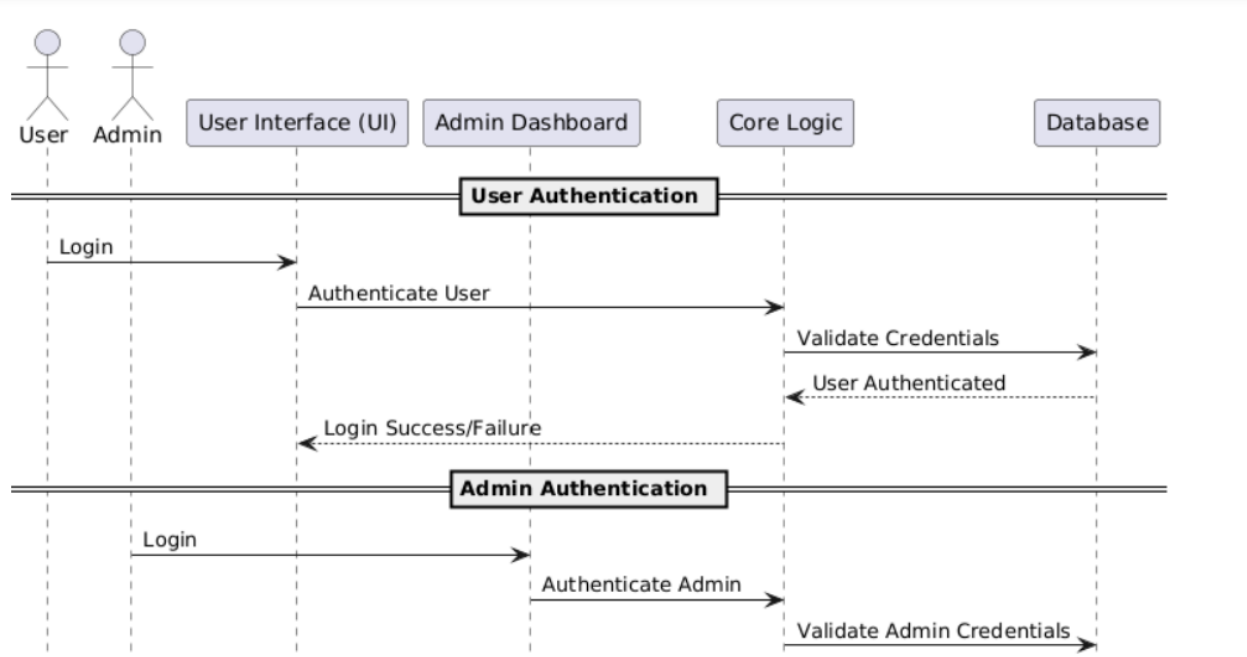


Figure 5: User and Admin Verification

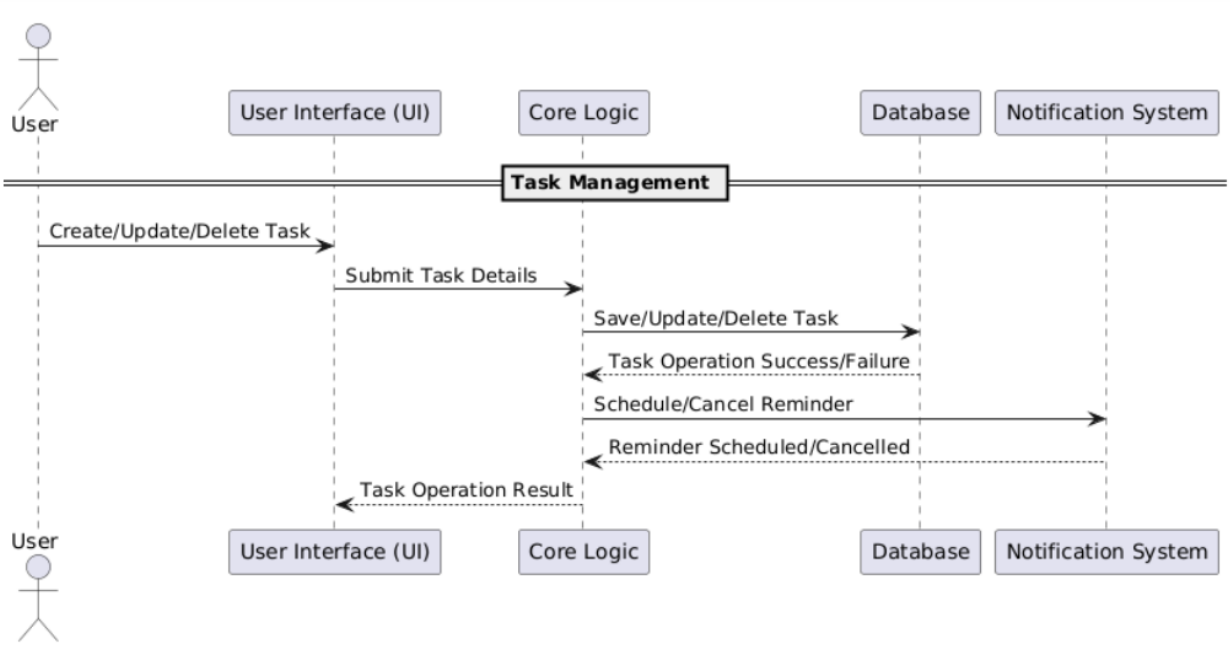


Figure 6: Task Management

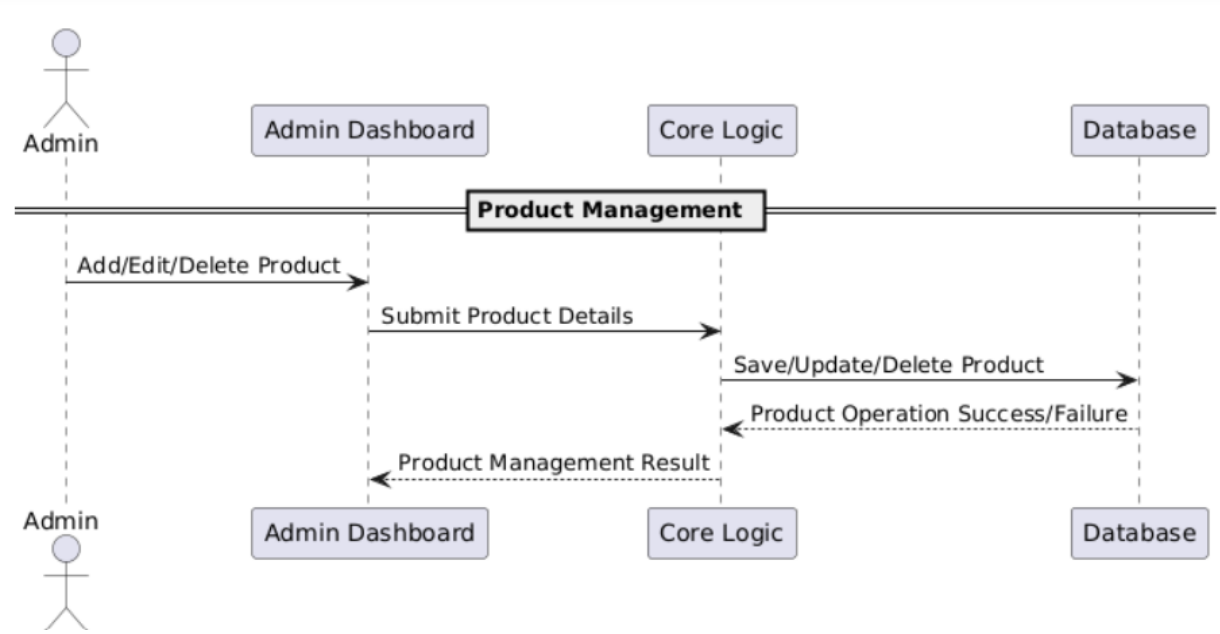


Figure 7: Product Management

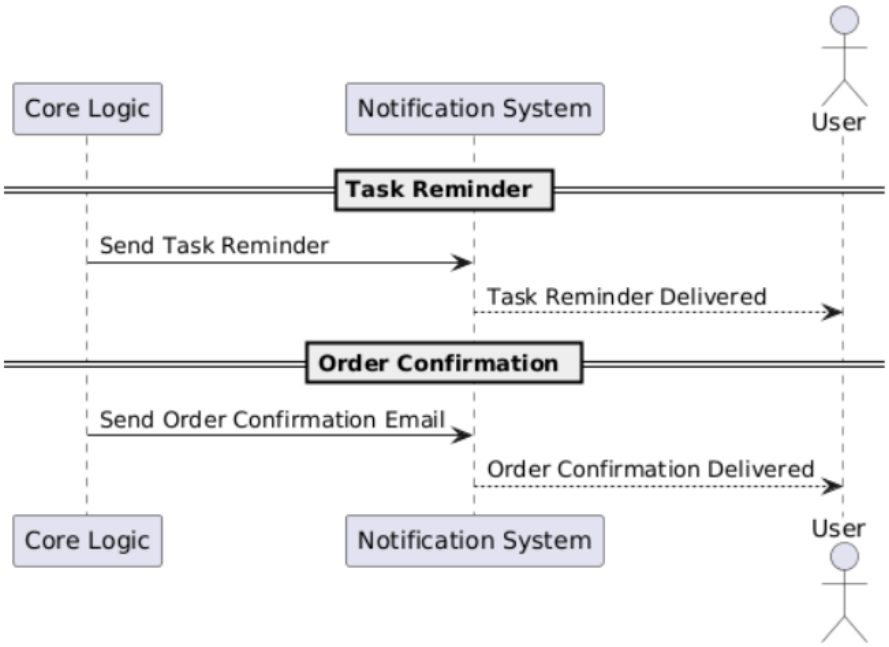


Figure 8: Notification

3.7 Interface viewpoint

The Calendok System Interface integrates key modules to deliver scheduling, task management, and e-Commerce functionality. The main components are the following.

- **User Interface**: Provides calendar views, task management, and a shopping platform for purchasing planners.
- **E-Commerce Module**: Supports secure payments via Payment Gateway API and order tracking through Logistics API.
- **Admin Dashboard**: Allows administrators to manage user schedules, product listings, and system operations.
- **Database System**: Stores user data, tasks, product inventory, and order history for seamless operation.

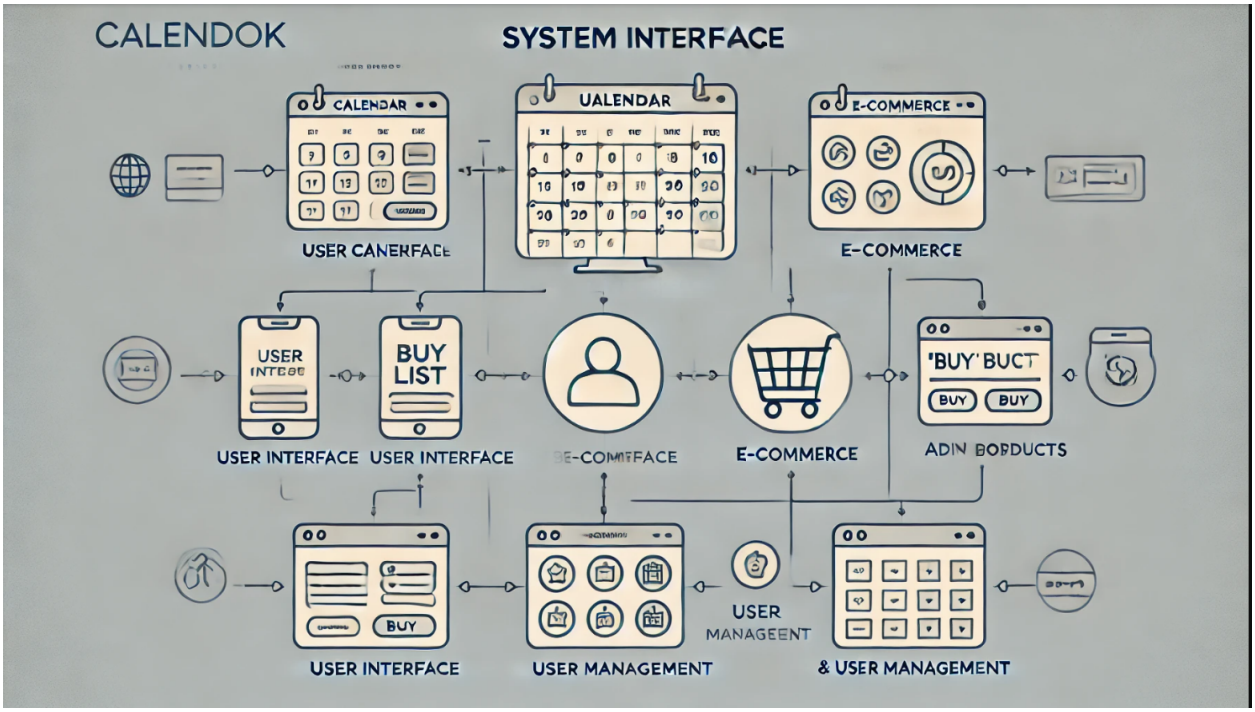


Figure 9: Interface viewpoint

4 Data Design

4.1 Data Description

The data design for the Smart Calendar system will focus on structuring the data needed for user management, task management, notifications, and calendar integrations. The design will ensure that data is organized, easily accessible, and secure.

4.2 Dataset Description

4.3 Database design description

Users Table to Tasks Table: One-to-Many relationship (a user can have multiple tasks).
Users Table to Notifications Table: One-to-Many relationship (a user can receive multiple notifications).

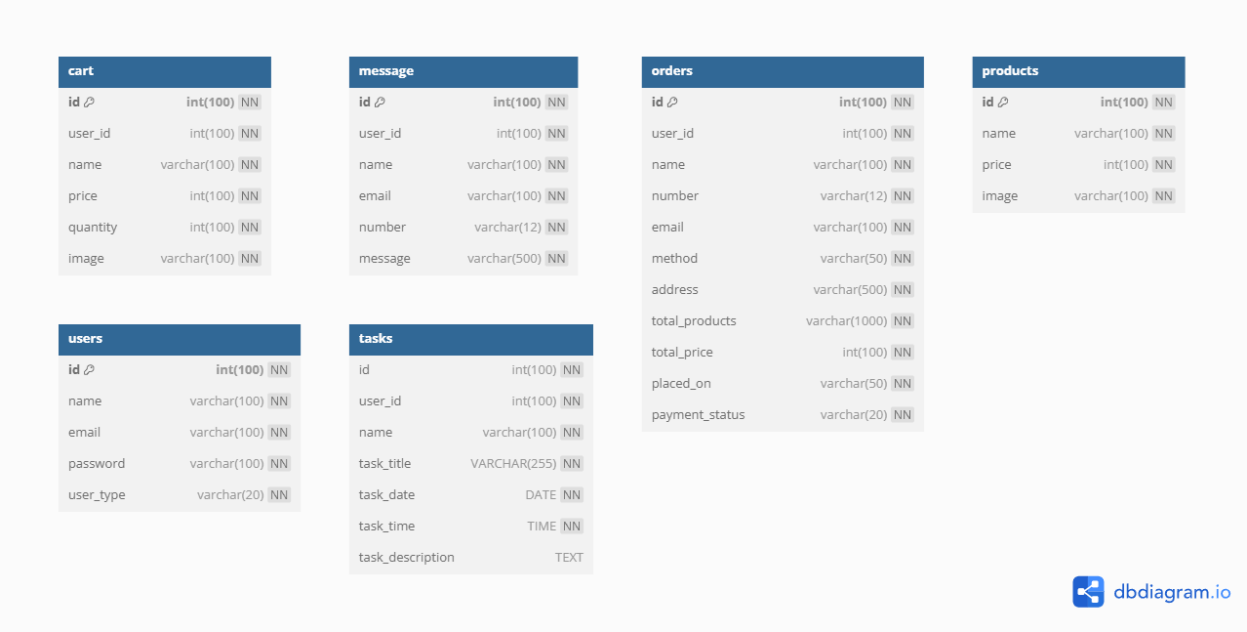


Figure 10: database

5 Human Interface Design

5.1 User Interface

5.1.1 5.1.1 User

Description: The user is an individual who interacts with the calendar system for personal or professional scheduling needs.

Features Accessible by the User:

- **View Calendar:**
  - Users can view their calendar in different formats (e.g., daily, weekly, monthly).
  - Color-coded event types for easier differentiation.
- **Add Events:**
  - Users can create events by specifying details such as title, date, time, and description.
- **Edit Events:**
  - Update or modify the details of existing events.
- **Delete Events:**
  - Remove events permanently.

- **Set Reminders:**
  - Notifications for upcoming events via push notifications.
- **Search Events:**
  - Search for specific events using date ranges.

Interface Elements:

- Navigation Bar: Includes links to different calendar views and settings.
- Interactive Calendar Grid: Displays days, weeks, or months with clickable time slots.
- Modal Dialogs: Used for creating, editing, or deleting events.

5.1.2 5.1.2 Admin

Description: The admin is responsible for managing user accounts, permissions, and system settings.

Features Accessible by the Admin:

- **User Management:**
  - View a list of registered users.
  - Add or remove user accounts.
  - Reset passwords and manage permissions.
- **Event Oversight:**
  - Access and edit events created by users if needed.
  - Monitor system-wide activity logs for compliance.
- **System Configuration:**
  - Customize global settings such as default time zones, notification templates, and calendar views.

Interface Elements:

- Dashboard: Overview of system metrics (e.g., number of users, total events).
- User Management Panel: List of users with actions like edit, or delete.
- Settings Page: Configuration options for the calendar system.
- Logs Viewer: Access to activity and error logs for debugging.

Accessibility Considerations:

- Simplified layout with intuitive navigation.
- Searching within user lists or logs.
- Role-based access controls to ensure data privacy.

5.2 Screen Images

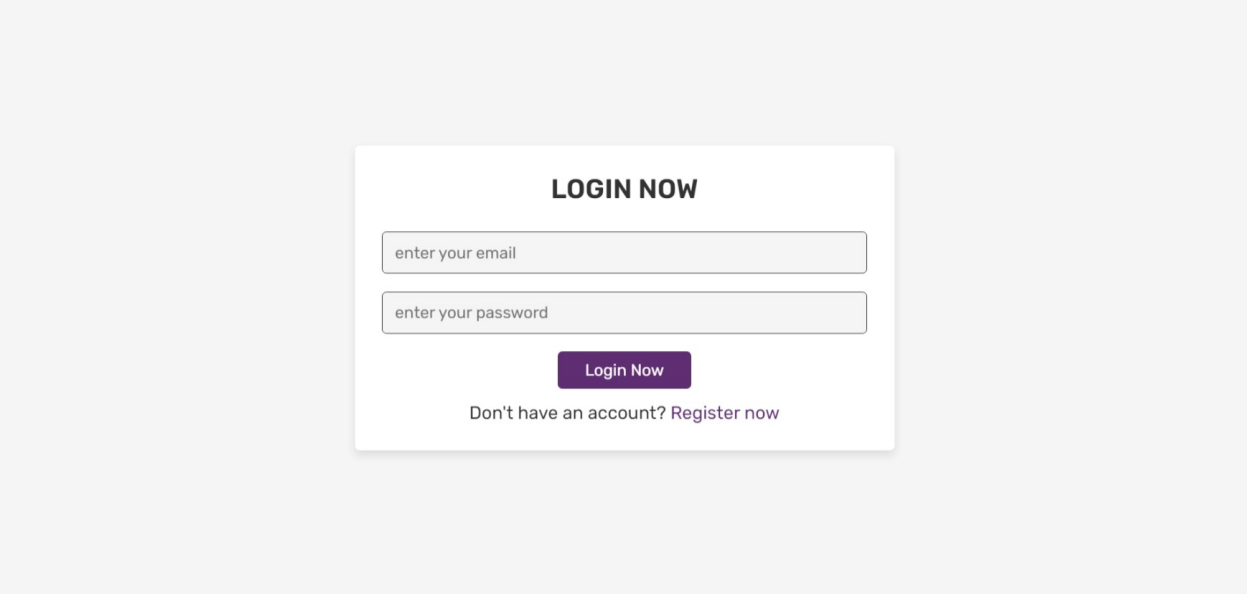


Figure 11: login page

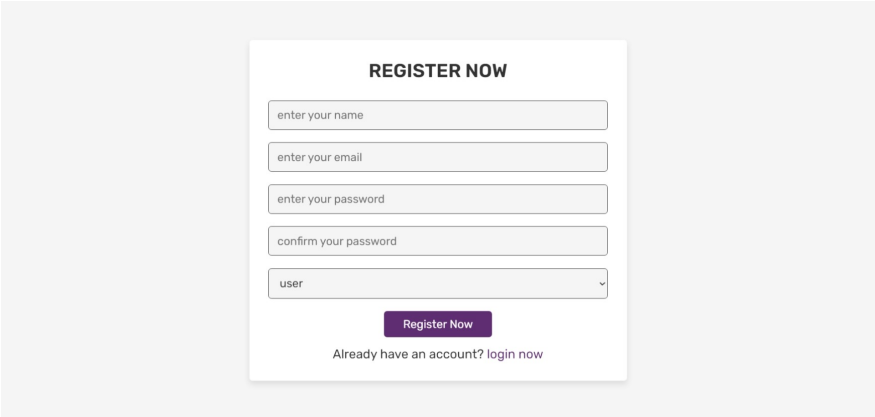


Figure 12: register page

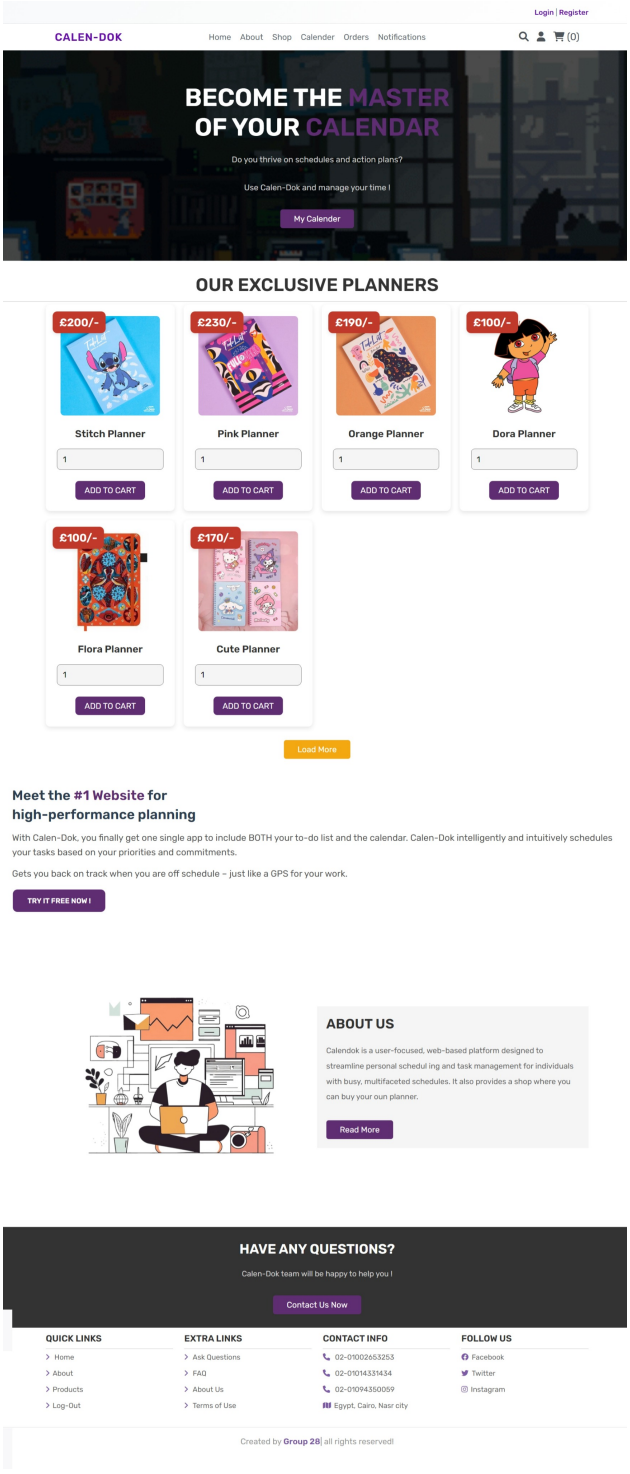


Figure 13: home page

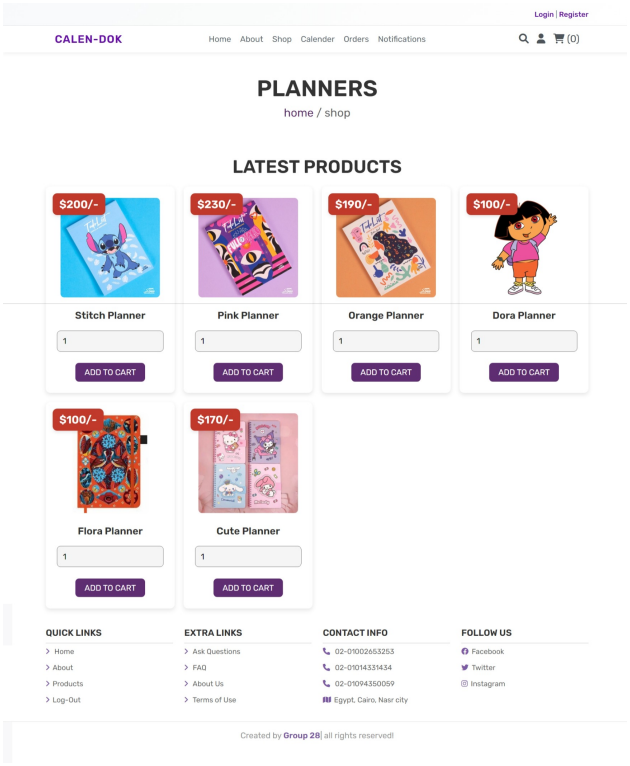


Figure 14: shop planners page

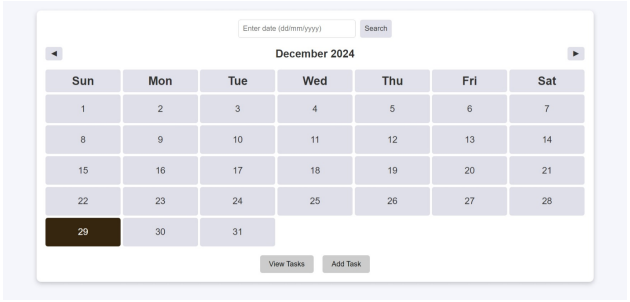


Figure 15: calendar page

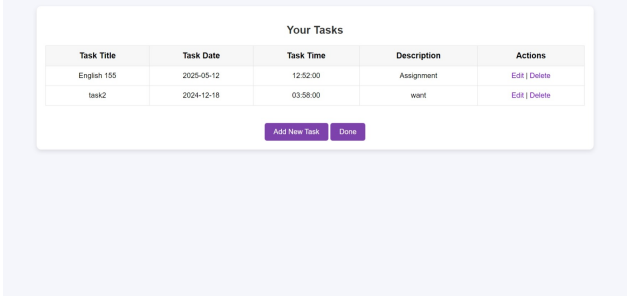


Figure 16: task page

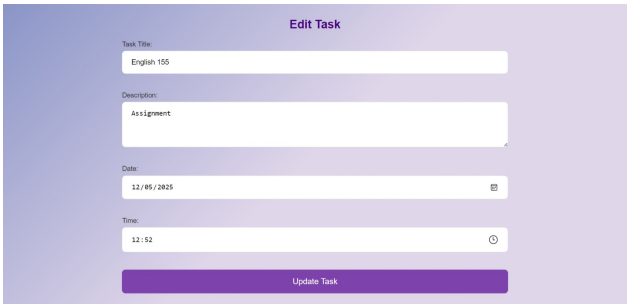


Figure 17: edit task page

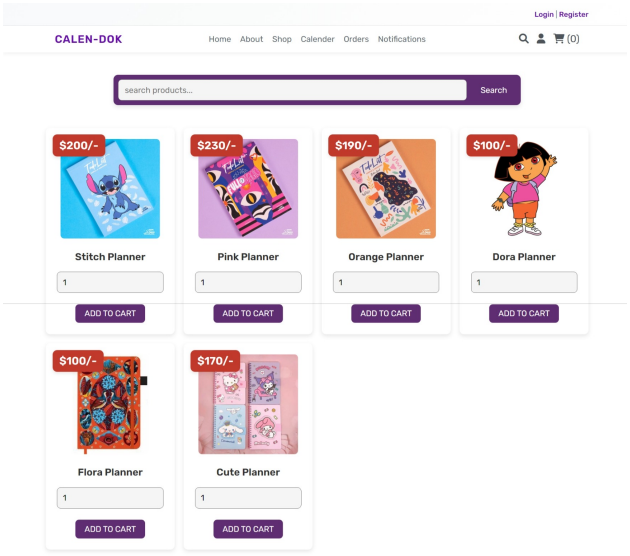


Figure 18: search page

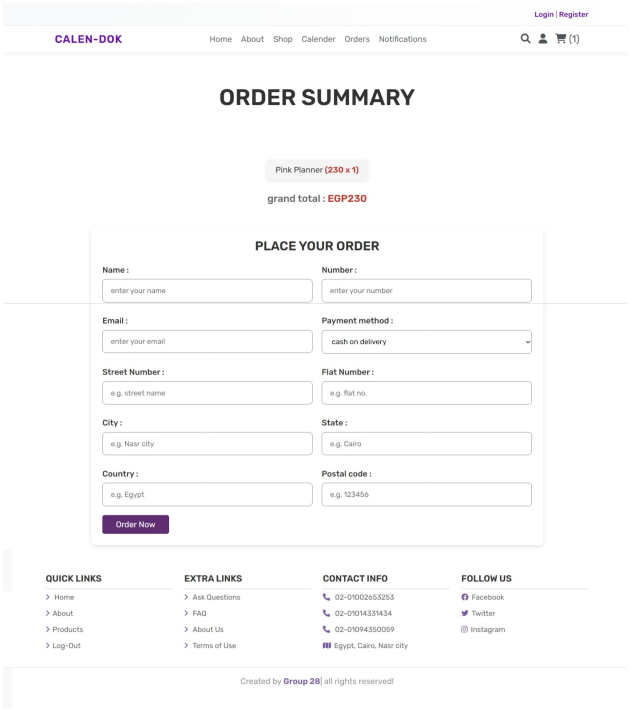


Figure 19: checkout page

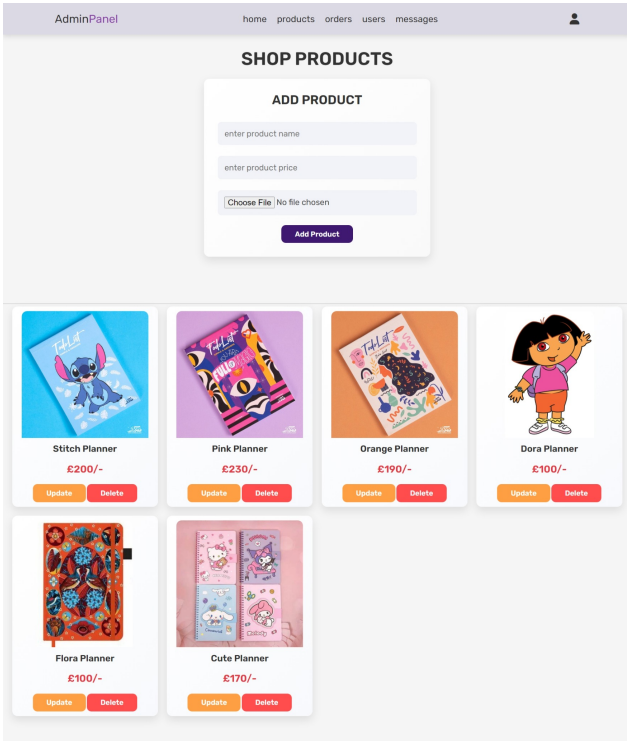


Figure 20: products admin page

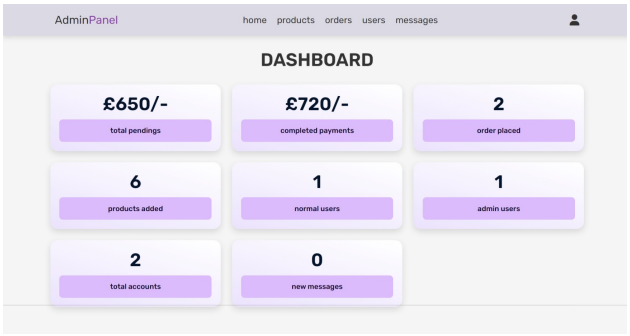


Figure 21: admin dashboard page

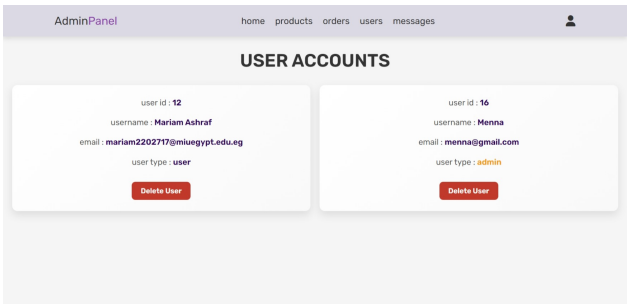


Figure 22: user page

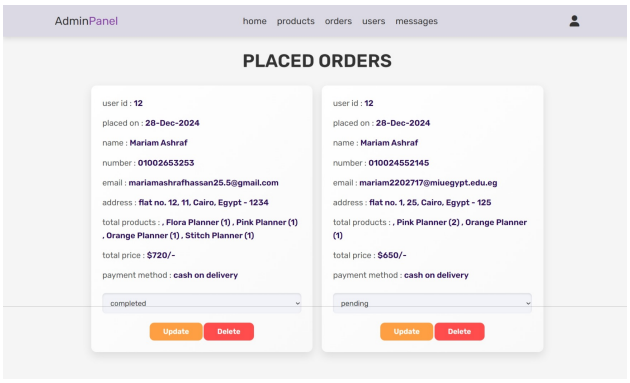


Figure 23: orders page

5.3 Screen Objects and Actions

- login screen
  Objects:
  Username Input Field, Password Input Field, Login Button, Forgot Password Link
  Actions:
  User enters their username and password, User clicks the Login Button.
  User clicks the Forgot Password Link to initiate password recovery.

- Dashboard calendar screen
  Objects:
  Calendar View (Monthly/Weekly/Daily), Add Task Button, List of Upcoming Tasks
  Notifications Panel
  Actions:
  User clicks on a date to view tasks for that day.
  User clicks the Add Task Button to create a new task.
  User views notifications by clicking on the Notifications Panel.

- Task Management Screen
  Objects:
  Task Title Input Field, Task Description Input Field, Due Date Selector
  Save Task Button, Cancel Button
  Actions:
  User enters task details and selects a due date.
  User clicks Save Task to store the task in the database.
  User clicks Cancel to discard changes and return to the previous
  screen.

- Notifications Screen
  Objects:
  List of Notifications, Mark as Read Button , Delete Notification Button
  Actions:
  User views notifications listed chronologically.
  User clicks Delete to remove notifications from the list.

- register Screen
  Objects:
  User Profile Information Fields (Name, Email, etc.)
  Password Fields, Save Changes Button,

- Logout
  Actions:
  User updates personal information and clicks Save Changes.

User changes their password and clicks Save Changes.
User clicks Logout to exit the application.

6 Requirements Matrix

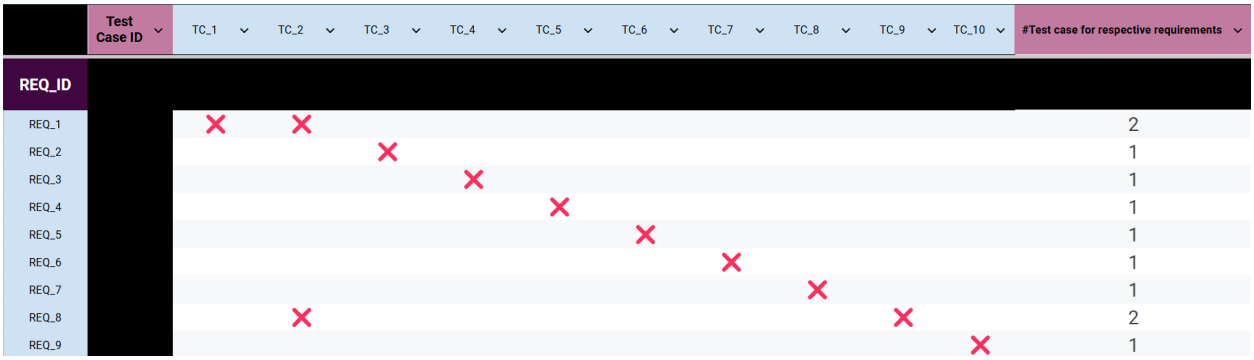
Test Case Mapping:

- **TC_1**: Register with valid inputs → Maps to **REQ_1**.
- **TC_2**: Register with duplicates→ Maps to **REQ_1, REQ_8**. (Duplicate name **RE1**, Duplicate email **RE8**)
- **TC_3**: Register with mismatched passwords → Maps to **REQ_2**.
- **TC_4**: Register with missing required fields → Maps to **REQ_3**.
- **TC_5**: Login with valid admin credentials → Maps to **REQ_4**.
- **TC_6**: Login with valid user credentials → Maps to **REQ_5**.
- **TC_7**: Login with invalid credentials → Maps to **REQ_6**.
- **TC_8**: Login with missing required fields → Maps to **REQ_7**.
- **TC_9**: Register existing user → Maps to **REQ_8**.
- **TC_10**: Ensure passwords are hashed → Maps to **REQ_9**.

Unit Test Cases:
Tests for Register Functionality

1. **Test User Creation**
    - Mock the database to simulate user insertion.
    - Assert that the query runs successfully for valid input.
2. **Test Existing User Check**
    - Mock the database to simulate an existing user.
    - Assert that the error message is generated.
3. **Test Password Mismatch**
    - Simulate different values for password and cpassword.
    - Assert that the error message is generated.

Tests for Login Functionality:



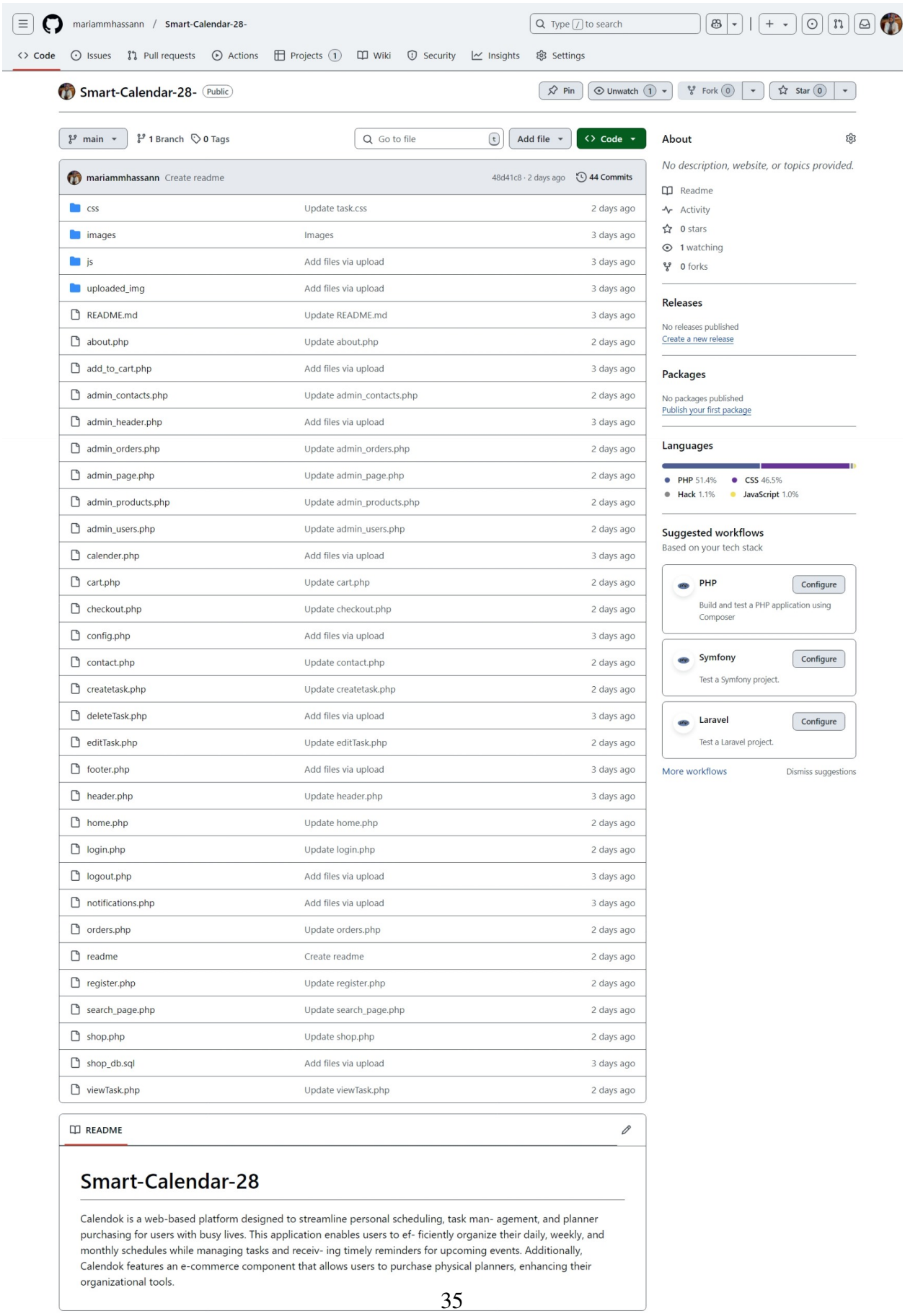
Test Case ID	TC_1	TC_2	TC_3	TC_4	TC_5	TC_6	TC_7	TC_8	TC_9	TC_10	#Test case for respective requirements
REQ_ID											
REQ_1	✕	✕									2
REQ_2			✕								1
REQ_3				✕							1
REQ_4					✕						1
REQ_5						✕					1
REQ_6							✕				1
REQ_7								✕			1
REQ_8		✕							✕		2
REQ_9										✕	1

Figure 24: RTM

1. **Test Successful Login**
   - Mock the database to return a valid user.
   - Assert redirection to the correct page (admin_page.php or home.php(user)).
2. **Test Invalid Login**
   - Mock the database to return no user for invalid credentials.
   - Assert that the error message is generated.
3. **Test Session Variables**
   - Simulate a successful login.
   - Assert that session variables ($SESSION['user_name']) are set correctly.

Table 10: Requirements Ratrix

Req. ID	Requirement Description	Test Case ID
REQ_1	Users can successfully register.	TC_1, TC_2
REQ_2	Registration fails when passwords do not match.	TC_3
REQ_3	Registration fails if required fields are missing.	TC_4
REQ_4	Users can successfully log in as admin.	TC_5
REQ_5	Users can successfully log in as a regular user.	TC_6
REQ_6	Login fails with incorrect credentials.	TC_7
REQ_7	Login fails if required fields are missing.	TC_8
REQ_8	Existing users cannot register with the same email.	TC_9
REQ_9	Passwords are securely hashed during registration and login.	TC_10


mariammhassann / Smart-Calendar-28-
Type / to search
Code
Issues
Pull requests
Actions
Projects 1
Wiki
Security
Insights
Settings
Smart-Calendar-28- Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
Add file
Code
mariammhassann Create readme 48d41c8 · 2 days ago 44 Commits
css Update task.css 2 days ago
images Images 3 days ago
js Add files via upload 3 days ago
uploaded_img Add files via upload 3 days ago
README.md Update README.md 3 days ago
about.php Update about.php 2 days ago
add_to_cart.php Add files via upload 3 days ago
admin_contacts.php Update admin_contacts.php 2 days ago
admin_header.php Add files via upload 3 days ago
admin_orders.php Update admin_orders.php 2 days ago
admin_page.php Update admin_page.php 2 days ago
admin_products.php Update admin_products.php 2 days ago
admin_users.php Update admin_users.php 2 days ago
calender.php Add files via upload 3 days ago
cart.php Update cart.php 2 days ago
checkout.php Update checkout.php 2 days ago
config.php Add files via upload 3 days ago
contact.php Update contact.php 2 days ago
createtask.php Update createtask.php 2 days ago
deleteTask.php Add files via upload 3 days ago
editTask.php Update editTask.php 2 days ago
footer.php Add files via upload 3 days ago
header.php Update header.php 3 days ago
home.php Update home.php 2 days ago
login.php Update login.php 2 days ago
logout.php Add files via upload 3 days ago
notifications.php Add files via upload 3 days ago
orders.php Update orders.php 2 days ago
readme Create readme 2 days ago
register.php Update register.php 2 days ago
search_page.php Update search_page.php 2 days ago
shop.php Update shop.php 2 days ago
shop_db.sql Add files via upload 3 days ago
viewTask.php Update viewTask.php 2 days ago
About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
PHP 51.4%
CSS 46.5%
Hack 1.1%
JavaScript 1.0%
Suggested workflows
Based on your tech stack
PHP
Configure
Build and test a PHP application using Composer
Symfony
Configure
Test a Symfony project.
Laravel
Configure
Test a Laravel project.
More workflows
Dismiss suggestions
README
Smart-Calendar-28
Calendok is a web-based platform designed to streamline personal scheduling, task man- agement, and planner purchasing for users with busy lives. This application enables users to ef- ficiently organize their daily, weekly, and monthly schedules while managing tasks and receiv- ing timely reminders for upcoming events. Additionally, Calendok features an e-commerce component that allows users to purchase physical planners, enhancing their organizational tools.
© 2024 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact
Manage cookies
Do not share my personal information