Department of Mathematics and Computer Science
Faculty of Data Analysis
UNIME

Web Programming Report
Plant Store
E-Commerce

Student:Duishenalieva Milena (551401)
Student: Moldobaeva Adinai (551567)
Professor: Armando Ruggeri

Academic Year 2024-2025

Table of Contents

1. Project Overview

1.1. Objective

The objective of this project is to develop an online store specializing in the sale of indoor plants and plant-related products. The platform allows users to explore a catalog of houseplants, view detailed descriptions and images, and make secure purchases directly through the website. The backend is developed using PHP and MySQL to handle database operations, user authentication, and order processing. On the frontend, HTML, CSS, and JavaScript are used to deliver an intuitive, aesthetically pleasing, and interactive user experience tailored to plant enthusiasts.

1.2. Scope

The online store targets individuals interested in purchasing indoor plants for home or office decoration. It provides a user-friendly interface that enables customers to browse the catalog, filter and search for specific plants, manage a shopping cart, and complete the checkout process. The platform also includes an administrative dashboard that allows authorized users to manage product listings (including plant names, care instructions, and images), view and process customer orders, and maintain user accounts. The project encompasses both frontend and backend development, with a focus on delivering a clean, accessible, and scalable e-commerce solution for plant lovers.

2. Technology Stack

The development of the online store will leverage the following technologies: • Frontend: HTML, CSS, JavaScript • Backend: PHP • Database: MySQL The frontend technologies will be utilized to design the layout, style, and interactivity of the online store interface. PHP will power the backend logic, handling user authentication, product management, and order processing. MySQL will serve as the database management system for storing product information, user data, and order details.

3. Approach

In developing the online store for indoor plants, our primary focus was on simplicity, usability, and efficiency. The frontend design emphasizes a clean and nature-inspired visual interface that is intuitive for users of all experience levels. To ensure a smooth and enjoyable shopping experience, we implemented responsive design practices that allow the platform to adapt seamlessly to different screen sizes and devices.

On the backend, the development process followed best practices in PHP programming to guarantee maintainability, security, and scalability. We implemented robust session-based authentication and input validation mechanisms to safeguard user data and ensure secure access control for both regular users and administrators. The MySQL database structure was optimized for fast data retrieval and transactional integrity, supporting efficient browsing, searching, and order processing.

Throughout the project, we adopted an iterative development methodology. Frequent testing and code reviews were performed to identify and resolve bugs early in the development cycle. Feedback-driven improvements were integrated at each stage, ensuring the platform remains user-focused and functionally sound. By combining a thoughtful user interface with solid backend architecture, we aim to deliver a reliable, scalable, and engaging e-commerce platform tailored to plant lovers and indoor gardening enthusiasts.

4. Database Structure

The database of the online plant store is designed to handle all key aspects of an e-commerce platform, including user management, product listings, shopping cart functionality, order processing, and customer reviews. The database consists of the following tables:

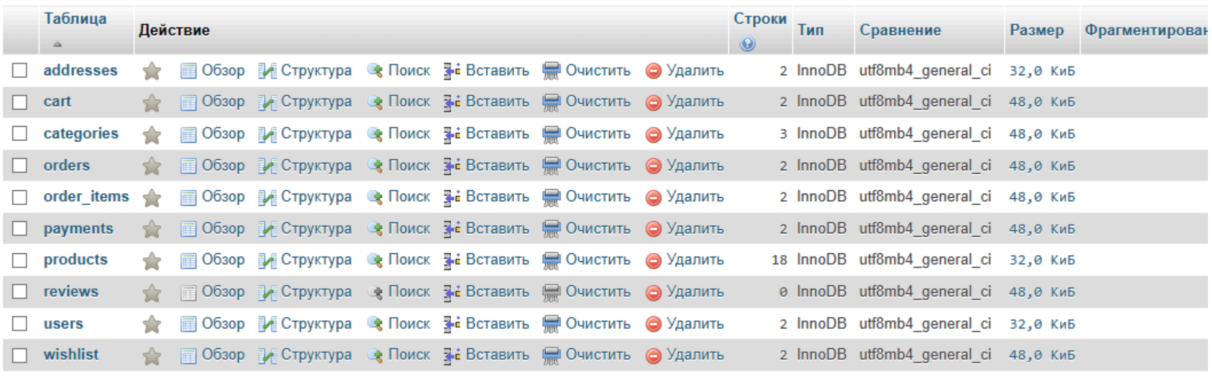


Таблица	Действие						Строки	Тип	Сравнение	Размер	Фрагментирован
addresses	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2	InnoDB	utf8mb4_general_ci	32,0 КиБ	
cart	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2	InnoDB	utf8mb4_general_ci	48,0 КиБ	
categories	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	3	InnoDB	utf8mb4_general_ci	48,0 КиБ	
orders	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2	InnoDB	utf8mb4_general_ci	48,0 КиБ	
order_items	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2	InnoDB	utf8mb4_general_ci	48,0 КиБ	
payments	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2	InnoDB	utf8mb4_general_ci	48,0 КиБ	
products	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	18	InnoDB	utf8mb4_general_ci	32,0 КиБ	
reviews	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	0	InnoDB	utf8mb4_general_ci	48,0 КиБ	
users	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2	InnoDB	utf8mb4_general_ci	32,0 КиБ	
wishlist	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2	InnoDB	utf8mb4_general_ci	48,0 КиБ	

4.1 Users Table

The users table stores information about all registered users of the online plant store, including both administrators and regular customers.

Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique numeric identifier for each user.

- username
  Type: VARCHAR(100)
  The user's display name. Used to personalize the experience and identify users in the frontend.

- email
  Type: VARCHAR(100) — Unique
  The user's email address. It is used as the primary credential for login and communication.

- password
  Type: VARCHAR(255)
  The hashed password for secure login authentication.

- avatar
  Type: VARCHAR(255) — Nullable
  Optional image filename or URL representing the user's profile picture. Can be used in the user dashboard or account section.

- is_admin
  Type: TINYINT(1) — Default: 0
  A flag indicating the user's role:

  - 0 = regular customer

  - 1 = administrator

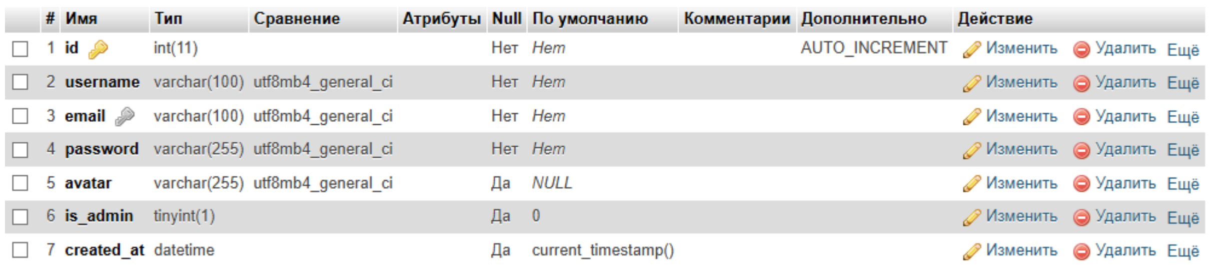
- created_at
  Type: DATETIME — Default: CURRENT_TIMESTAMP
  The date and time the user account was created. Useful for account tracking and analytics.

Usage:

- Handles user authentication and role-based access control.

- Supports profile customization via avatars.

- Enables distinction between admins and customers for restricted areas (e.g., admin panel).



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
☐	1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить Удалить Ещё
☐	2	**username**	varchar(100)	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	3	**email**	varchar(100)	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	4	**password**	varchar(255)	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	5	**avatar**	varchar(255)	utf8mb4_general_ci		Да	*NULL*			Изменить Удалить Ещё
☐	6	**is_admin**	tinyint(1)			Да	0			Изменить Удалить Ещё
☐	7	**created_at**	datetime			Да	current_timestamp()			Изменить Удалить Ещё

4.2 Products Table

The products table contains the catalog of items available in the online plant store, including houseplants and related products.

Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique identifier for each product.

- name
  Type: VARCHAR(100)
  The name of the product, such as the plant type or accessory.

- description
  Type: TEXT
  A detailed description of the product, including care instructions, plant origin, or other relevant information.

- price
  Type: DECIMAL(10,2)
  The cost of the product. Stored with two decimal places to represent currency accurately.

- image
  Type: VARCHAR(512)
  The file name or path to the product image, used for displaying the product visually in the store interface.

- category_id
  Type: INT(11) — Nullable
  Optional foreign key that can link to a category table. Indicates the category the product belongs to (e.g., succulents, pots).

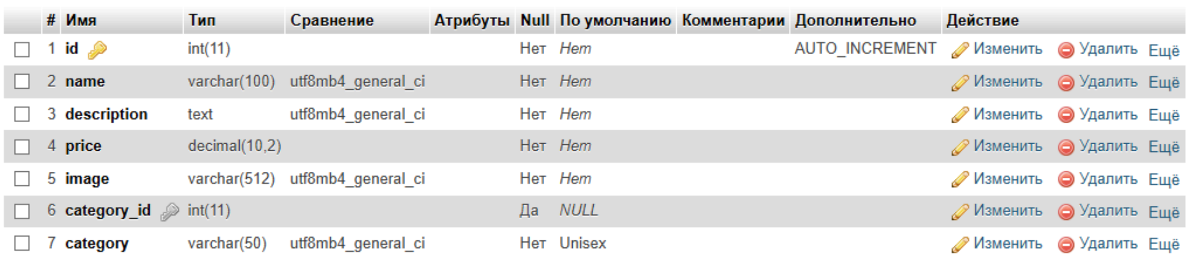
- category
  Type: VARCHAR(50)
  A string representation of the product category. Used when category information is stored directly without a relational reference.

Usage:

- Displays all available products in the shop and homepage.

- Enables filtering or sorting by category.

- Supports product image previews, detailed descriptions, and price display.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие		
☐	1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить	Удалить	Ещё
☐	2	**name**	varchar(100)	utf8mb4_general_ci		Нет	*Нет*			Изменить	Удалить	Ещё
☐	3	**description**	text	utf8mb4_general_ci		Нет	*Нет*			Изменить	Удалить	Ещё
☐	4	**price**	decimal(10,2)			Нет	*Нет*			Изменить	Удалить	Ещё
☐	5	**image**	varchar(512)	utf8mb4_general_ci		Нет	*Нет*			Изменить	Удалить	Ещё
☐	6	**category_id**	int(11)			Да	*NULL*			Изменить	Удалить	Ещё
☐	7	**category**	varchar(50)	utf8mb4_general_ci		Нет	Unisex			Изменить	Удалить	Ещё

4.3 cart Table

The cart table stores the products that users have added to their shopping carts, enabling the platform to manage temporary selections before a purchase is finalized.

Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique identifier for each item in the cart.

- user_id
  Type: INT(11) — Foreign key
  Refers to the id of the user who added the product to their cart.

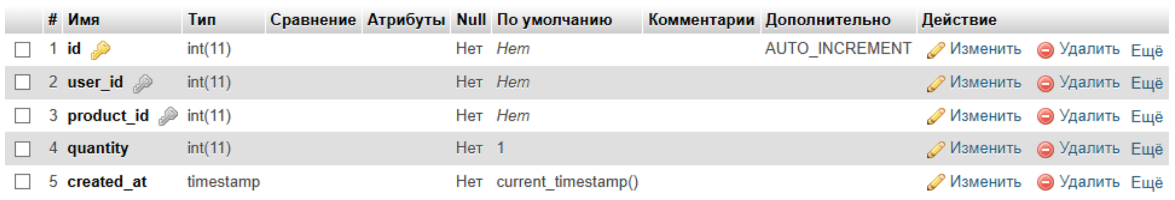
- product_id
  Type: INT(11) — Foreign key
  Refers to the id of the product that was added to the cart.

- quantity
  Type: INT(11) — Default: 1
  Indicates the number of units of the specified product that the user intends to purchase.

- created_at
  Type: TIMESTAMP — Default: CURRENT_TIMESTAMP
  Records the date and time when the item was added to the cart.

Usage:

- Temporarily stores product selections for each user before checkout.
- Supports operations like updating quantity, removing products, or clearing the entire cart.
- Acts as a bridge between browsing and placing an order.



#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	int(11)			Нет	Нет		AUTO_INCREMENT	Изменить Удалить Ещё
2	user_id	int(11)			Нет	Нет			Изменить Удалить Ещё
3	product_id	int(11)			Нет	Нет			Изменить Удалить Ещё
4	quantity	int(11)			Нет	1			Изменить Удалить Ещё
5	created_at	timestamp			Нет	current_timestamp()			Изменить Удалить Ещё

4.4 Orders Table

The orders table stores finalized orders placed by users, linking each purchase to a specific user and shipping address.

Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique identifier for each order.

- user_id
  Type: INT(11) — Foreign key
  References the id of the user who placed the order.

- address_id
  Type: INT(11) — Foreign key
  Refers to the id in the addresses table, indicating where the order should be delivered.

- total_price
  Type: DECIMAL(10,2)
  Represents the total monetary value of the order.

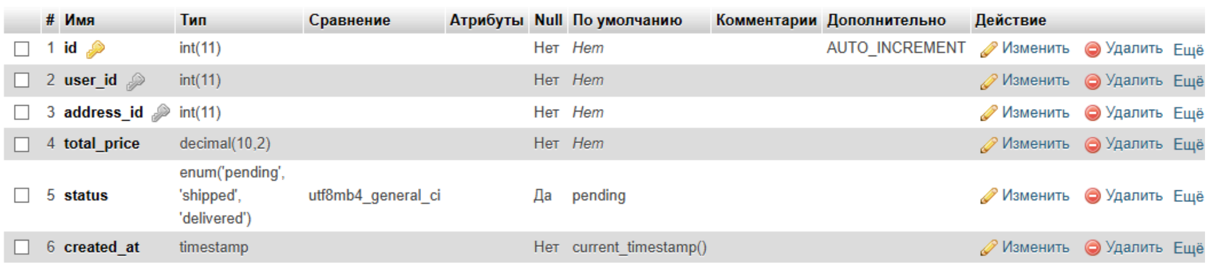
- status
  Type: ENUM('pending', 'shipped', 'delivered') — Default: pending
  Indicates the current status of the order in the processing workflow:

  - pending: Order has been placed but not yet shipped.
  - shipped: Order is on the way.
  - delivered: Order has reached the customer.

- created_at
  Type: TIMESTAMP — Default: CURRENT_TIMESTAMP
  Automatically records the date and time when the order was placed.

Usage:

- Represents individual transactions within the system.
- Links users, delivery addresses, and order totals.
- Enables tracking and status updates throughout the order lifecycle.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие		
☐	1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить	Удалить	Ещё
☐	2	**user_id**	int(11)			Нет	*Нет*			Изменить	Удалить	Ещё
☐	3	**address_id**	int(11)			Нет	*Нет*			Изменить	Удалить	Ещё
☐	4	**total_price**	decimal(10,2)			Нет	*Нет*			Изменить	Удалить	Ещё
☐	5	**status**	enum('pending', 'shipped', 'delivered')	utf8mb4_general_ci		Да	pending			Изменить	Удалить	Ещё
☐	6	**created_at**	timestamp			Нет	current_timestamp()			Изменить	Удалить	Ещё

4.5 Order_items Table

The order_items table stores detailed information about the individual products included in each order. It represents a many-to-one relationship between products and orders, allowing multiple products to be associated with a single order.

Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique identifier for each item entry within an order.

- order_id
  Type: INT(11) — Foreign key
  Refers to the corresponding order in the orders table. It links the item to the order it belongs to.

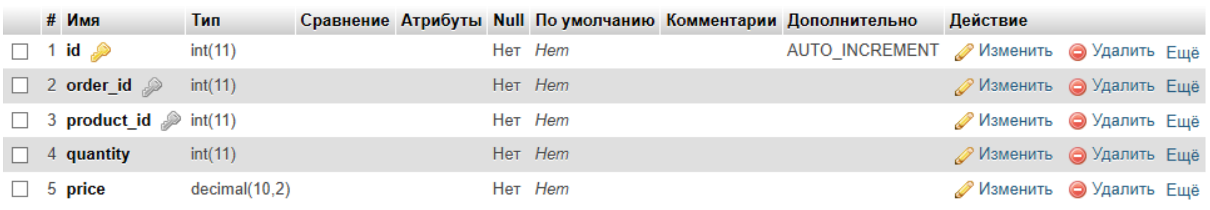
- product_id
  Type: INT(11) — Foreign key
  Identifies the specific product included in the order. Connects to the products table.

- quantity
  Type: INT(11)
  Indicates how many units of the product were ordered.

- price
  Type: DECIMAL(10,2)
  Captures the price of the product at the time of purchase. This allows tracking even if product prices later change.

Usage:

- Allows decomposition of an order into multiple items.
- Enables inventory tracking and accurate billing.
- Supports order history review with item-level detail.



#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие		
1	id	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить	Удалить	Ещё
2	order_id	int(11)			Нет	*Нет*			Изменить	Удалить	Ещё
3	product_id	int(11)			Нет	*Нет*			Изменить	Удалить	Ещё
4	quantity	int(11)			Нет	*Нет*			Изменить	Удалить	Ещё
5	price	decimal(10,2)			Нет	*Нет*			Изменить	Удалить	Ещё

4.6 Wishlist Table

The wishlist table stores information about the products that users have marked as favorites or intend to purchase later. This feature enhances user experience by allowing them to save products of interest.

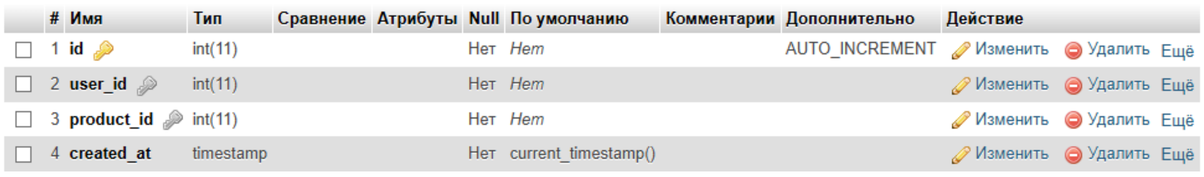
Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique identifier for each wishlist entry.
- user_id
  Type: INT(11) — Foreign key
  Refers to the user who added the product to their wishlist. Links to the users table.
- product_id
  Type: INT(11) — Foreign key
  Identifies the product saved in the wishlist. Links to the products table.
- created_at
  Type: TIMESTAMP — Default: current_timestamp()
  Stores the timestamp when the item was added to the wishlist, helping track user behavior over time.

Usage:

- Allows users to bookmark products for future reference.

- Improves personalization and marketing strategies.
- Supports conversion tracking from wishlist to purchase.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие		
☐	1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить	Удалить	Ещё
☐	2	**user_id**	int(11)			Нет	*Нет*			Изменить	Удалить	Ещё
☐	3	**product_id**	int(11)			Нет	*Нет*			Изменить	Удалить	Ещё
☐	4	**created_at**	timestamp			Нет	current_timestamp()			Изменить	Удалить	Ещё

4.7 Categories Table

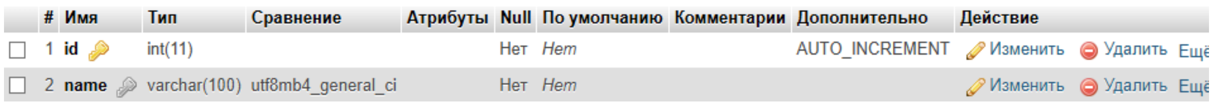
The categories table stores information about the various product categories available in the online plant store. Categories help organize products and make browsing easier for users.

Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique identifier for each product category.
- name
  Type: VARCHAR(100)
  The name of the category, such as “Indoor Plants”, “Succulents”, or “Plant Accessories”.

Usage:

- Provides logical grouping of products in the store.
- Facilitates filtered browsing and searching by category.
- Supports dynamic assignment of products to categories via the category_id field in the products table.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие		
☐	1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить	Удалить	Ещ
☐	2	**name**	varchar(100)	utf8mb4_general_ci		Нет	*Нет*			Изменить	Удалить	Ещ

4.8 Addresses Table

The addresses table stores shipping or billing addresses associated with each user. It enables users to manage delivery locations for their orders.

Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique identifier for each address entry.

- user_id
  Type: INT(11) — Foreign key
  Links the address to the corresponding user in the users table.

- street
  Type: VARCHAR(255)
  The street name and house number of the address.

- city
  Type: VARCHAR(100)
  The city where the address is located.

- postal_code
  Type: VARCHAR(20)
  The postal or ZIP code for the address.

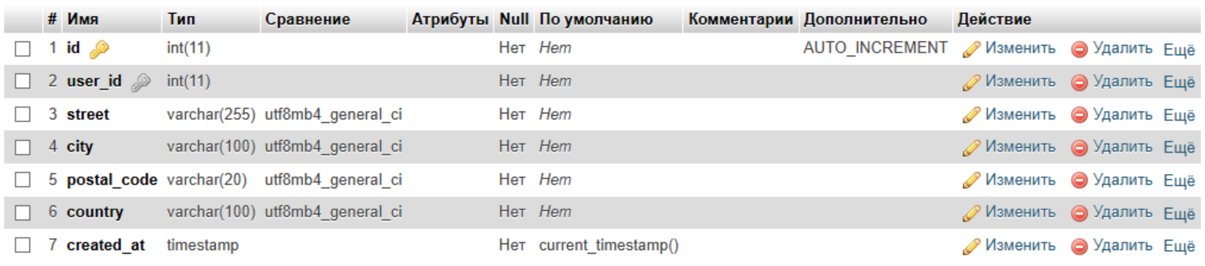
- country
  Type: VARCHAR(100)
  The country of the address.

- created_at
  Type: TIMESTAMP — default: current timestamp
  The date and time when the address was added.

Usage:

- Allows users to store one or multiple delivery addresses.

- Used during the checkout process to assign delivery destinations to orders.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
☐	1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить Удалить Ещё
☐	2	**user_id**	int(11)			Нет	*Нет*			Изменить Удалить Ещё
☐	3	**street**	varchar(255)	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	4	**city**	varchar(100)	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	5	**postal_code**	varchar(20)	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	6	**country**	varchar(100)	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	7	**created_at**	timestamp			Нет	current_timestamp()			Изменить Удалить Ещё

4.9 Payments Table

The payments table stores information related to payment transactions for orders. It links each payment to a specific order and user, and tracks the method and status of the transaction.

Attributes:

- id
  Type: INT(11) — Primary key, Auto Increment
  A unique identifier for each payment entry.

- order_id
  Type: INT(11) — Foreign key
  Links the payment to the corresponding order in the orders table.

- payment_method
  Type: ENUM('card', 'paypal', 'cash')
  Specifies the payment method chosen by the user (e.g., card, PayPal, or cash).

- payment_status
  Type: ENUM('pending', 'completed', 'failed') — default: pending
  Indicates the status of the payment. It can be:

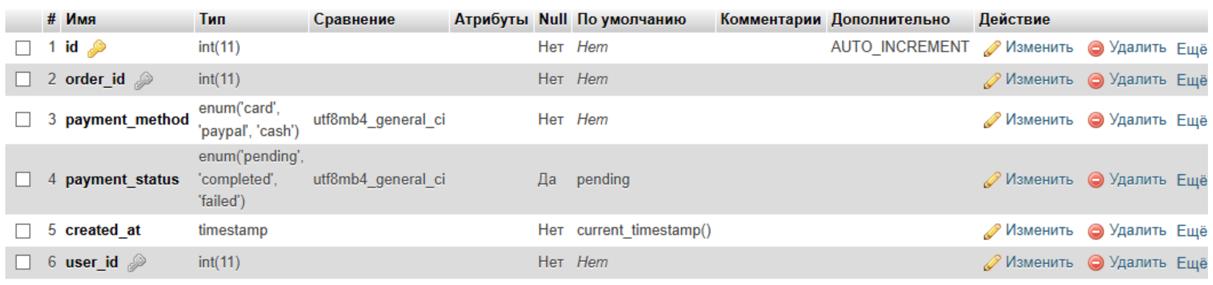
  - pending: the payment is in progress or awaiting confirmation,
  - completed: the payment was successful,
  - failed: the payment was not successful.

- created_at
  Type: TIMESTAMP — default: current timestamp
  The date and time when the payment record was created.

- user_id
  Type: INT(11) — Foreign key
  References the user who made the payment.

Usage:

- This table ensures secure tracking of all financial transactions.
- Useful for auditing, order verification, and customer service.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
☐	1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить Удалить Ещё
☐	2	**order_id**	int(11)			Нет	*Нет*			Изменить Удалить Ещё
☐	3	**payment_method**	enum('card', 'paypal', 'cash')	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	4	**payment_status**	enum('pending', 'completed', 'failed')	utf8mb4_general_ci		Да	pending			Изменить Удалить Ещё
☐	5	**created_at**	timestamp			Нет	current_timestamp()			Изменить Удалить Ещё
☐	6	**user_id**	int(11)			Нет	*Нет*			Изменить Удалить Ещё

4.10. Reviews Table

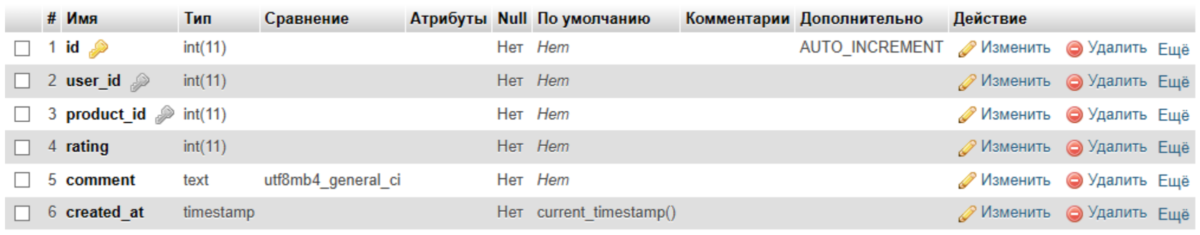
Contains customer feedback for individual products.

Attributes:

- id: Unique identifier for each review (auto-incremented).
- user_id: Foreign key referencing the user who submitted the review.
- product_id: Foreign key referencing the product being reviewed.
- rating: Numerical score provided by the user, typically on a scale (e.g., 1 to 5).
- comment: Textual comment written by the user to share their opinion or experience.
- created_at: Timestamp indicating when the review was created.

Usage:

The reviews table is used to store customer feedback on products available in the online plant store. Each entry represents a review submitted by a registered user, including a rating and an optional comment. This feedback helps other customers make informed purchasing decisions and allows administrators to monitor product satisfaction and quality.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
☐	1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить Удалить Ещё
☐	2	**user_id**	int(11)			Нет	*Нет*			Изменить Удалить Ещё
☐	3	**product_id**	int(11)			Нет	*Нет*			Изменить Удалить Ещё
☐	4	**rating**	int(11)			Нет	*Нет*			Изменить Удалить Ещё
☐	5	**comment**	text	utf8mb4_general_ci		Нет	*Нет*			Изменить Удалить Ещё
☐	6	**created_at**	timestamp			Нет	current_timestamp()			Изменить Удалить Ещё

By structuring the database across these specialized tables, the platform ensures efficient data management, scalability, and support for a complete e-commerce workflow—from browsing to checkout and feedback.

5. Web Pages

5.1 Home Page

Purpose:
The Home Page serves as the main landing page of the online indoor plant store inleaf, aiming to provide a welcoming first impression and easy access to the shopping experience.

Design & Layout:

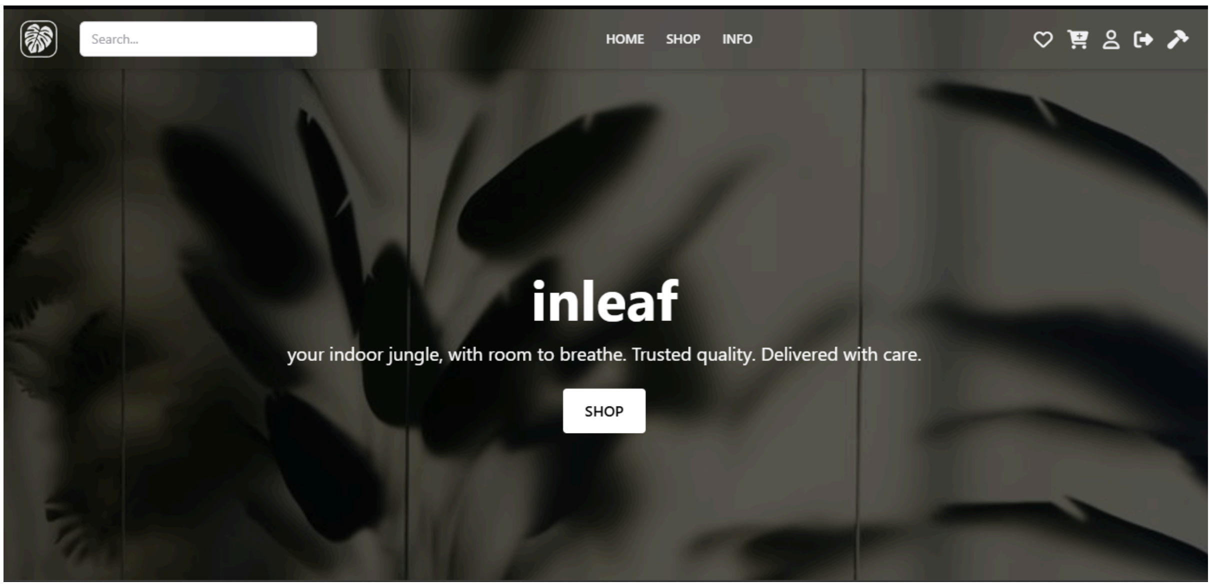
- Hero Section:
  - Large, visually appealing background image with plant-themed design.
  - Prominent branding: site name "inleaf" in bold typography.
  - Subtitle slogan: "your indoor jungle, with room to breathe. Trusted quality. Delivered with care."
  - Central "SHOP" button directing users to the product catalog.
- Top Navigation Bar:
  - Logo (top-left corner): links to the homepage.
  - Search Bar: allows users to search for products by keyword.
  - Navigation Links (right side):
    - HOME, SHOP, INFO
    - Icons for:
      - Wishlist
      - Cart
      - User Account/Login
      - Logout and Admin Panel

Functionality:

- Fully responsive and styled with Tailwind CSS.
- Dynamic rendering of products if required (future extension).
- Routes the user to shop or login/register based on interaction.

User Experience:

- Focuses on clean aesthetics and intuitive navigation.
- Offers immediate access to main shopping functions and account tools.


Search...
HOME
SHOP
INFO
inleaf
your indoor jungle, with room to breathe. Trusted quality. Delivered with care.
SHOP

5.2 Login Page

Purpose:
The login page provides a secure gateway for registered users and administrators to access their personalized content on the platform. It ensures that only authorized individuals can interact with sensitive data such as orders, payment information, and account settings.

Features:

- A clean, minimalist user interface with a centered login form.
- Two input fields:
    - Username or Email: Accepts either the user's registered email or username.

    - Password: Accepts the secure password (usually hashed in the backend).
- A Login button that triggers authentication via a POST request.
- A link to the registration page for new users: "Don't have an account? Register now".

Functionality:

- On form submission, the input data is sent to the server for verification.
- If the credentials are valid:
    - The user is redirected to the homepage or admin dashboard depending on the user_type.
- If the credentials are invalid:
    - An error message is displayed (e.g., "Invalid login credentials").

Security Measures:

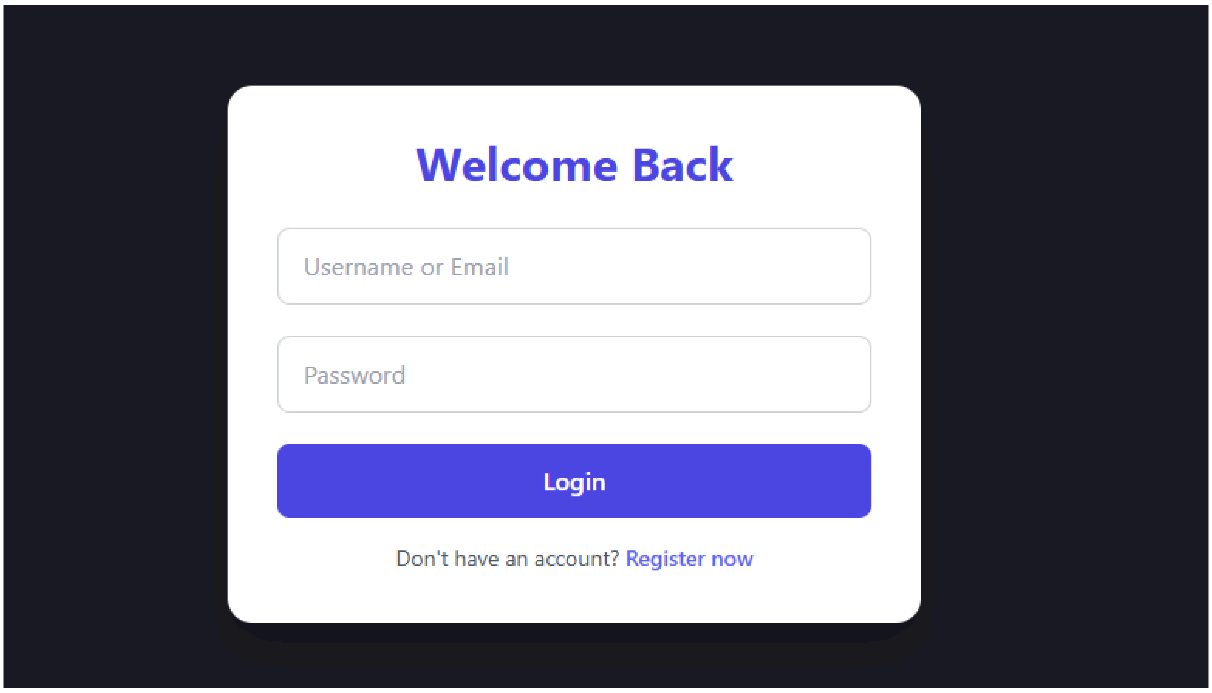
- Passwords are never stored in plain text; they are hashed using secure algorithms (e.g., bcrypt or password_hash).
- SQL injection protection via prepared statements in PHP.
- Session management ensures that logged-in users remain authenticated across pages.

Design & Responsiveness:

- The form is fully responsive and adapts to different screen sizes (mobile, tablet, desktop).
- Styled using Tailwind CSS, ensuring consistency in spacing, colors, and layout.
- Visual elements (e.g., button hover states, shadows, spacing) improve user experience.

Usage Scenario:

- A returning customer visits the website to place an order.
- They click the login icon in the navbar and are taken to the login page.
- Upon entering valid credentials, they are redirected to their account dashboard with access to their cart, wishlist, and past orders.



5.3 Register Page

Purpose:
The registration page allows new users to create an account on the platform, enabling them to place orders, save products to their wishlist, and leave reviews. It is a critical entry point for onboarding new customers.

Features:

- A user-friendly form layout with labeled input fields.
- Fields include:
  - Full Name: The user’s full name.
  - Email Address: Used as a unique identifier and for future logins.
  - Password: Chosen by the user, stored securely in hashed form.
  - Confirm Password: Ensures the user has entered their intended password.
- A Register button to submit the form.
- A link back to the login page: "Already have an account? Login now".

Functionality:

- On form submission, the following backend validations are performed:
    - Email must be unique.
    - Password must meet minimum security requirements (length, characters, etc.).
    - Password and confirmation must match.
- If successful:
    - A new user record is created in the users table with user_type = 'customer'.
    - The user is automatically logged in and redirected to the homepage or user dashboard.
- If unsuccessful:
    - Appropriate error messages are shown (e.g., "Email already in use", "Passwords do not match").

Security Measures:

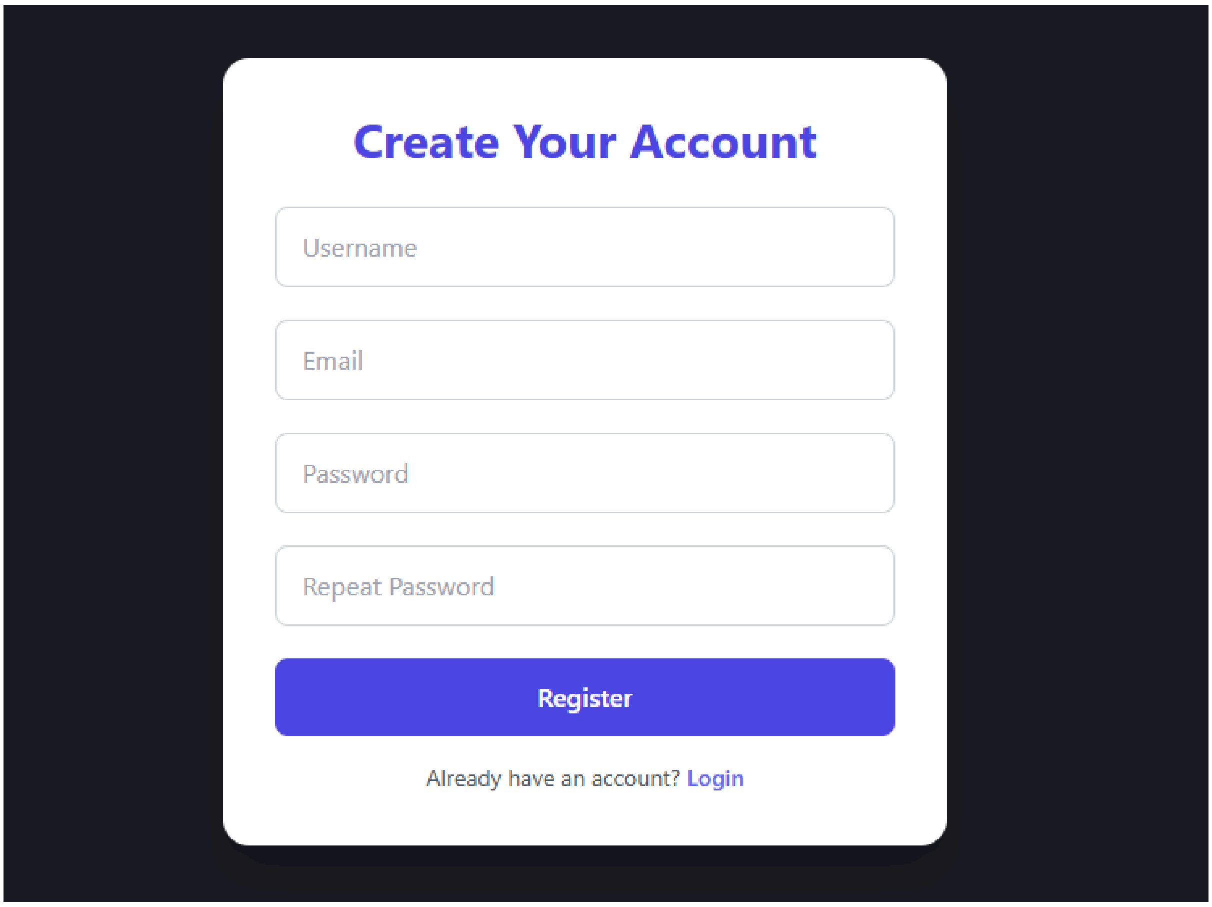
- Passwords are hashed before being stored in the database (using password_hash or similar).
- Backend uses input validation and sanitization to prevent SQL injection.
- May include basic CAPTCHA or rate limiting to prevent spam registrations (optional).

Design & Responsiveness:

- Built with Tailwind CSS for a clean and modern look.
- Fully responsive form: optimized for mobile and desktop use.
- Visual feedback on errors and valid fields (e.g., red border on invalid inputs).

Usage Scenario:

- A new visitor wants to place an order and needs to create an account.
- They click "Register" on the login page, fill in their information, and submit the form.
- After successful registration, they are redirected and can begin shopping immediately.



5.4 Shop Page (Catalog)

Purpose:
The Shop page serves as the central product catalog, allowing users to browse all available products in a structured, visually appealing, and user-friendly format. It is a key part of the user experience, facilitating product discovery and purchasing.

Features:

- A responsive product grid displaying available items from the database.
- Each product card includes:
  - Product Image for visual identification.
  - Product Name as the title.
  - Category Name (e.g., “Plants”).
  - Price (e.g., $89.00).

    - “Add to cart” button.
    - Wishlist Icon (heart) to add to the wishlist.
- A category dropdown filter (top right corner) for narrowing results based on product category.
- Product data is dynamically retrieved from the products and categories tables.

Functionality:

- Clicking “Add to cart” triggers a backend action to insert/update the product in the user's cart (stored in the session or database).
- Clicking the heart icon adds the product to the user's wishlist (insert into wishlist table).
- Selecting a category from the dropdown sends a GET request to filter products by category_id.

Backend Integration:

- Retrieves all products with optional filtering by category.
- May include pagination or lazy loading if the number of products is large.
- Prices are fetched from the products table and formatted to two decimal places.

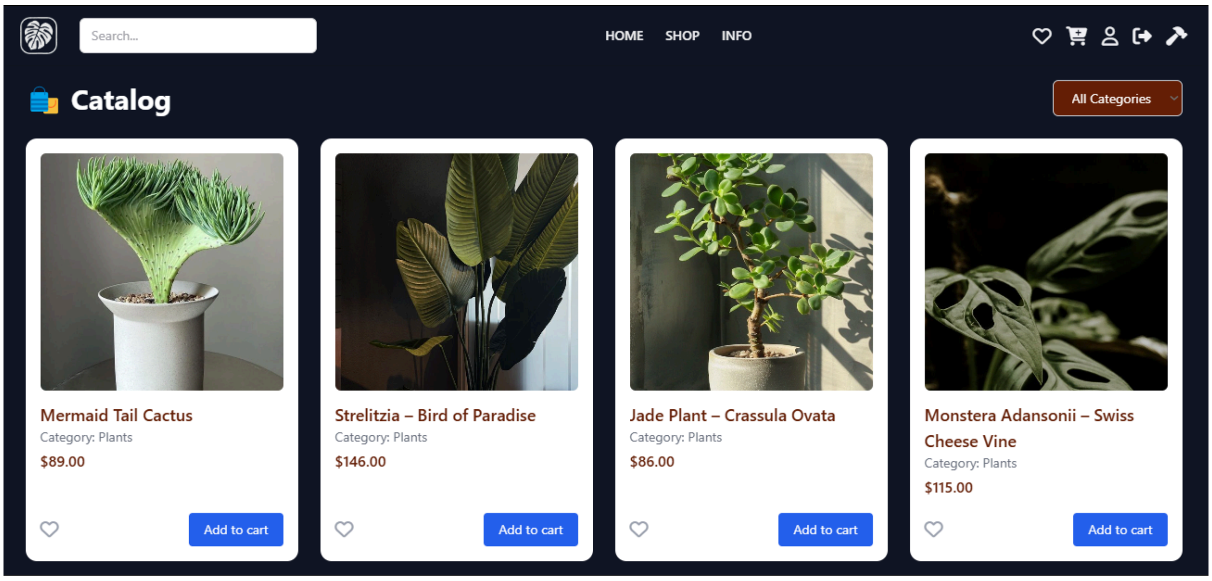
Design & Responsiveness:

- Styled with Tailwind CSS using a grid layout with spacing, rounded cards, shadows, and hover effects.
- Fully responsive across devices:
    - 1 column on small screens.
    - 2–4 columns on larger screens.

User Flow Example:

- A user enters the Shop page, sees multiple plants.
- They use the dropdown to filter by category “Succulents”.

- They click "Add to cart" for a product — the cart icon updates.
- They also save one item to their wishlist via the heart icon.



Catalog

Mermaid Tail Cactus
Category: Plants
$89.00
Add to cart

Strelitzia – Bird of Paradise
Category: Plants
$146.00
Add to cart

Jade Plant – Crassula Ovata
Category: Plants
$86.00
Add to cart

Monstera Adansonii – Swiss Cheese Vine
Category: Plants
$115.00
Add to cart

5.5 Product Detail Page

Purpose:
The Product Detail Page provides users with comprehensive information about a selected product. It enables users to make informed purchase decisions by viewing product details, selecting size options, adjusting quantity, and adding the product to the cart or proceeding directly to checkout.

Features:

- Large product image for clear visual presentation.
- Product name (e.g., Mermaid Tail Cactus) displayed prominently as the page title.
- Detailed product description, often including:
  - Botanical details or origin.
  - Visual characteristics and uniqueness.
  - Ideal use cases (e.g., gift, décor).
- Price displayed clearly and styled distinctively (e.g., bold red).

- Plant size selector table, allowing users to view available sizes (XS to XL), marked with a check (✓).
- Quantity control, with "+" and "–" buttons to increment or decrement the item count.
- Add to Cart button.
- Checkout button for immediate purchase.
- Wishlist (heart) icon for saving the item for later.

Functionality:

- Clicking "Add to Cart" triggers a PHP/JavaScript handler to add the selected item (with size and quantity) to the user's session or database cart.
- "Checkout" redirects the user to the checkout page with the current item preloaded.
- The size availability is dynamically determined based on product inventory (optional feature).
- All actions are logged and connected with the current user (via session or database).

Backend Integration:

- The page retrieves data from the products table using the product_id passed via URL (GET parameter).
- Size availability could be stored in a separate table (product_sizes) or as part of the product metadata.
- Wishlist functionality links with a wishlist table containing user_id and product_id.

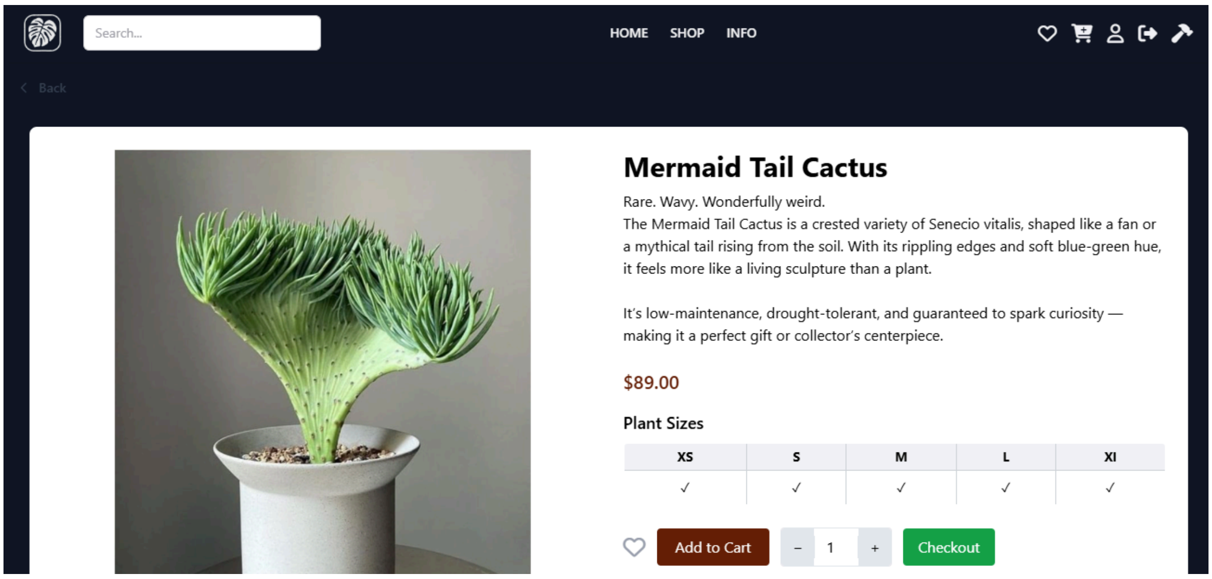
Design & Responsiveness:

- Styled with Tailwind CSS, using a clean two-column layout:
    - Left: product image.
    - Right: product details and action buttons.
- Fully responsive design:
    - On smaller screens, layout stacks vertically.

    - Touch-friendly controls for mobile.

User Flow Example:

- A user navigates from the Shop page and clicks on the Mermaid Tail Cactus.
- They review the description, choose size M, set quantity to 2.
- They add the item to cart, then proceed to checkout directly.



Mermaid Tail Cactus

Rare. Wavy. Wonderfully weird.
The Mermaid Tail Cactus is a crested variety of Senecio vitalis, shaped like a fan or a mythical tail rising from the soil. With its rippling edges and soft blue-green hue, it feels more like a living sculpture than a plant.

It's low-maintenance, drought-tolerant, and guaranteed to spark curiosity — making it a perfect gift or collector's centerpiece.

$89.00

Plant Sizes

XS	S	M	L	XI
✓	✓	✓	✓	✓

5.6 Wishlist Page

Purpose:
The Wishlist Page allows users to save products they are interested in for future reference or purchase. This feature enhances user experience by enabling personalized product tracking without immediate commitment.

Features:

- Grid layout of saved products, each displayed as a card.
- Each card includes:
    - Product image.
    - Product name (clickable, links to product detail page).
    - Price (highlighted in color, e.g., gold/yellow).

    - Delete button to remove item from the wishlist.
- Back button to return to the previous page (e.g., catalog or home).
- Search bar and top navigation bar are consistent with other pages for seamless UX.

Functionality:

- Wishlist is linked to the current user session (user_id) and stored in the backend (likely in a wishlist table).
- Each item can be removed individually via the Delete button, which triggers a PHP handler to delete the row (DELETE FROM wishlist WHERE user_id = ? AND product_id = ?).
- Clicking the product name redirects the user to the Product Detail Page.
- The wishlist can be persistently stored in the database or temporarily in the session for guests.

Backend Integration:

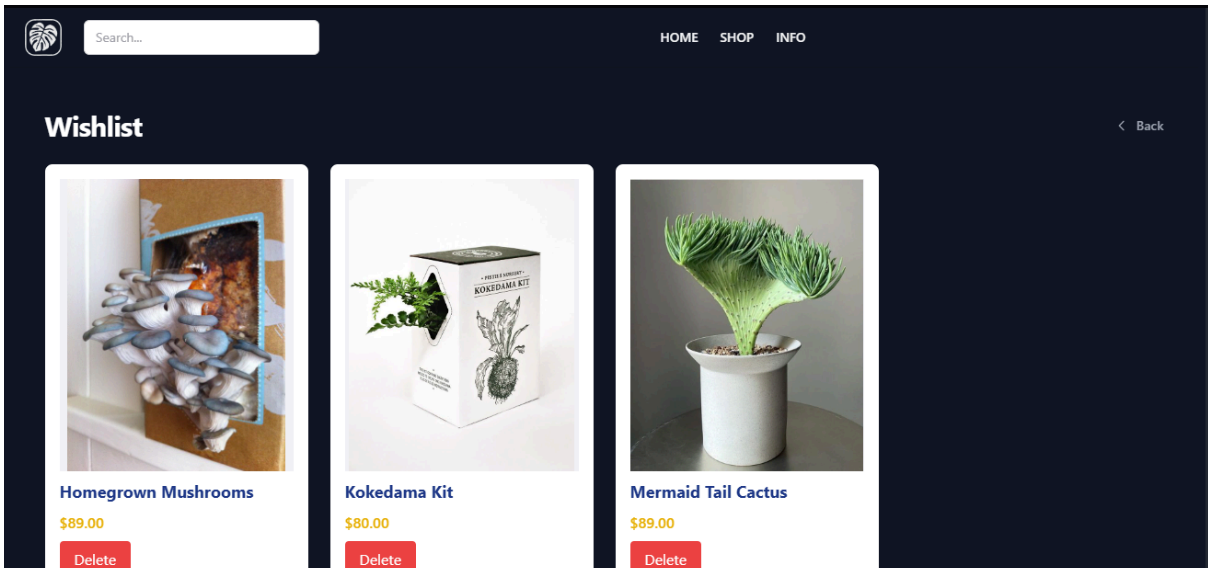
- The wishlist table stores:
    - user_id (FK to users)
    - product_id (FK to products)
    - Optionally: created_at timestamp
- Data is fetched on page load using a SELECT query that joins wishlist and products to retrieve product details for the logged-in user.

Design & Responsiveness:

- Fully responsive layout using Tailwind CSS.
- On wider screens, products appear in multiple columns.
- On smaller devices, cards stack vertically with consistent padding and spacing.
- Dark mode styling with soft backgrounds and vivid action buttons (e.g., red for "Delete").

User Flow Example:

- A user browsing the catalog clicks on the heart icon to save a product.
- The product appears on the Wishlist Page.
- Later, the user visits the Wishlist to decide which items to buy or remove.
-



5.7 Cart Page

Purpose:
The Cart Page allows users to review, update, or remove items selected for purchase before proceeding to checkout. It serves as a temporary storage space for selected products during a shopping session.

Features:

- Card-based layout displaying each item in the user's cart.
- Each product card includes:
  - Image of the product.
  - Product name (clickable link to Product Details page).
  - Quantity control buttons: increment (+) and decrement (–) to update quantity.
  - Price (displayed dynamically based on quantity).
  - Delete button (red) to remove item from the cart.
- A “Proceed to Checkout” button at the bottom of the page, styled in green for clarity.

Functionality:

- Items are fetched from the cart_items or orders_temp table using the current session or user_id.
- Quantity updates trigger an AJAX or POST request to update the quantity in the backend:
  - UPDATE cart SET quantity = quantity ± 1 WHERE user_id = ? AND product_id = ?
- Removal sends a DELETE request to remove the selected product:
  - DELETE FROM cart WHERE user_id = ? AND product_id = ?
- Total price is automatically recalculated as items are updated (in full version).
- The checkout button redirects the user to the Checkout Page where address and payment details are entered.

Backend Integration:

- Likely uses a cart table with fields:
  - user_id (FK)
  - product_id (FK)
  - quantity
  - created_at, updated_at
- PHP/SQL queries manage state across page reloads or session continuation.

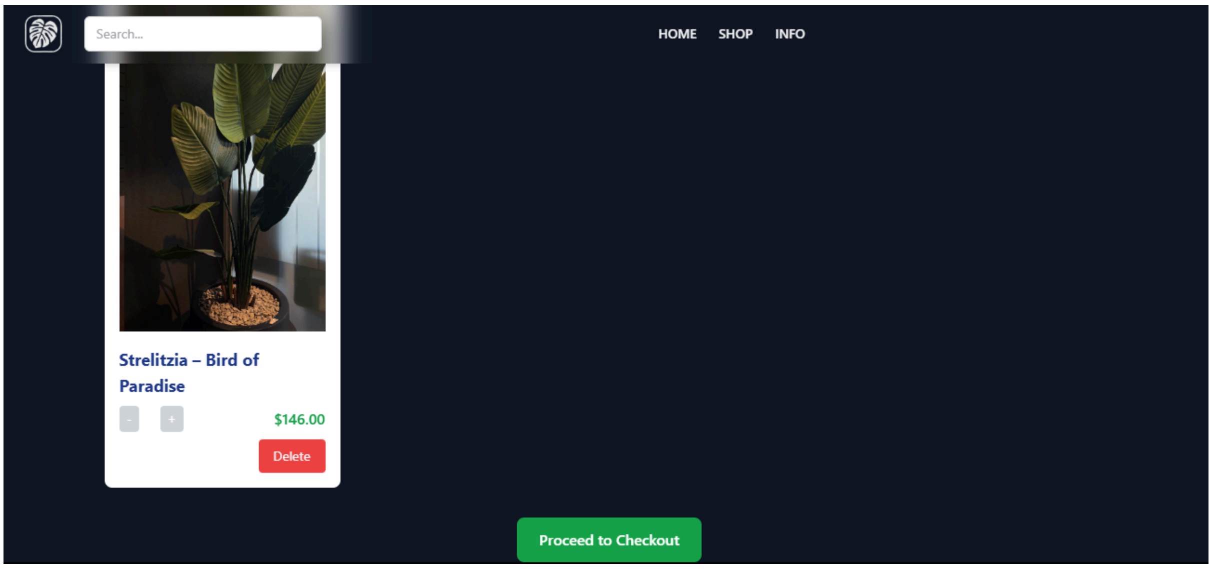
Design & Responsiveness:

- Fully responsive using Tailwind CSS:
  - On mobile: vertical stacking.
  - On desktop/tablet: horizontal grouping with clear margins and spacing.
- Buttons are color-coded for clarity:
  - Quantity controls: neutral grey.
  - Delete: red.

- Checkout: green.

User Flow Example:

- A user adds multiple products to the cart from the catalog.
- On the Cart Page, they review selected items, increase quantity of one plant, and remove another.
- They click "Proceed to Checkout" to complete the order.



5.8 Checkout Page

Purpose:
The Checkout Page is the final step in the purchase process. It collects the user's delivery information and payment method before confirming the order.

Layout Overview:
The interface is split into two primary sections:

1. Order Summary (left panel):
   - Displays a list of selected products with quantity and subtotal.
   - Shows the total amount to be paid.
2. Delivery & Payment (right panel):
   - Contains input fields for delivery address.

    - Provides payment method selection.
    - Navigation buttons for proceeding or going back.

Step-based Form Navigation:

- This is Step 1 of 2, using a progress bar at the top to indicate form advancement.
- Users enter a new address or choose a saved address.
- The form collects:
    - Street
    - Postal Code
    - Country

Payment Method Selection:

- A dropdown menu lets the user select a payment option (e.g., Credit Card, PayPal, Cash on Delivery).
- Backend logic verifies selected method and matches it with predefined payment providers.

Functionality and Backend:

- On submission, a POST request sends the delivery and payment data.
- Data is validated (e.g., valid postal code, selected method).
- On success, the user is redirected to the Order Confirmation Page (Step 2).

User Actions:

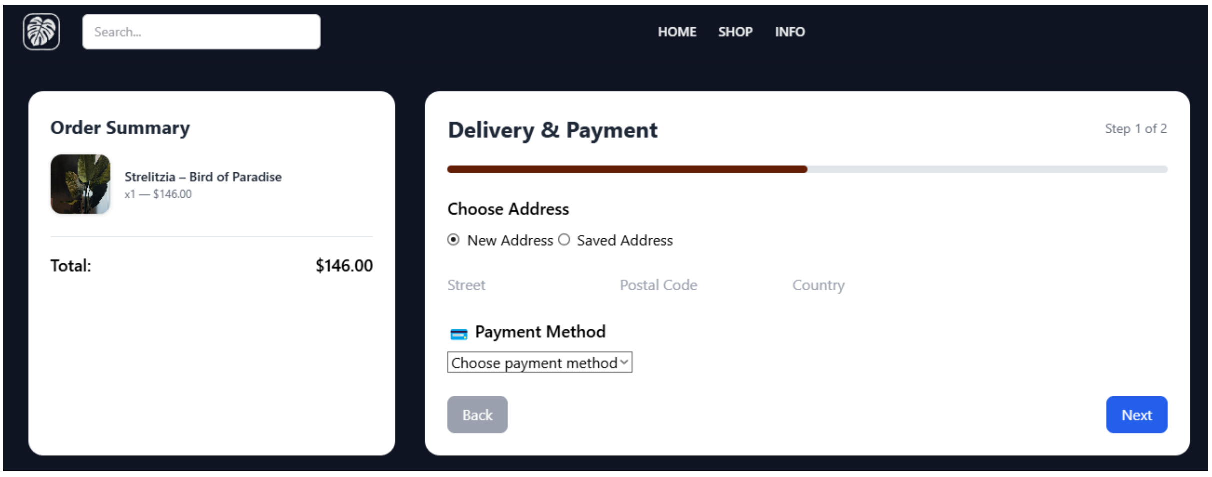
- Fill out delivery info.
- Select payment method.
- Click “Next” to continue to confirmation.

Styling and Responsiveness:

- Fully responsive thanks to Tailwind CSS.
- Clear form layout:
    - Text inputs, radio buttons, and dropdowns are styled uniformly.
    - Buttons (Back, Next) are color-coded for intuitive navigation.

Benefits:

- Clean and minimal UI increases user trust.
- Step-based approach reduces cognitive load.
- Supports both new and returning users via address options.



Order Summary

Strelitzia – Bird of Paradise
x1 — $146.00

Total: $146.00

Delivery & Payment

Step 1 of 2

Choose Address

● New Address ○ Saved Address

Street Postal Code Country

Payment Method

Choose payment method

5.9 Admin Dashboard

Purpose:
The Admin Dashboard serves as a centralized management interface, allowing administrators to oversee the product catalog, manage product information, and maintain the website content efficiently.

Layout Overview:

The Admin Dashboard is divided into two main sections:

1. Add New Product (Left Panel)
   This section allows administrators to seamlessly add new products to the store. It includes the following input fields:

- Product Name: The name/title of the new product.
- Description: Detailed product information, care instructions, or special notes.
- Price: Selling price of the product.
- Select Category: Dropdown menu for categorizing the product (linked with the categories table).
- Image Upload: Provides an intuitive "drag and drop" or "click-to-select" method for uploading product images.
- "Add Product" Button: Submits product details to the database.

2. Product List (Right Panel)
   Displays all existing products in a visually structured manner for easy reference and quick access to product details. Each product card includes:

- Product Image for visual identification.
- Product Name (which can be clicked for editing details or viewing more info).

Functionality:

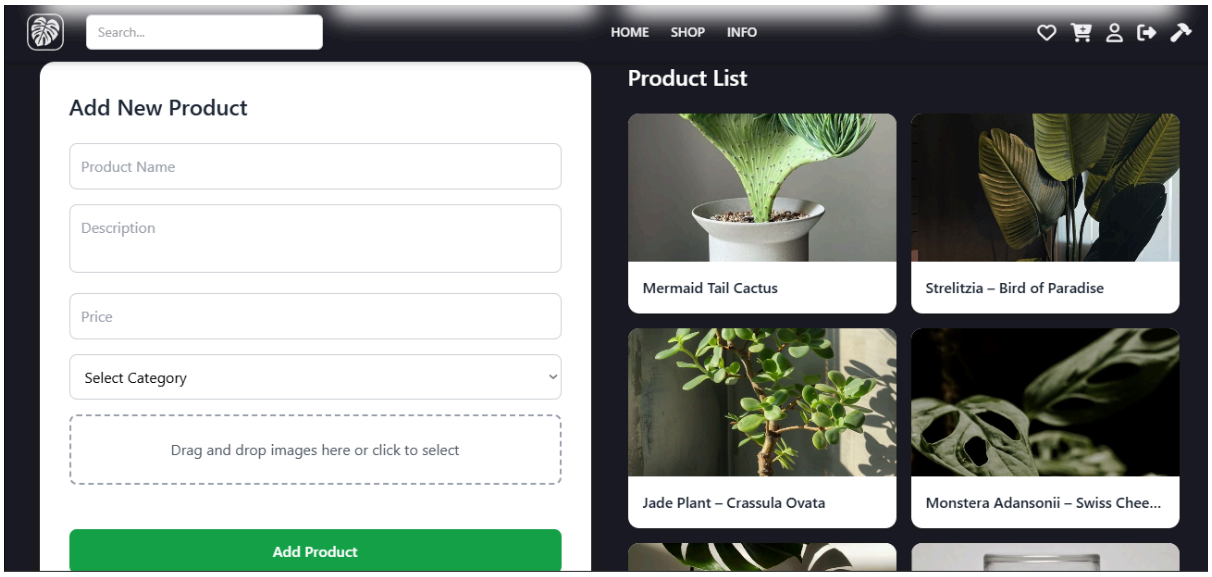
- Adding new products directly updates the product catalog in the database.
- Products displayed in the "Product List" section are dynamically retrieved from the database.
- Easy image management integrated through an intuitive upload interface.

Design & Responsiveness:

- Built with Tailwind CSS, ensuring consistency, modern appearance, and seamless responsiveness across different screen sizes.
- Simplified visual organization enhances usability for administrative tasks.

Usage Scenario:

- Administrators can quickly add new plant products to the online store, edit existing listings, and maintain an organized inventory, ensuring the store content remains accurate and up-to-date.


Search...
HOME
SHOP
INFO
Add New Product
Product Name
Description
Price
Select Category
Drag and drop images here or click to select
Add Product
Product List
Mermaid Tail Cactus
Strelitzia – Bird of Paradise
Jade Plant – Crassula Ovata
Monstera Adansonii – Swiss Chee...

5.10 Profile Page

Purpose:
The Profile Page provides users centralized access to their personal details, order history, and saved shipping addresses. It serves as the main interface for managing and updating user-specific data within the online store.

Layout Overview:

The Profile Page is structured into three primary sections:

1. User Information Panel (Left Section):
    - Displays the user's name and registered email address.
    - Provides functionality to upload or update a profile avatar image.
    - Includes an "Edit Profile" button, allowing users to change their personal details, such as username, password, or contact information.
2. Order History Panel ("My Orders", Upper Right Section):
    - Presents a table listing past and current user orders.
    - Table columns include:
        - #: Order ID or number
        - Item: Thumbnail image and name of the ordered product

        - Qty: Quantity ordered
        - Sum: Total cost of the order
        - Status: Current status of the order (e.g., Pending, Shipped, Delivered)
        - Date: Date when the order was placed
        - Action: Option to delete or manage the order

Address Management Panel ("My Addresses", Lower Right Section):

    - Allows users to add, modify, or delete their shipping addresses directly from their profile page.
    - Each address entry typically includes:
        - Street
        - Postal Code
        - Country
    - Users can add new addresses using the "+" button, and save or remove addresses using respective "Save" and "Delete" buttons.

Functionality and Interaction:

- Profile updates and avatar changes utilize form submissions with PHP and JavaScript for dynamic processing.
- Order management and address changes are synchronized with database entries, reflecting instantly on the interface.
- AJAX requests may be used for real-time updates without requiring full page reloads.

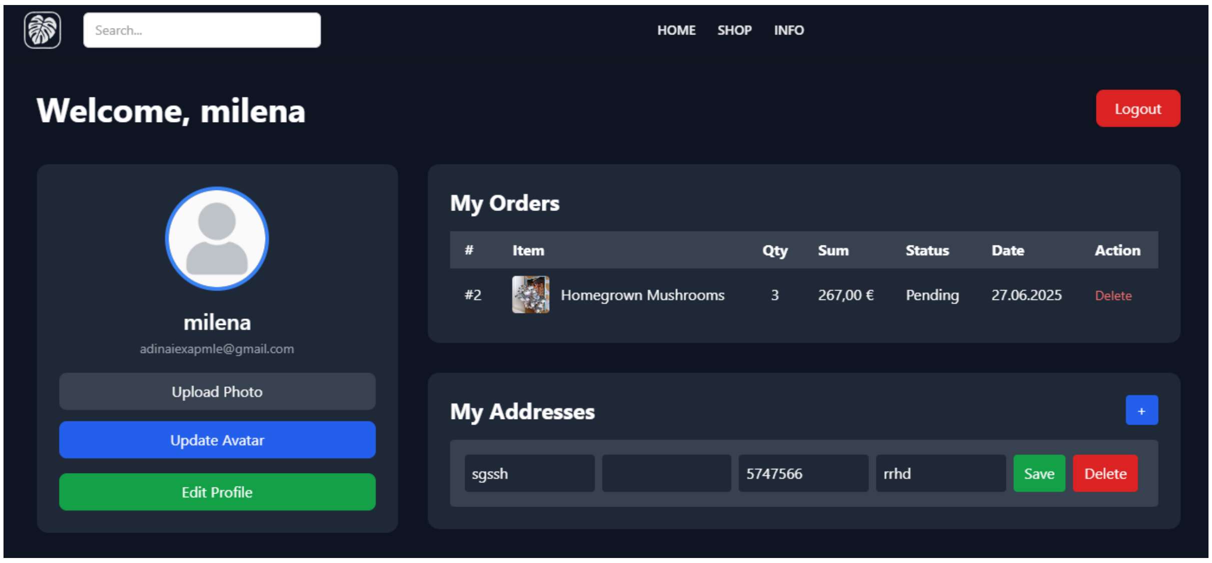
Design & Responsiveness:

- Developed using Tailwind CSS, the page maintains a consistent design aesthetic across devices.

- Responsive layout adapts content presentation effectively on desktop, tablet, and mobile screens.

Typical User Flow:

- Users log into their account and navigate to the Profile Page.
- They review recent orders



Welcome, milena

milena
adinaiexapmle@gmail.com

My Orders

#	Item	Qty	Sum	Status	Date	Action
#2	Homegrown Mushrooms	3	267,00 €	Pending	27.06.2025	Delete

My Addresses

sgssh		5747566	rrhd

5.11. Header Section

Purpose:
The Header Section provides consistent and quick navigation across the entire online store, granting immediate access to essential user actions, including product searches, navigation between main pages, and user account management.

Components:

The header comprises three primary areas:

- Logo and Search Bar (Left Side)
    - Logo: Represents the brand identity of the store ("inleaf") and acts as a quick link back to the homepage.

    - Search Bar: Allows users to quickly search for specific products or categories, enhancing user experience and navigation.

- Main Navigation Menu (Center)
    - Clearly labeled navigation links: HOME, SHOP, and INFO, providing intuitive navigation to key pages.

- User Interaction Icons (Right Side)
  Icons provide quick access to the following user-focused functionalities:
    - Wishlist (Heart icon): Direct access to saved items.
    - Cart (Cart icon): Quick view and access to items selected for purchase.
    - User Account (User icon): Access to login or personal profile page.
    - Logout/Login (Door icon): Allows users to log in or log out of their account.
    - Theme Toggle (Brush icon): Enables users to switch between dark and light modes for visual comfort.
    - Admin Panel (Hammer icon, visible for admin only): Direct access for administrators to manage the store.

Functionality:

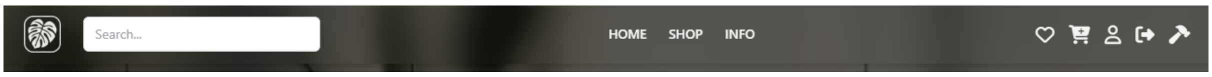
- The header is dynamically updated based on user login status (e.g., showing Admin Panel link for admins only).
- Search functionality sends queries to the database to retrieve and display matching products.
- Wishlist and cart icons dynamically reflect the number of items saved or added by the user.

Design & Responsiveness:

- Styled using Tailwind CSS to ensure responsiveness, clean design, and consistency across all pages.
- Compact and accessible on all devices, including desktops, tablets, and mobile phones.

User Flow Example:

- A user can quickly search for a plant, navigate to their cart, check wishlist items, manage their account, or easily access administrative features if authorized.

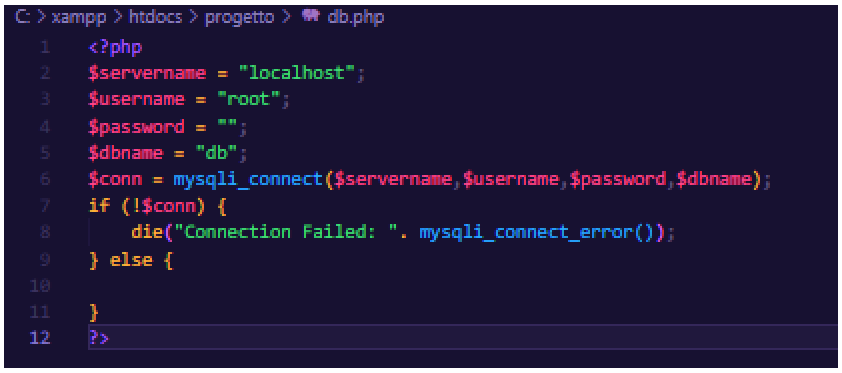


6. Key Functions

This section outlines the main backend and frontend functionalities implemented in the online plant store project. These features support essential actions such as authentication, product management, cart operations, and checkout, ensuring a seamless user experience for both customers and administrators.

6.1. Database Connection

A central configuration file is used to establish a connection between the PHP scripts and the MySQL database. This ensures that all database queries throughout the application are connected via a single secure and reusable entry point. It improves maintainability and reduces duplication of code.


C: > xampp > htdocs > progetto > db.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "db";
$conn = mysqli_connect($servername,$username,$password,$dbname);
if (!$conn) {
    die("Connection Failed: ". mysqli_connect_error());
} else {

}
?>
```

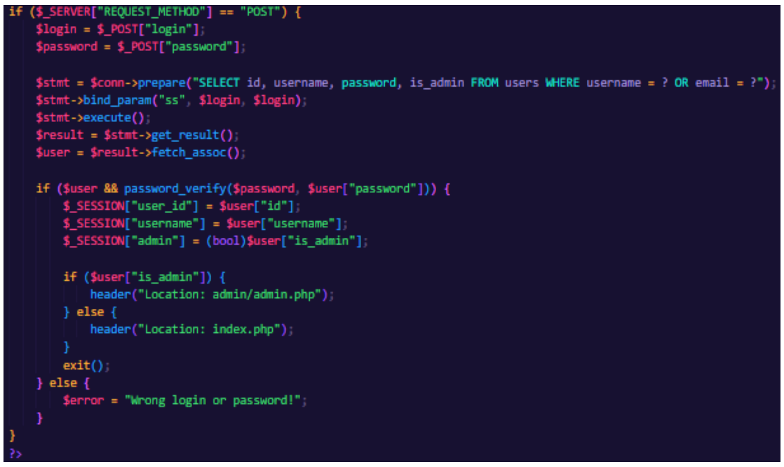
6.2. User Authentication (Login & Registration)

The application supports user authentication via two main functionalities:

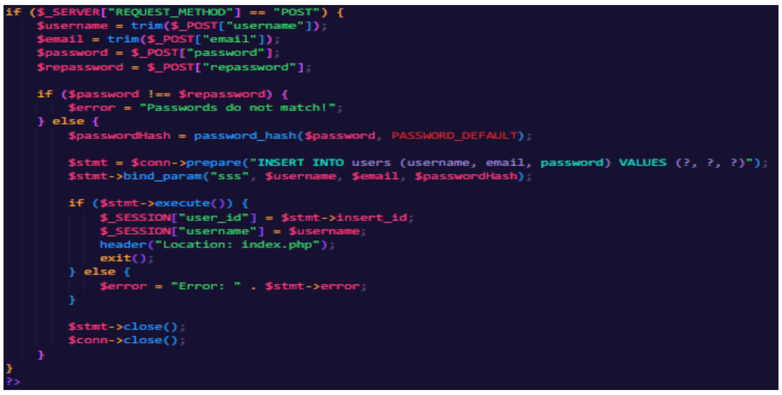
- Login: Validates the user's credentials against the database.

- Registration: Allows new users to sign up by providing their name, email, and password. Passwords are securely hashed before being stored.

These mechanisms ensure secure access to user-specific data such as wishlist, orders, and profile information.



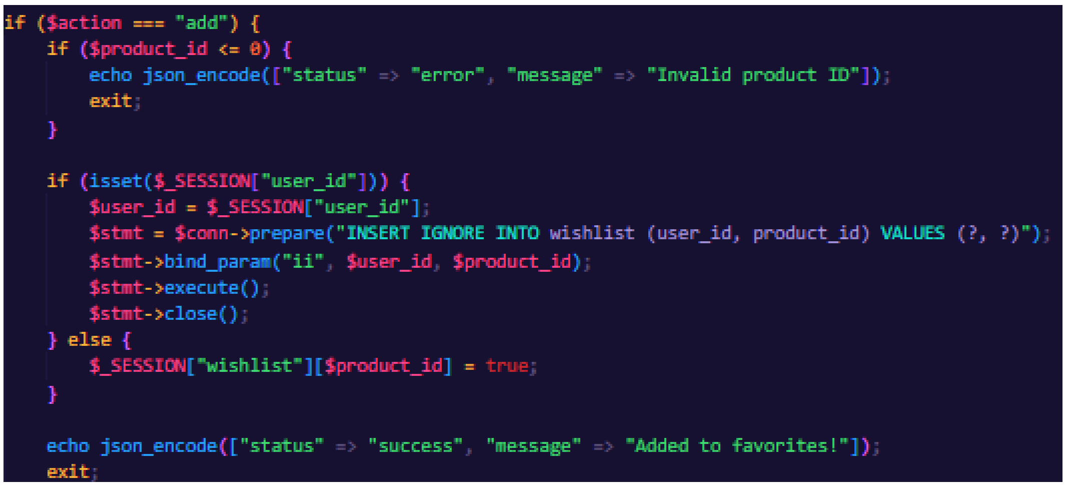
```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $login = $_POST["login"];
    $password = $_POST["password"];

    $stmt = $conn->prepare("SELECT id, username, password, is_admin FROM users WHERE username = ? OR email = ?");
    $stmt->bind_param("ss", $login, $login);
    $stmt->execute();
    $result = $stmt->get_result();
    $user = $result->fetch_assoc();

    if ($user && password_verify($password, $user["password"])) {
        $_SESSION["user_id"] = $user["id"];
        $_SESSION["username"] = $user["username"];
        $_SESSION["admin"] = (bool)$user["is_admin"];

        if ($user["is_admin"]) {
            header("Location: admin/admin.php");
        } else {
            header("Location: index.php");
        }
        exit();
    } else {
        $error = "Wrong login or password!";
    }
}
?>
```



```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST["username"]);
    $email = trim($_POST["email"]);
    $password = $_POST["password"];
    $repassword = $_POST["repassword"];

    if ($password !== $repassword) {
        $error = "Passwords do not match!";
    } else {
        $passwordHash = password_hash($password, PASSWORD_DEFAULT);

        $stmt = $conn->prepare("INSERT INTO users (username, email, password) VALUES (?, ?, ?)");
        $stmt->bind_param("sss", $username, $email, $passwordHash);

        if ($stmt->execute()) {
            $_SESSION["user_id"] = $stmt->insert_id;
            $_SESSION["username"] = $username;
            header("Location: index.php");
            exit();
        } else {
            $error = "Error: " . $stmt->error;
        }

        $stmt->close();
        $conn->close();
    }
}
?>
```

6.3. Wishlist Management

Logged-in users can add products to their wishlist. The wishlist page retrieves items associated with the current user and displays them with options to remove.



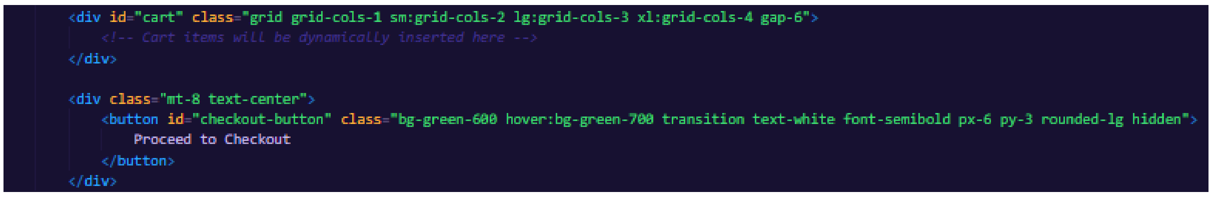
```php
if ($action === "add") {
    if ($product_id <= 0) {
        echo json_encode(["status" => "error", "message" => "Invalid product ID"]);
        exit;
    }

    if (isset($_SESSION["user_id"])) {
        $user_id = $_SESSION["user_id"];
        $stmt = $conn->prepare("INSERT IGNORE INTO wishlist (user_id, product_id) VALUES (?, ?)");
        $stmt->bind_param("ii", $user_id, $product_id);
        $stmt->execute();
        $stmt->close();
    } else {
        $_SESSION["wishlist"][$product_id] = true;
    }

    echo json_encode(["status" => "success", "message" => "Added to favorites!"]);
    exit;
```

6.4. Shopping Cart System

The cart functionality allows users to:

- Add products to their cart.
- Adjust product quantities.
- Remove items.
- Proceed to checkout.

All cart items are session-based or linked to the logged-in user. Prices are dynamically updated based on quantity.



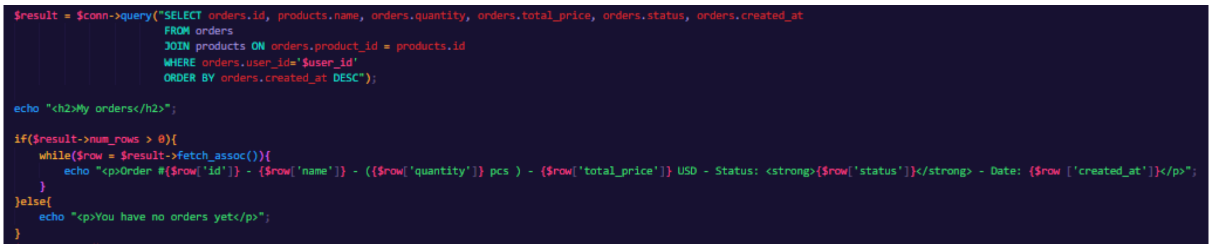
```html
<div id="cart" class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-6">
    <!-- Cart items will be dynamically inserted here -->
</div>

<div class="mt-8 text-center">
    <button id="checkout-button" class="bg-green-600 hover:bg-green-700 transition text-white font-semibold px-6 py-3 rounded-lg hidden">
        Proceed to Checkout
    </button>
</div>
```

6.5. Checkout Process

The checkout is a two-step process:

1. Selecting or entering a delivery address.
2. Choosing a payment method and confirming the order.

The backend validates user input and creates an order record in the database, linking products, quantities, and selected shipping information.

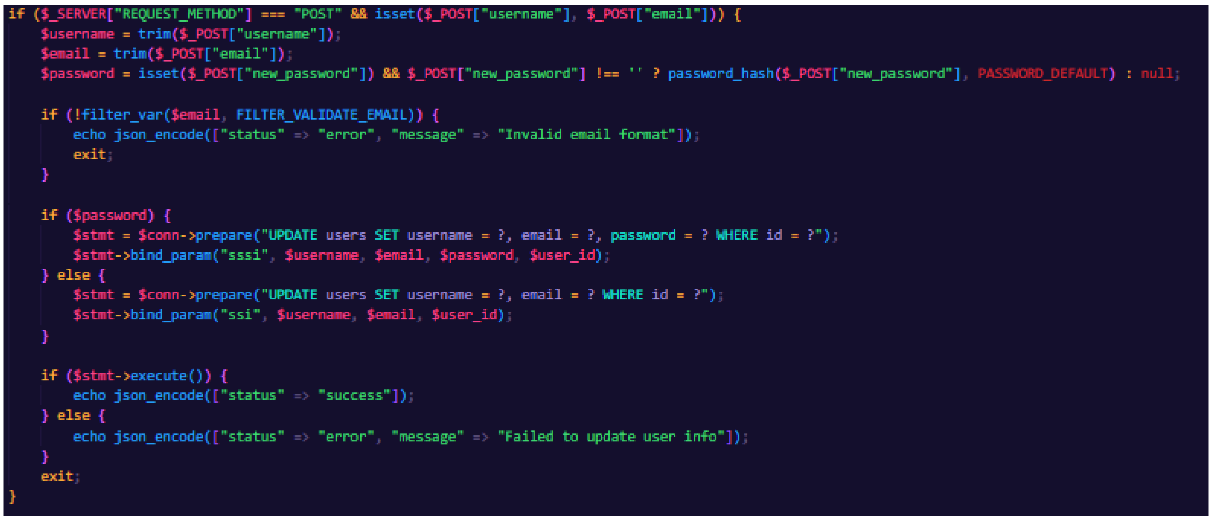


```php
$result = $conn->query("SELECT orders.id, products.name, orders.quantity, orders.total_price, orders.status, orders.created_at
                        FROM orders
                        JOIN products ON orders.product_id = products.id
                        WHERE orders.user_id='$user_id'
                        ORDER BY orders.created_at DESC");

echo "<h2>My orders</h2>";

if($result->num_rows > 0){
    while($row = $result->fetch_assoc()){
        echo "<p>Order #{$row['id']} - {$row['name']} - ({$row['quantity']} pcs ) - {$row['total_price']} USD - Status: <strong>{$row['status']}</strong> - Date: {$row ['created_at']}</p>";
    }
}else{
    echo "<p>You have no orders yet</p>";
}
```

6.6. User Dashboard

The dashboard displays:

- User profile and the ability to upload/update an avatar.
- Order history with status tracking.
- Address management (add/edit/delete addresses).



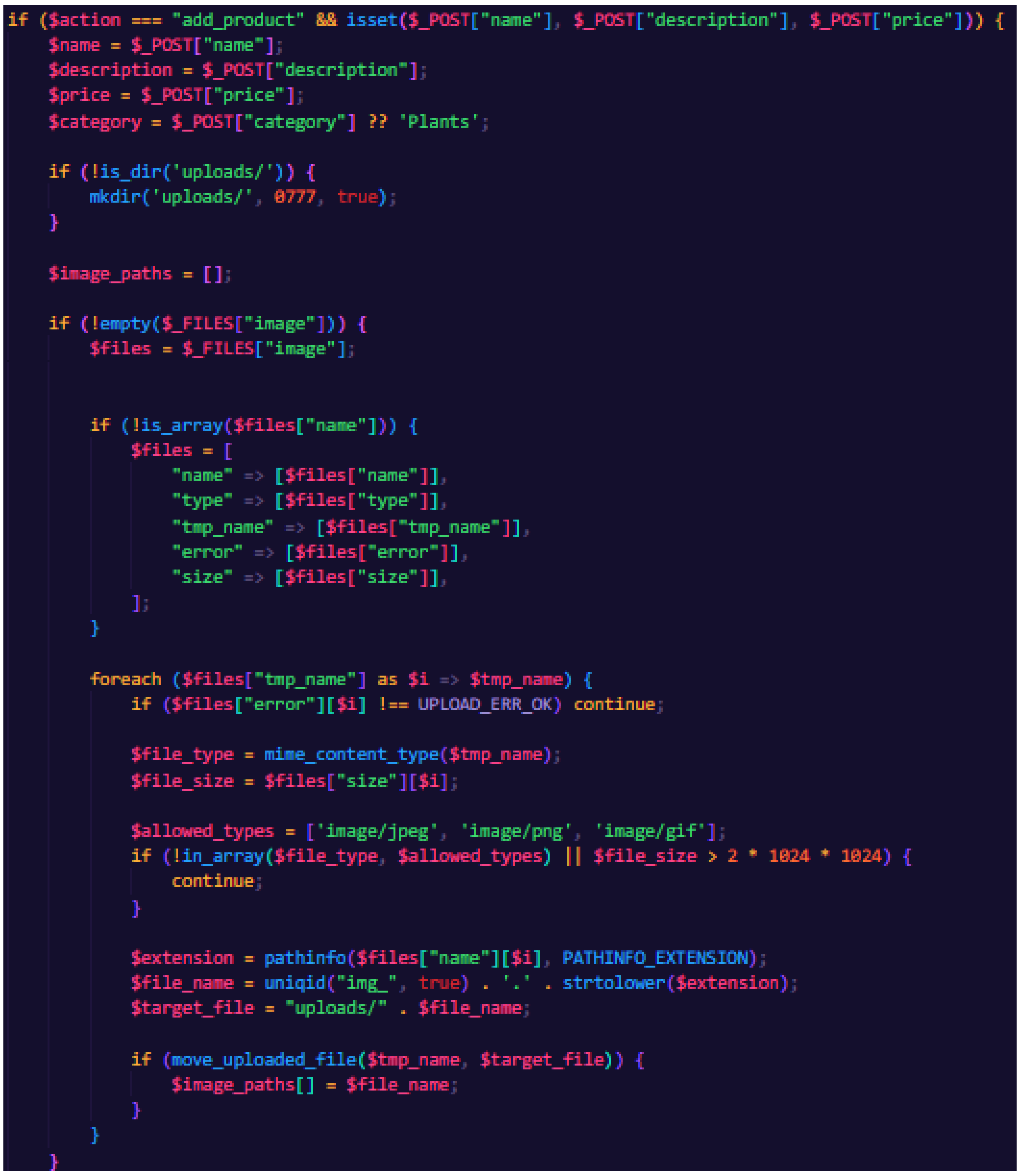
```php
if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST["username"], $_POST["email"])) {
    $username = trim($_POST["username"]);
    $email = trim($_POST["email"]);
    $password = isset($_POST["new_password"]) && $_POST["new_password"] !== '' ? password_hash($_POST["new_password"], PASSWORD_DEFAULT) : null;

    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo json_encode(["status" => "error", "message" => "Invalid email format"]);
        exit;
    }

    if ($password) {
        $stmt = $conn->prepare("UPDATE users SET username = ?, email = ?, password = ? WHERE id = ?");
        $stmt->bind_param("sssi", $username, $email, $password, $user_id);
    } else {
        $stmt = $conn->prepare("UPDATE users SET username = ?, email = ? WHERE id = ?");
        $stmt->bind_param("ssi", $username, $email, $user_id);
    }

    if ($stmt->execute()) {
        echo json_encode(["status" => "success"]);
    } else {
        echo json_encode(["status" => "error", "message" => "Failed to update user info"]);
    }
    exit;
}
```

.

6.7 Admin Panel: Product Management

Administrators have access to a dashboard where they can:

- Add new products (name, description, price, category, image).
- View all existing products.
- Delete or update product information.



```php
if ($action === "add_product" && isset($_POST["name"], $_POST["description"], $_POST["price"])) {
    $name = $_POST["name"];
    $description = $_POST["description"];
    $price = $_POST["price"];
    $category = $_POST["category"] ?? 'Plants';

    if (!is_dir('uploads/')) {
        mkdir('uploads/', 0777, true);
    }

    $image_paths = [];

    if (!empty($_FILES["image"])) {
        $files = $_FILES["image"];


        if (!is_array($files["name"])) {
            $files = [
                "name" => [$files["name"]],
                "type" => [$files["type"]],
                "tmp_name" => [$files["tmp_name"]],
                "error" => [$files["error"]],
                "size" => [$files["size"]],
            ];
        }

        foreach ($files["tmp_name"] as $i => $tmp_name) {
            if ($files["error"][$i] !== UPLOAD_ERR_OK) continue;

            $file_type = mime_content_type($tmp_name);
            $file_size = $files["size"][$i];

            $allowed_types = ['image/jpeg', 'image/png', 'image/gif'];
            if (!in_array($file_type, $allowed_types) || $file_size > 2 * 1024 * 1024) {
                continue;
            }

            $extension = pathinfo($files["name"][$i], PATHINFO_EXTENSION);
            $file_name = uniqid("img_", true) . '.' . strtolower($extension);
            $target_file = "uploads/" . $file_name;

            if (move_uploaded_file($tmp_name, $target_file)) {
                $image_paths[] = $file_name;
            }
        }
    }
```

7. Folder Structure

For our Web Programming project, we organized the application into a set of clearly structured folders, each serving a distinct functional purpose. This modular layout enhances clarity, scalability, and maintainability of the entire platform.

- admin/
  This folder contains all administrative interface components. It includes pages for managing products (admin_item.php), viewing and processing orders (admin_orders.php), handling user accounts (admin_user_panel.php), and uploading product images (upload.php). The admin dashboard is also located here, providing real-time statistics and administrative navigation.

- cart/
  This directory manages the shopping cart system. It includes cart.php to display the cart, cart_handler.php for managing item operations, and checkout.php and place_order.php for handling the order process. The folder ensures a seamless checkout experience for customers.

- profile/
  The profile folder contains files for user profile management, including profile.php for displaying personal data and profile_handler.php for updating user information.

- uploads/
  This folder stores product images uploaded through the admin panel. It keeps all media assets well-organized and accessible for dynamic product display across the site.

- Root PHP Files (e.g., index.php, login.php, register.php, search.php)
  These are key entry points for users and handle the main pages of the website, including the product catalog, login and registration forms, the homepage, and search functionality. Keeping them in the root ensures direct and efficient access.

- Supporting Files

  - db.php: Manages database connection and is included in all backend scripts.

  - script.js: Enhances interactivity through DOM manipulation and user interface behavior.

  - header.php / footer.php: Encapsulate reusable HTML for consistent layout across pages.

This structured folder system reflects a clean separation of frontend and backend responsibilities. It ensures that user-related, admin-related, and shared resources are grouped logically, which simplifies ongoing development, debugging, and expansion of the application.