Part 1: Setting up Database and Controllers

I. Create a Database and Set Up Connection

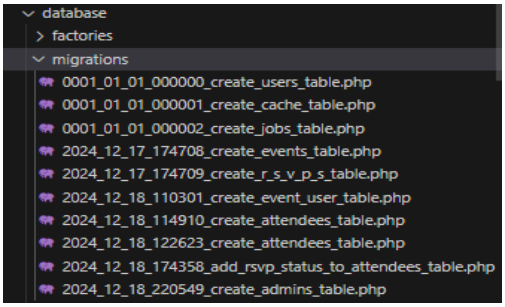
1. **MySQL Database**:
   DATABASE eventify1.1
   .env file

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=eventify1.1_db
DB_USERNAME=root
DB_PASSWORD=

SESSION_DRIVER=file
```

Session Configuration:
config/session.php

```
'driver' => env('SESSION_DRIVER', 'file '),
```

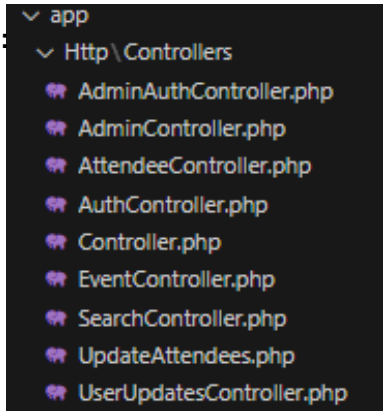
2. **Create Tables Using Migrations**:
   Laravel uses migrations to create tables. To run migrations:



Creating migration files like:
php artisan make:migration create_events_table

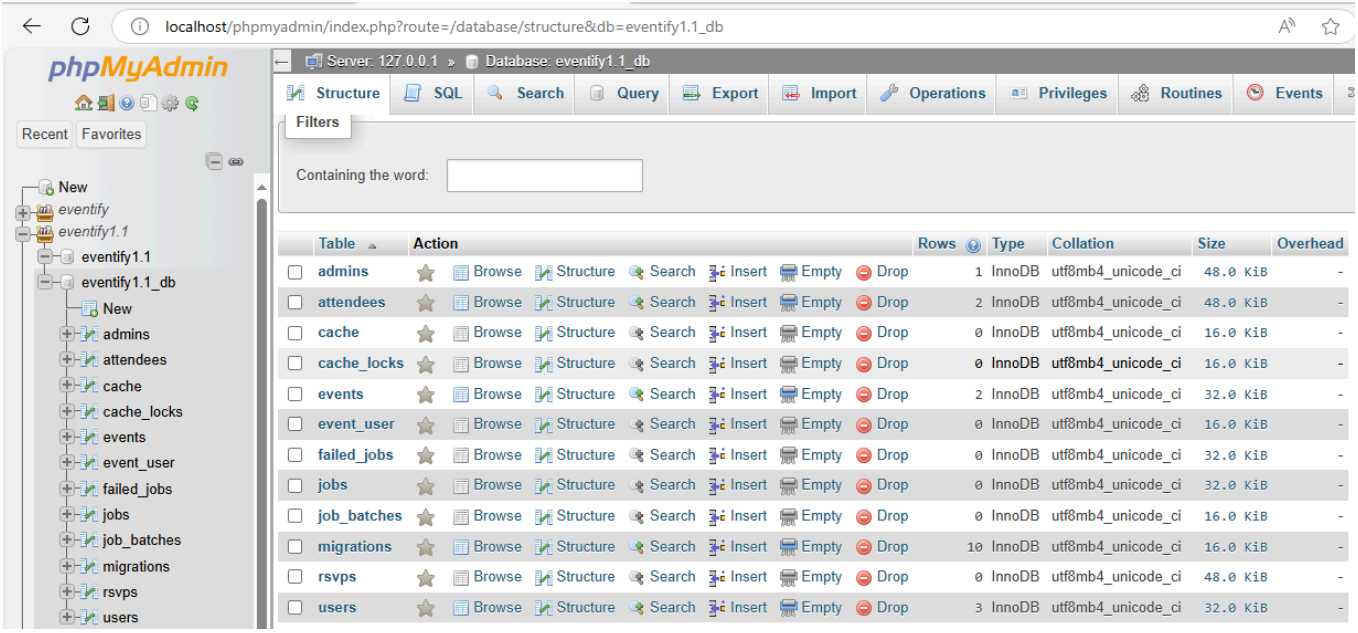
Define the schema in the migration file, then run:
php artisan migrate

II. Step 2: Create Controllers: Each controller serves different routes and methods to interact with data.



Example: AdminController

```
class AdminController extends Controller
{
    1 reference | 0 overrides
    public function show(): View
    {
        $events = Event::all();
        return view('admin.AdminManageEvent', compact(var_name: 'events'));
    }
    1 reference | 0 overrides
    public function edit(string $id): View{
        $event = Event::findOrFail($id);
        return view('admin.EventEdit', compact(var_name: 'event'));
    }
    0 references | 0 overrides
    public function editUser(Request $request, $id): void{
    }
    1 reference | 0 overrides
    public function update(Request $request, string $id): RedirectResponse{
        $request->validate([
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'event_date' => 'required|date',
            'location' => 'required|string',
            'category' => 'required|string',
        ]);
        $event = Event::findOrFail($id);
        $event->update($request->all());
        return redirect()->route('admin.ManageEvent')->with('success', 'Event updated successfully');
    }
    1 reference | 0 overrides
    public function destroy(string $id): RedirectResponse{
        $event = Event::findOrFail($id);
        $event->delete();
        return redirect()->route('admin.ManageEvent')->with('success', 'Event deleted successfully');
    }
    1 reference | 0 overrides
    public function dashboard(): View{
        $events = Event::all(); // Fetch all events
        return view('admin.dashboard', compact(var_name: 'events'));
    }
}
```



III.: Register Controllers in routes/web.php

In routes/web.php, register the routes and associate them with controller methods.

```php
Route::get('/events/add-attendees', [EventController::class, 'addAttendeesForm'])->name('attendees');
Route::post('/events/add-attendees', [EventController::class, 'storeAttendee'])->name('events.addAttendees');
Route::get('events/{event}/edit', [EventController::class, 'edit'])->name('events.edit');

Route::post('checkname', [AttendeeController::class, 'checkName'])->name('checkname');
Route::put('attendees/{id}/rsvp', [AttendeeController::class, 'updateRsvp'])->name('updateRsvp');

Route::put('events/{event}', [EventController::class, 'update'])->name('events.update');
Route::delete('events/{event}', [EventController::class, 'destroy'])->name('events.destroy');

Route::get('/organizerdash', [EventController::class, 'index'])->name('home');

// Auth Routes
Route::get('register', [AuthController::class, 'showRegisterForm'])->name('register');
Route::post('register', [AuthController::class, 'register']);
Route::get('login', [AuthController::class, 'showLoginForm'])->name('login');
Route::post('login', [AuthController::class, 'login']);
Route::post('logout', [AuthController::class, 'logout'])->name('logout');

// Admin Auth Routes
Route::get('admin/register', [AdminAuthController::class, 'showRegisterForm'])->name('admin.register');
Route::post('admin/register', [AdminAuthController::class, 'register']);
Route::get('admin/login', [AdminAuthController::class, 'showLoginForm'])->name('admin.login');
Route::post('admin/login', [AdminAuthController::class, 'login']);
Route::get('admin/logout', [AdminAuthController::class, 'logout'])->name('admin.logout');

// Event Routes
Route::resource('events', EventController::class);
Route::post('events/{event}/rsvp', [EventController::class, 'rsvp'])->name('events.rsvp');
```

```php
// Attendees CRUD
Route::get('/attendees/{attendee}/edit', [AttendeeController::class, 'edit'])->name('attendees.edit');
Route::put('/attendees/{attendee}', [AttendeeController::class, 'update'])->name('attendees.update');
Route::delete('/attendees/{attendee}', [AttendeeController::class, 'destroy'])->name('attendees.destroy');

Route::get('/dashboard', [AdminController::class, 'dashboard'])->name('dashboard');
Route::get('admin/manage', [AdminController::class, 'show'])->name('admin.ManageEvent');
Route::get('manage/{id}/edit', [AdminController::class, 'edit'])->name('admin.editing');
Route::put('admin/{event}', [AdminController::class, 'update'])->name('admin.updating');
Route::delete('admin/{event}', [AdminController::class, 'destroy'])->name('admin.destroying');

//admin edit attendees
Route::get('/admin/manage-events', [UpdateAttendees::class, 'index'])->name('admin.ManageAttendeesAdmin');
Route::get('/admin/attendees/{id}/edit', [UpdateAttendees::class, 'edit'])->name('admin.attendees.edit');
Route::put('/admin/attendees/{id}', [UpdateAttendees::class, 'update'])->name('admin.attendees.update');
Route::delete('/admin/attendees/{id}', [UpdateAttendees::class, 'destroy'])->name('admin.attendees.destroy');
Route::resource('users', UserUpdatesController::class);

// users
// Show all users
Route::get('/users', [UserUpdatesController::class, 'index'])->name('users.ManageUsersLog');

// Show the form for editing a user
Route::get('/users/{id}/edit', [UserUpdatesController::class, 'edit'])->name('users.edit');

// Update a user
Route::put('/users/{id}', [UserUpdatesController::class, 'update'])->name('users.update');

// Delete a user
Route::delete('/users/{id}', [UserUpdatesController::class, 'destroy'])->name('users.destroy');
```

Part 2: Account Registration and Login

I. Implement Registration and Login Pages: **resources/views/auth/.**

Login Form: views/auth/login.blade.php

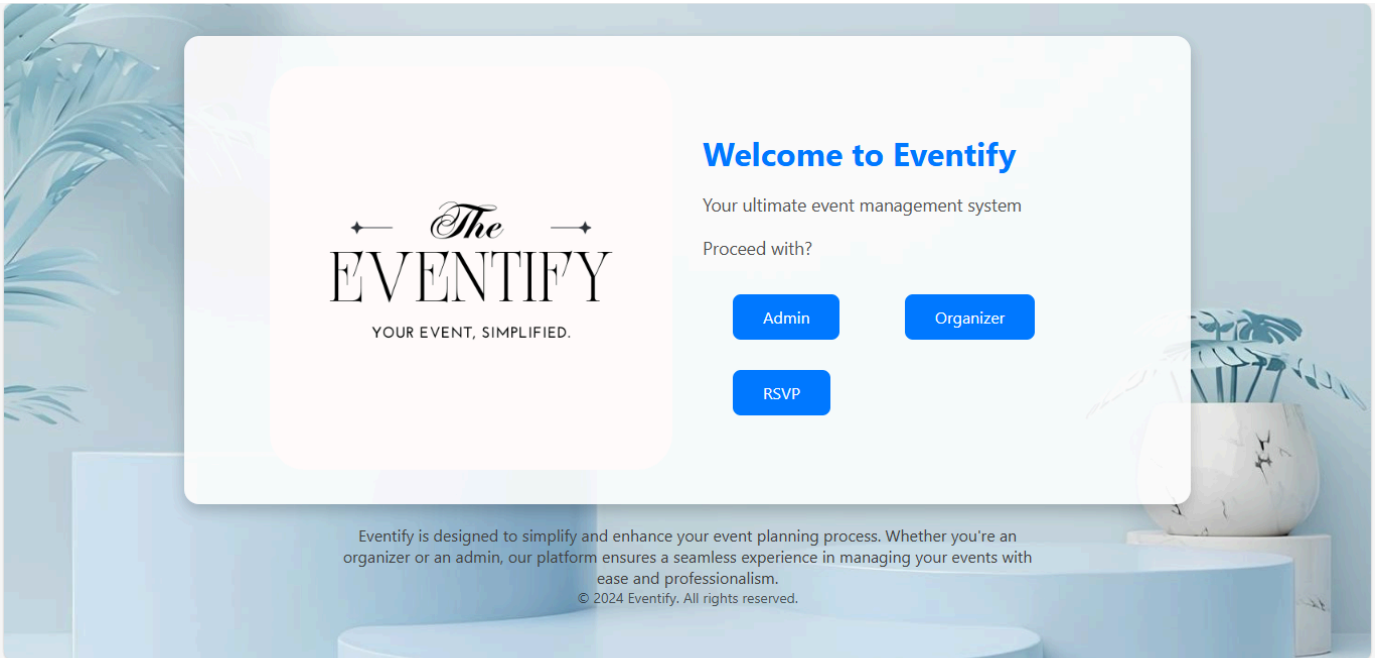
```blade
<body>
    <div class="register-container">
        <!-- Image and Intro Section -->
        <div class="image-section">
        <img src="{{ asset('images/eventify.png') }}" alt="Eventify Logo">
            <p>Your journey to unforgettable events starts here. Join us today and explore endless possibilities.</p>
        </div>

        <!-- Form Section -->
        <div class="form-section">
            <h1>Log into Your Account</h1>
            <form action="{{ url('login') }}" method="POST">
                @csrf
                <input type="email" name="email" placeholder="Email" class="form-control" required>
                <input type="password" name="password" placeholder="Password" class="form-control" required>
                <button type="submit">Login</button>
            </form>
            <div class="footer">
                Don't have an account? <a href="{{ url('register') }}">Register here</a>
                <br>
                <h5></h5><a href="javascript:history.back()" class="back-button">Back</a>
            </div>
        </div>
    </div>
</body>
```

Register Form: views/auth/register.blade.php

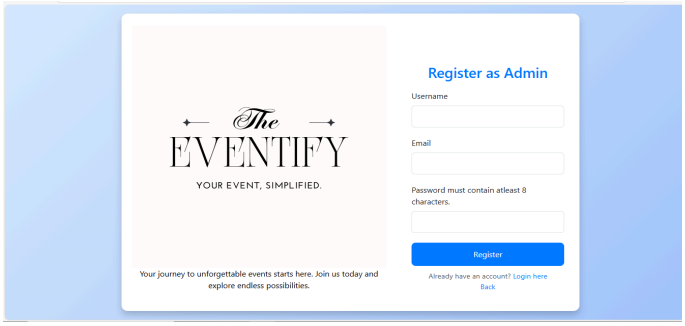
```blade
<body>
    <div class="register-container">
        <!-- Image and Intro Section -->
        <div class="image-section">
        <img src="{{ asset('images/eventify.png') }}" alt="Eventify Logo">
            <p>Your journey to unforgettable events starts here. Join us today and explore endless possibilities.</p>
        </div>

        <!-- Form Section -->
        <div class="form-section">
            <h1>Register</h1>
            <form action="{{ url('register') }}" method="POST">
                @csrf
                <input type="text" name="name" placeholder="Name" class="form-control" required>
                <input type="email" name="email" placeholder="Email" class="form-control" required>
                <input type="password" name="password" placeholder="Password" class="form-control" required>
                <input type="password" name="password_confirmation" placeholder="Confirm Password" class="form-control" required>
                <button type="submit">Register</button>
            </form>
            <div class="footer">
                Already have an account? <a href="{{ url('login') }}">Login here</a>
                <br>
                <a href="javascript:history.back()" class="back-button">Back</a>
            </div>
        </div>
    </div>
</body>
```

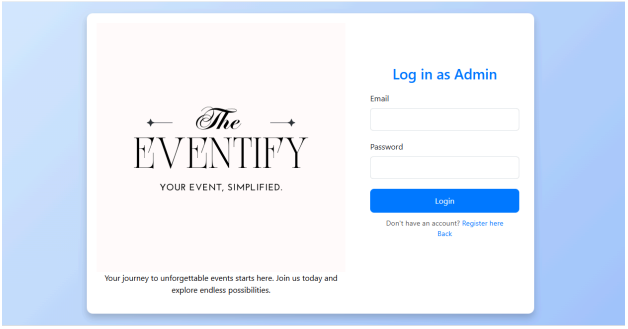
- Users register and log in via **AuthController**, which uses Laravel's authentication features.
    - The form posts data to the login/register route.
    - Includes CSRF protection (**@csrf**) to prevent cross-site request forgery.
    - Collects user email and password for authentication.

- Implement Role Selection: When registering, we can choose roles like 'Organizer' or 'Admin'. Can add a role column to users table and handle role assignment in the registration process.

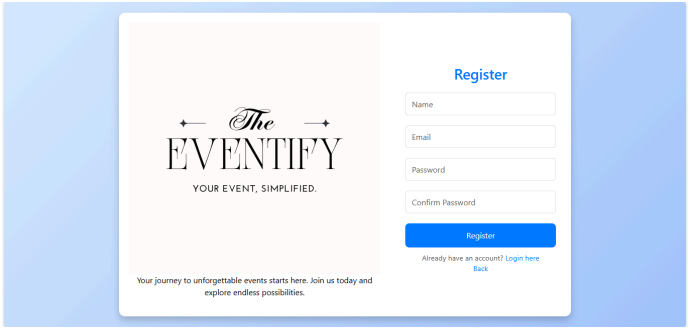


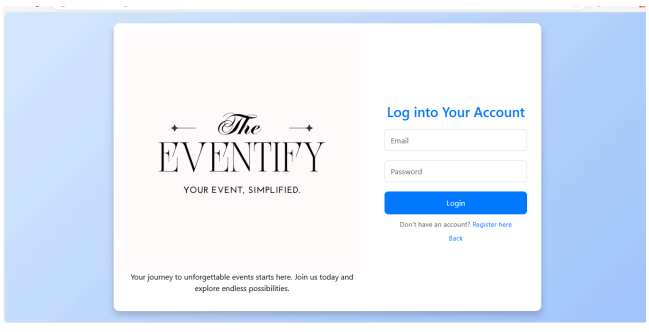
For Admin:





For Organizer:





Part 3: Populating Pages Using Data from Database

I. Fetch Data in Controller and Pass to Views

In AdminController, fetch data from the Event model and pass it to the view.

$events

```php
public function show(): View
{
    $events = Event::all();
    return view('admin.AdminManageEvent', compact(var_name: 'events'));
}
1 reference | 0 overrides
public function edit(string $id): View{
    $event = Event::findOrFail($id);
    return view('admin.EventEdit', compact(var_name: 'event'));
}
0 references | 0 overrides
public function editUser(Request $request, $id): void{
}
1 reference | 0 overrides
public function update(Request $request, string $id): RedirectResponse{
    $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'required|string',
        'event_date' => 'required|date',
        'location' => 'required|string',
        'category' => 'required|string',
    ]);
    $event = Event::findOrFail($id);
    $event->update($request->all());
    return redirect()->route('admin.ManageEvent')->with('success', 'Event updated successfully');
}
1 reference | 0 overrides
public function destroy(string $id): RedirectResponse{
    $event = Event::findOrFail($id);
    $event->delete();
    return redirect()->route('admin.ManageEvent')->with('success', 'Event deleted successfully');
}
1 reference | 0 overrides
public function dashboard(): View{
    $events = Event::all(); // Fetch all events
    return view('admin.dashboard', compact(var_name: 'events'));
}
```

Model/Event.php

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

20 references | 0 implementations
class Event extends Model
{
    use HasFactory;
    0 references
    protected $table = 'events';

    0 references
    protected $fillable = [
        'title',
        'description',
        'event_date',
        'location',
        'category',
        'user_id',
    ];

    0 references | 0 overrides
    public function attendees(): HasMany
    {
        return $this->hasMany(Attendee::class);
    }
```

Key Functions:

- **show():** Fetches all events and passes them to a view.
- **edit()**: Loads a specific event for editing.
- **update():** Updates an event's details in the database.
- **destroy():** Deletes an event.

Provides CRUD functionality for event management.

The **$events** variable in the AdminController is used to store event data fetched from the database, typically using the Event model. It represents a collection of event records that can be passed to a view for display or further processing. Each record in $events is accessed as $event to display details such as ID, title, and description.

The **AdminController** handles administrative tasks related to events, including creating, editing, updating, and deleting events. It interacts with the Event model to perform database operations and passes data to the admin views. The **dashboard()** method loads the admin dashboard, serving as the central hub for managing events. This controller ensures efficient, organized, and secure handling of admin functionalities in the application.

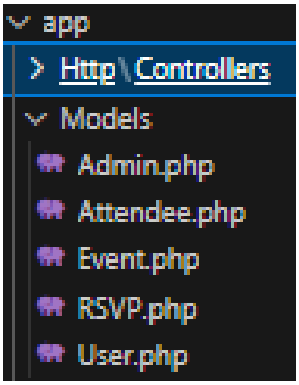
The **Event model** represents the events table in the database and manages event-related data. It uses the HasFactory trait for generating test data and specifies the fillable fields (title, description, event_date, location, category, user_id) to allow secure mass assignment. The model defines a one-to-many relationship with the Attendee model using the **attendees()** method.

Display Data in Views
resources/views/admin/AdminManageEvent.blade.php:

```
@extends('layouts.adminlayout')
@section('title', 'Manage Events')
@section('header', 'Event List')

@section('content')
    <table id="MEEventTable">
        <tbody>
            @foreach ($events as $event)
            <tr>
                <th>ID</th>
                <th>Title</th>
                <th>Description</th>
                <th>Event Date</th>
                <th>Location</th>
                <th>Category</th>
                <th>Action</th>
            </tr>
            <tr class="MEEventRow">
                <td>{{ $event->id }}</td>
                <td>{{ $event->title }}</td>
                <td>{{ $event->description }}</td>
                <td>{{ $event->event_date }}</td>
                <td>{{ $event->location }}</td>
                <td>{{ $event->category }}</td>
                <td>
                    <a href="{{ route('admin.editing', $event->id) }}" class="MEEditLink">Edit Event</a>
                </td>
            </tr>
            @endforeach
        </tbody>
    </table>
@endsection
```

In the **AdminManageEvent.blade.php** view, the **$events** data is passed from the controller, where **Event::all()** retrieves all events from the database. The view uses a **@foreach** loop to display each event's details (e.g., id, title, description) in a table. Each row shows event data, and the "Edit Event" link is dynamically created for each event, allowing the admin to edit it. This process ensures that the event data from the database is displayed and managed efficiently in the admin interface.

Models: In **app/Models/Event.php**, and **app/Models/Attendee.php**, etc. define relationships and fillable fields:



Event model: Defines a one-to-many relationship with the Attendee model and specifies the fillable fields for event data.

Attendee model: Defines a belongs-to relationship with the Event model and specifies fillable fields for attendee data.

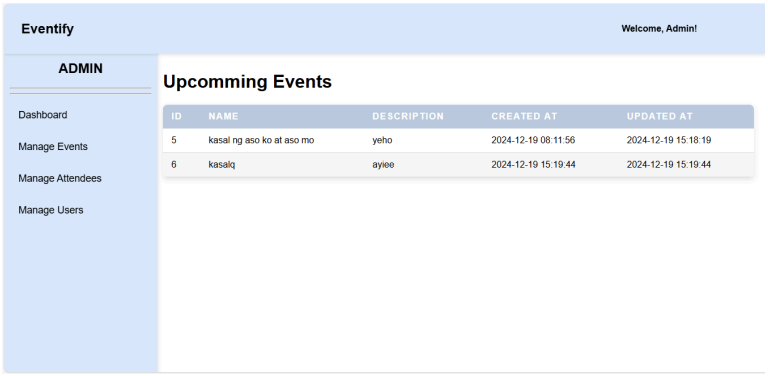
:

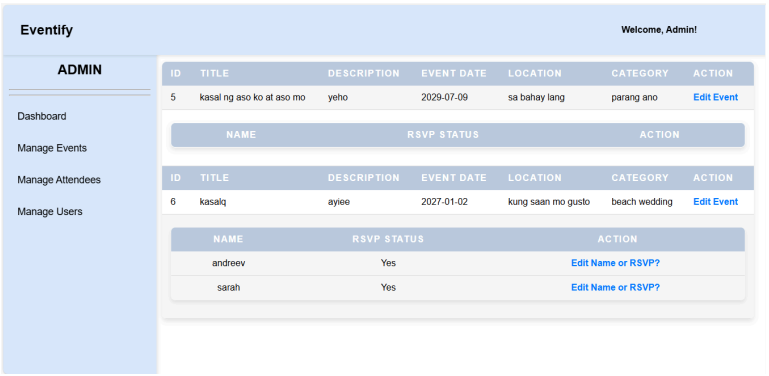
Event Management:

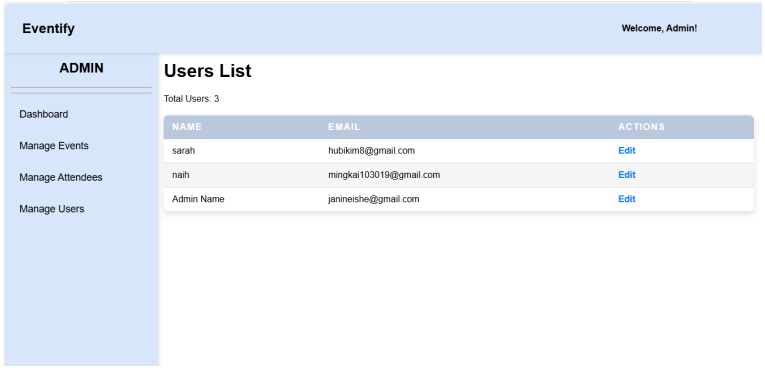
Eventify

- User Registration and Authentication
- Event Creation and Management
- RSVP System
- Event Search and Filtering
- Admin Dashboard

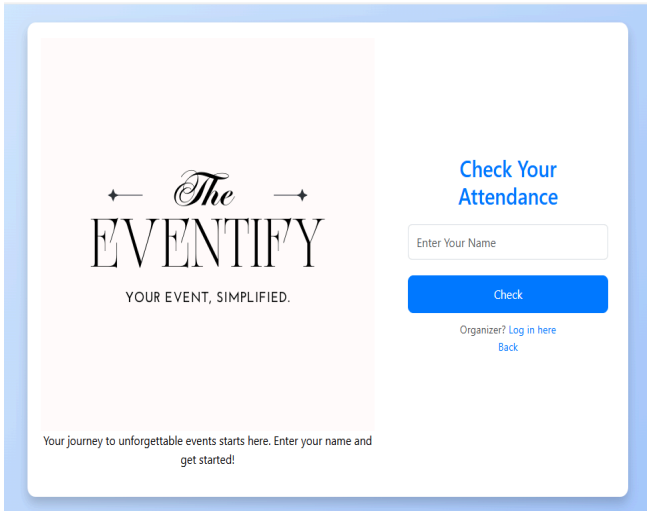
For Admin
Dashboard:

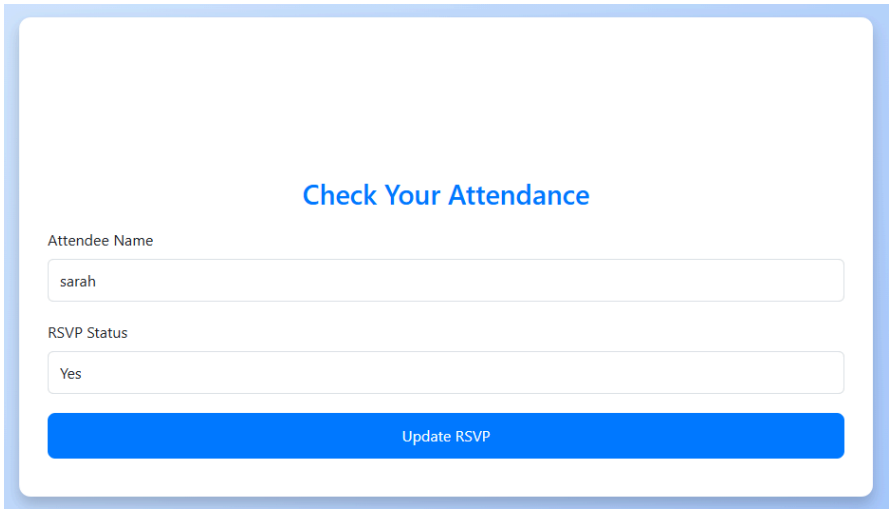


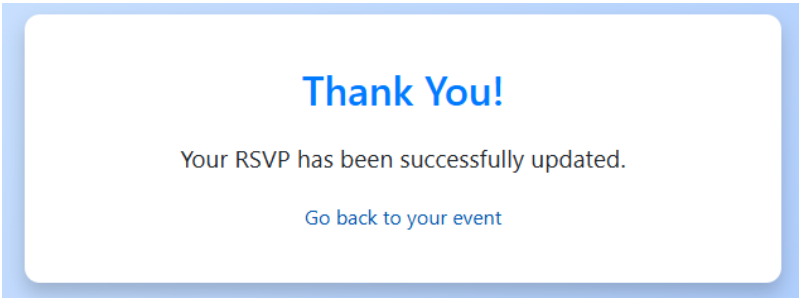




RSVP

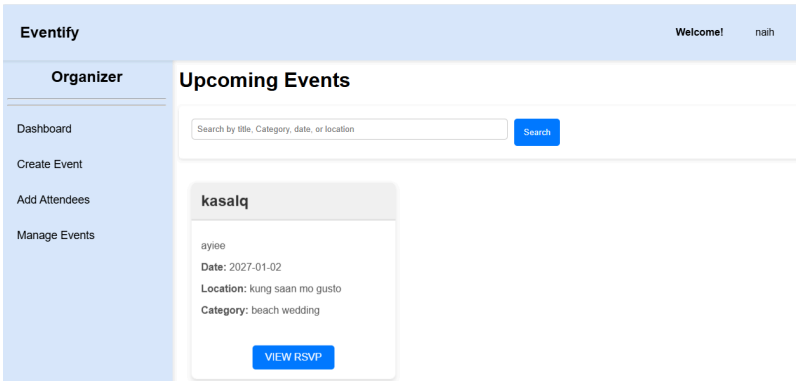


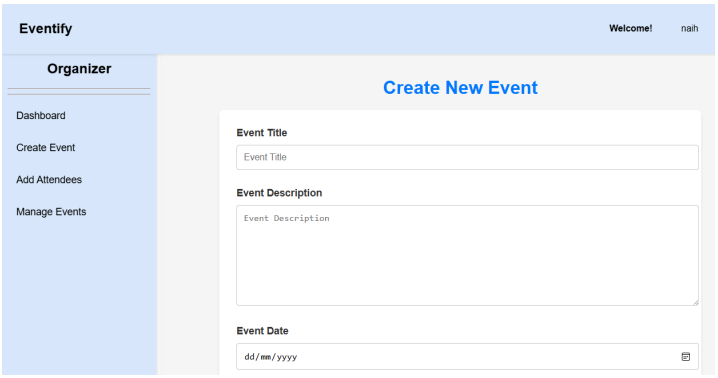


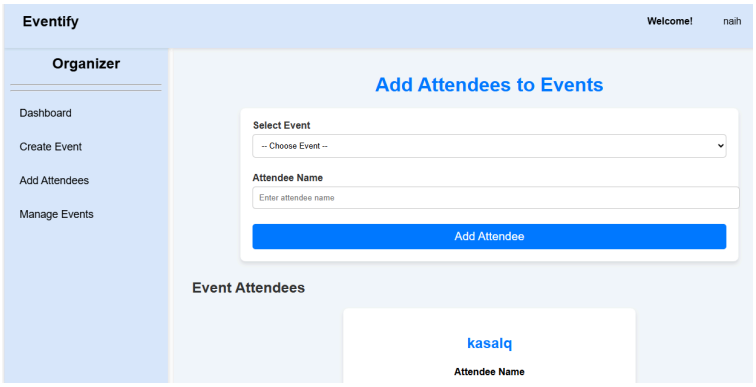


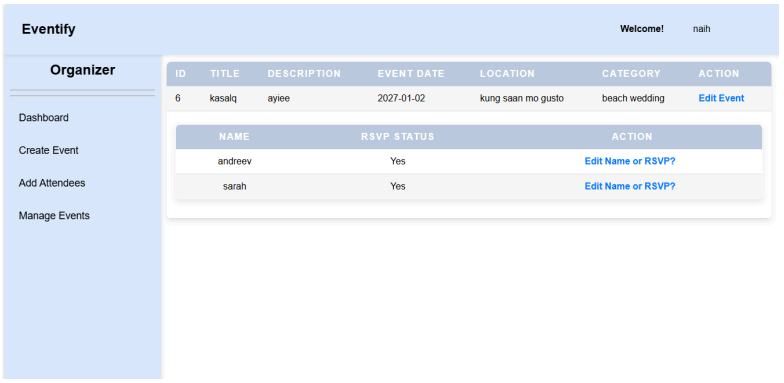
For Organizer

Dashboard:



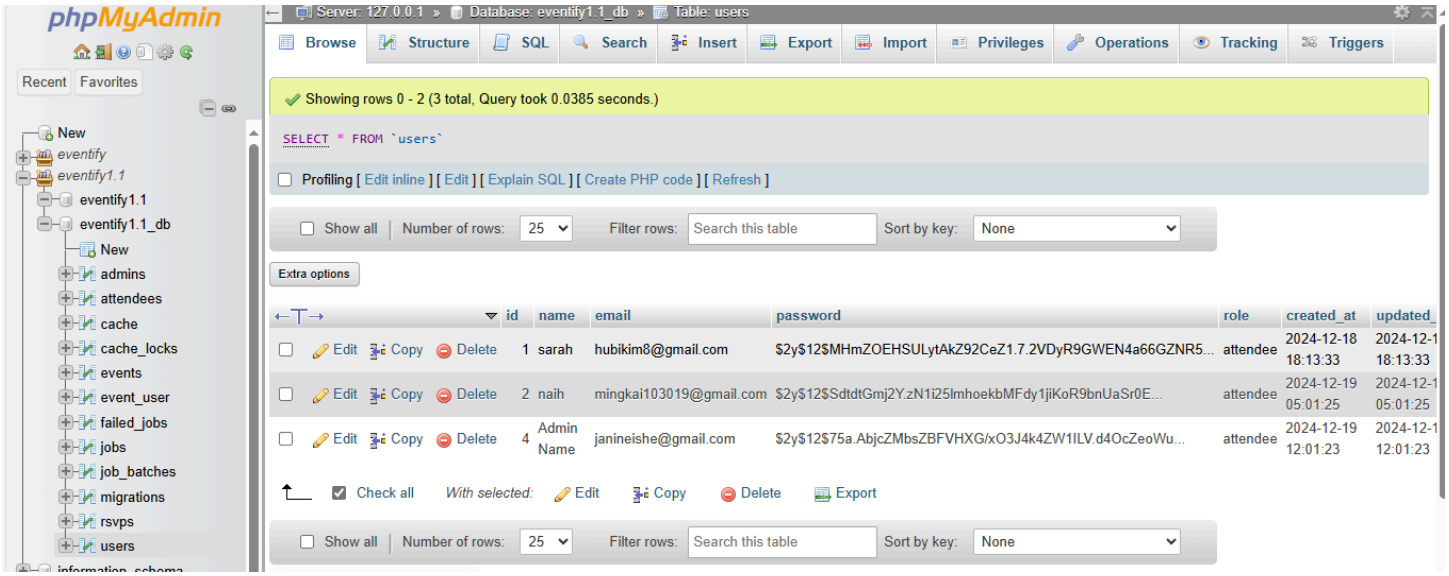




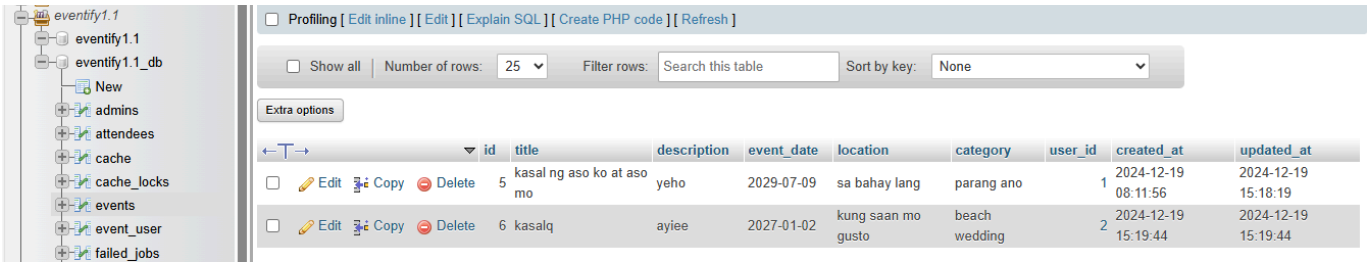


Database

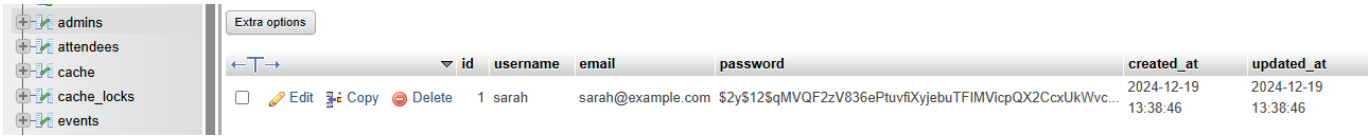
For users:



For events:



For admins:



For attendees: