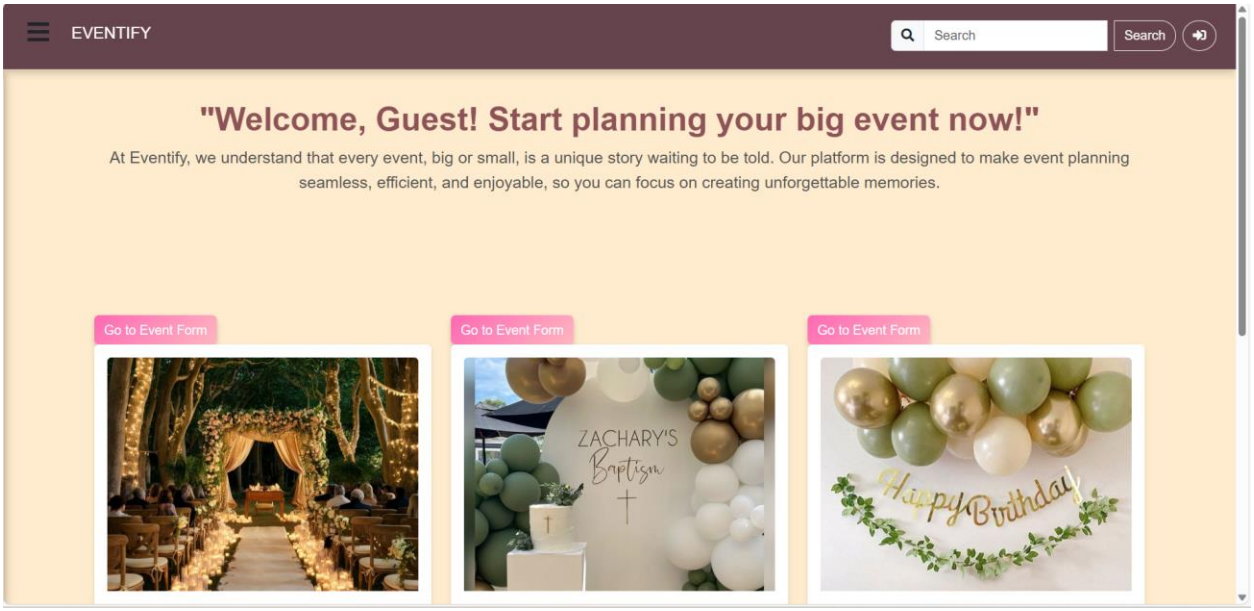
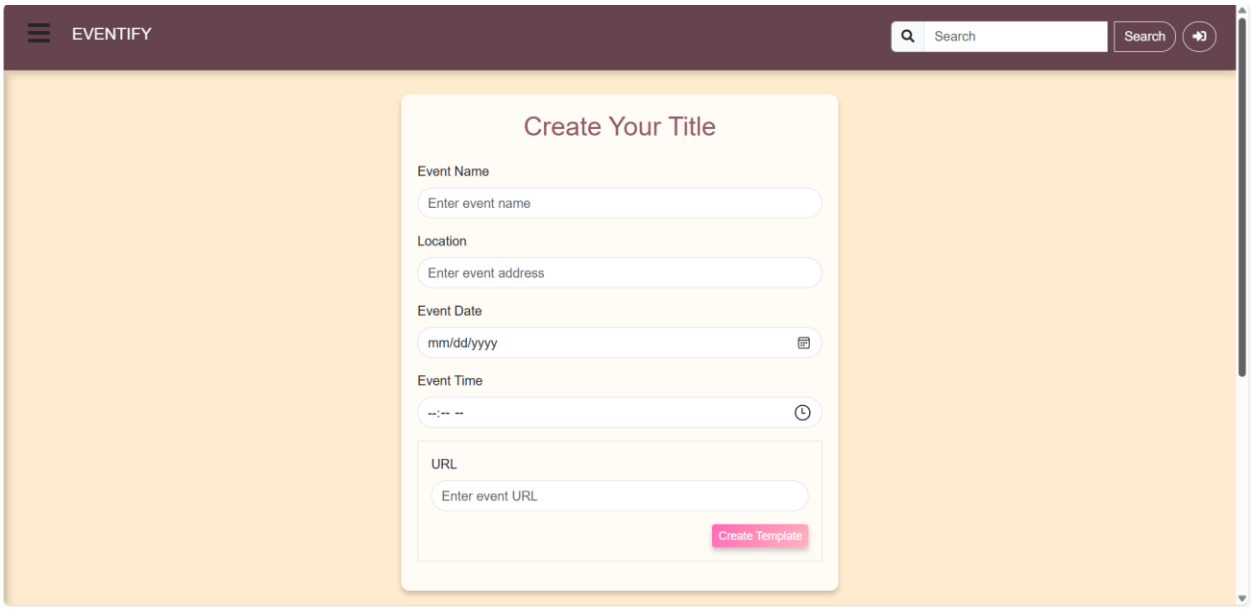
Jela Mae N. Gñilo **BSIT-3C**





HomeController.php

```php
class HomeController extends Controller
{
    public function home(Request $request)
    {
        // Fetch username from session or fallback to Guest
        $username = $request->session()->get('username', 'Guest');

        // Dummy data for events
        $events = [
            [
                'title' => 'Wedding',
                'description' => 'Plan the perfect wedding event.',
                'image' => asset('assets/images/wedding.jpg')
            ],
            [
                'title' => 'Birthday',
                'description' => 'Celebrate with an unforgettable birthday party.',
                'image' => asset('assets/images/baptism.jpg')
            ],
            [
                'title' => 'Graduation',
                'description' => 'Make your graduation day special.',
                'image' => asset('assets/images/birthday.jpg')
            ],

        ];

        // Return the homepage view with username and events
        return view('homepage', ['username' => $username, 'events' => $events]);
    }

    public function form()
    {
        return view('form');
    }
}
```

- This controller manages the display of the homepage with dynamic session data and static event content, as well as the rendering of a form page. This controller is responsible for handling requests to the homepage and a form page. The home method retrieves the username from the session using the $request->session()->get() method, defaulting to 'Guest' if no username is found. It also defines a set of dummy event data, each containing a title, description, and image URL generated with the asset() function. This event data, along with the username, is passed to the homepage view, where it can be displayed. The form method simply returns the form view, allowing users to access a form page.

Parameter Handling

- The parameter handling is managed within the home method using the Request object. The line $username = $request->session()->get('username', 'Guest'); retrieves the value of the username key from the session data. If the key is not present, it defaults to 'Guest', ensuring the application continues to function without errors.

RoleController.php

```php
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class RoleController extends Controller
{
    // Handle Organizer Login
    public function organizerLogin(Request $request)
    {
        $username = $request->input('username');
        $password = $request->input('password');

        // Fixed Organizer password check
        if ($password === 'organizer123') {
            // Redirect to the homepage with the username
            return redirect()->route('homepage')->with('username', $username);
        } else {
            // Redirect to error page if credentials are incorrect
            return redirect()->route('error.page');
        }
    }

    // Handle Admin Login
    public function adminLogin(Request $request)
    {
        $username = $request->input('username');
        $password = $request->input('password');

        // Fixed Admin password check
        if ($password === 'admin') {
            return view('admindashboard', ['username' => $username]);  // Pass username to view on success
        } else {
            // Redirect to error page if credentials are incorrect
            return redirect()->route('error.page');
        }
    }
```

- The RoleController handles login functionality for two roles: organizer and admin. In the organizerLogin method, it retrieves the username and password from the request inputs, checks if the password matches a fixed value ('organizer123'), and redirects the user to the homepage with the username in the session if successful. If the password is incorrect, it redirects to an error page. Similarly, in the adminLogin method, the password is checked against a fixed value ('admin'), and on success, the admindashboard view is returned with the username. If the credentials are invalid, the user is redirected to the error page.

Routes

```
// Home page for authenticated organizers
Route::get('/home', [HomeController::class, 'home'])->name('homepage');
Route::get('/user/form', [HomeController::class, 'form'])->name('form');
```

- The first route (/home) maps a GET request to the home method of the HomeController class. It is named 'homepage'. This route is likely intended for authenticated users (in this case, organizers) who will access the homepage.
- The second route (/user/form) maps a GET request to the form method of the HomeController. It is named 'form' and would render a form view. This route is typically used for displaying a form to the user, such as a registration or submission form.

```
// Role Routes
Route::get('/', function () {
    return view('user_role');
})->name('role');
```

- This routes handles the root URL (/). It maps the GET request to a closure that returns the user_role view. The view is likely designed to let users choose their role (e.g., admin or organizer) before logging in.

```
// Login actions for admin and organizer
Route::post('/user/admin', [RoleController::class, 'adminLogin'])->name('login.admin');
Route::post('/user/organizer', [RoleController::class, 'organizerLogin'])->name('login.organizer');
```

- These two routes handle POST requests for logging in as either an admin or an organizer. The first route (/user/admin) maps a POST request to the adminLogin method in the RoleController. This route is named 'login.admin' and handles the admin login process. The second route (/user/organizer) maps a POST request to the organizerLogin method in the RoleController. This route is named 'login.organizer' and handles the organizer login process.