Software Design Specification (SDS)

Project Title: Warima Events — Event Organizer Booking System

Version: 1.0

Prepared By: Warima Edgar 23/06030

Date: 10/4/2025

1. System Architecture Design

The system follows a typical web application architecture:

Frontend: HTML, CSS, and JavaScript for client-side interaction

Backend: PHP scripts handle form submissions, database queries, and session management

Database: MySQL is used to store users, events, and feedback reports

Hosting: Localhost environment

2. Component Design

2.1 Authentication Module

Handles user login and registration

Stores session data and user credentials

2.2 Event Booking Module

Captures event details from users (event type, location, date, etc.)

Submits data via POST to backend PHP endpoint

Stores events in events table with status tracking

2.3 Feedback/Report Module

Allows users to submit issues or suggestions

Captures IP address and browser information

Stores data in reports table

2.4 Admin Module

Displays total users and events

Loads stats using AJAX from admin_dashboard.php

Links to manage users and events

2.5 User Management

Admin can suspend or view users

Actions performed through manage_users.html and backend PHP endpoints

2.6 Event Management

Admin can approve or reject pending event requests

Interface found in manage_events.html

3. Interface Design

3.1 Web Interfaces

login.html: Login form with email/password input

registration.html: Registration form for new users

dashboard.html: User landing page with call-to-action and feedback section

event_form.html: Interactive form for booking events admin_dashboard.html: Overview of stats and quick admin links

confirmation.html: Shows success message after booking

3.2 Backend Interfaces

submit.php: Handles booking and feedback submissions

admin_dashboard.php: Returns stats for admin via JSON

get_users.php / update_user.php: Admin user management endpoints

get_events.php / update_event_status.php: Event approval management

4. Data Design

4.1 Database Tables

Users

Field	Type	Description
id	INT	Primary Key
name	VARCHAR	User's full name
email	VARCHAR	Unique email address
password	VARCHAR	Hashed password
role	ENUM	'user' or 'admin'
created_at	TIMESTAMP	Date of registration
suspended_until	DATETIME	Suspension expiry
updated_at	TIMESTAMP	Last updated
deleted_at	TIMESTAMP	Soft delete flag

events

Field	Type	Description
Id	INT	Primary Key
user_id	INT	FK to users table
event_type	VARCHAR	Type of event
location	VARCHAR	Event location
event_date	DATE	Scheduled date
guests	INT	Number of attendees
notes	TEXT	Additional notes
created_at	TIMESTAMP	Time of booking
status	ENUM	pending/approved/rejected

reports

Field	Type	Description
id	INT	Primary Key
user_id	INT	FK to users table
report_text	TEXT	User's feedback or issue
submission_date	DATETIME	Date submitted
status	ENUM	reviewed/pending
ip_address	VARCHAR	IP address of submitter
user_agent	TEXT	Browser/device info
created_at	TIMESTAMP	Timestamp of entry |

5. Algorithm Design

5.1 Event Submission

Validate all input fields

Insert form data into events table with status = 'pending'

Redirect user to confirmation page

5.2 Report Submission

Capture textarea content, user ID, IP, and user-agent

Insert into reports table with status = 'pending'

Return success/failure JSON

5.3 Admin Stats Fetch

Query COUNT(*) for users and events

Return data as JSON

Update admin dashboard stats

5.4 Admin Approval

List events with status = 'pending'

Allow admin to update status

Reflect updates on interface with AJAX

6. Security Considerations

Use PHP sessions to protect authenticated pages

Input sanitization using htmlspecialchars() or prepared statements (PDO)

Passwords must be hashed using password_hash()

Admin panel access is restricted to role = 'admin'

7. Error Handling and Logging

Use try-catch blocks for DB interactions

Display user-friendly error messages

Log server-side errors in a secure file (optional enhancement)

8. Performance Considerations

Use indexes on foreign keys and frequently searched fields (e.g., user_id, status)

Minimize data returned via AJAX

Cache statistics if needed (optional for future scaling)

9. Testing Strategy

Manual testing for form validation and session handling

Simulate failed submissions and invalid logins

Admin test cases: approving/rejecting, suspending users, checking stats

10. Deployment Strategy

Initially hosted locally

Can be deployed to shared hosting or VPS with Apache, PHP, MySQL stack

Environment config for DB credentials should be moved outside public folder

11. Maintenance and Updates

Plan to support updates like:

Email notifications

Event search/filter features

Feedback response system for admin

Source code can be versioned using Git