# 24-T1 OWL JUDGE

## Software Design Document

CS 4850 – Senior Project
Sections 02 & 03, Spring 2025
February 01, 2025

**Project Team**:

| **Name** | **Role** | **Contact** |
|---|---|---|
| **Lumiere Nathan Kue** | Documentation, Group Leader | 404-271-1584 |
| **Nathanael Fokou** | Developer | 404-663-5803 |
| **Bill Roan Simo** | Developer | 404-493-0240 |
| **Minh Quang Vu** | Developer | 678-579-7824 |

**Advisor / Instructor: Sharon Perry**
**Role: Facilitate project progress; advise on project planning and management.**
**Contact: 770-329-3895**

## Table of Contents

# 1. Introduction

## 1.1 Document Outline

This document serves as the Software Design Document (SDD) for the OwlJudge Web Application, an event judging platform designed to facilitate the Computing Showcase (C-Day) event at Kennesaw State University (KSU). The document outlines the system's architecture, design decisions, constraints, and implementation strategies.

## 1.2 System Overview

The OwlJudge Web Application is a cloud-hosted, multi-user, role-based platform that enables a structured, digital approach to evaluating capstone projects.

The current manual system is prone to errors, delays, and inefficiencies due to its reliance on paper-based processes. OwlJudge replaces this with a real-time, automated, and scalable solution.

Key features of the application include:

• Secure user authentication for administrators, judges, and participants.
• Event management features that allow the creation, modification, and assignment of judges to projects.
• A digital scoring system for standardized evaluations.
• Real-time ranking and analytics for event administrators.
• A cloud-hosted, scalable architecture that ensures high availability.

# 2. Design Considerations

## 2.1 Assumptions and Dependencies

• The system will be hosted on AWS Free Tier or Azure Student Account for cost efficiency.
• Users will access the system via modern web browsers and mobile devices using React Native.
• A stable internet connection is assumed for all users.

• The system will use PostgreSQL as the primary database and Redis for caching.
• All event-related data (projects, judges, scores) will be provided before each event.
• The system will comply with FERPA and university IT security policies.

### 2.2 General Constraints

• Security Compliance: Authentication and authorization must follow OAuth 2.0 and JWT best practices.
• Scalability: The system must support at least 100 concurrent users with minimal latency.
• Data Storage & Retention: Audit logs and event data must be retained for at least one semester.
• Performance: Score submissions and data retrieval must occur within 2 seconds even under peak load.
• Cross-platform support: The UI must be responsive and mobile-friendly to support various screen sizes.

### 2.3 Development Methods

The Agile Development Methodology is adopted, featuring:
• Sprint-based iterations to ensure incremental development and testing.
• Frequent usability testing to refine the judge's scoring experience.
• Continuous Integration/Continuous Deployment (CI/CD) to deploy updates efficiently.

## 3. Architectural Strategies

To meet scalability, security, and performance requirements, the system adopts a Microservices Architecture deployed on AWS Free Tier.

Key strategies include:

• **Frontend:** Developed using React Native for cross-platform compatibility (web and mobile).
• **Backend:** Built using Node.js (Express) or Python (FastAPI) for high-speed processing.
• **Database:** PostgreSQL for structured data storage and Redis for caching frequently accessed data.

- **Authentication:** Implemented using JWT and OAuth 2.0.
- **Performance Optimization:**
  - Load balancing to handle multiple concurrent requests.
  - Redis caching to speed up frequent database queries.

## 4. System Architecture

The system is divided into five main components:

**1. User Management Service:** Handles user registration, authentication, and authorization using JWT.
**2. Event Management Service:** Allows admins to create and manage events, assign judges, and track progress.
**3. Project & Judge Assignment Service:** Ensures balanced and fair assignment of judges to projects.
**4. Scoring System:** Provides an interactive UI for judges to submit scores and feedback in real time.
**5. Reporting & Analytics Module:** Generates reports and real-time visualizations for event progress tracking.

### 4.1 Subsystem Architecture

- **Frontend:** Developed using React Native, offering a mobile-first, responsive UI.
- **Backend:**
  Node.js (Express) or Python (FastAPI) handles REST APIs.
  JWT-based authentication ensures security.
- **Database:**
  PostgreSQL stores user, event, and score data.
  Redis optimizes performance by caching frequently accessed queries.

## 5. Policies and Tactics

- **Security:** All data in transit will be encrypted using TLS 1.2+.
- **Data Protection:** Passwords will be hashed using bcrypt or Argon2.
- **Error Handling:** Logs will track all system actions for debugging and audits.

# 6. Detailed System Design

## 6.1 Classification

The system is modular and follows a microservices architecture.

## 6.2 Constraints

- Must process 100+ concurrent users efficiently.
- Must ensure data integrity and real-time score aggregation.

## 6.3 Resources

- Database: PostgreSQL & Redis
- Hosting: AWS Free Tier
- Security: OAuth 2.0, JWT authentication

## 6.4 Interface/Exports

The system exposes RESTful APIs:
- /api/auth/ → User authentication and authorization
- /api/events/ → Event creation and management
- /api/scoring/ → Judges submit scores
- /api/reports/ → Event results and analytics

# 7. Glossary

- C-Day: Computing Showcase event at KSU.
- JWT: JSON Web Token for authentication.
- OAuth: Open standard for secure API authorization.
- Redis: In-memory caching database for performance optimization.
- PostgreSQL: SQL-based relational database management system.

## 8. References

1. Software Requirements Specification (SRS) – OwlJudge Web Application
2. FERPA Compliance Guide – Kennesaw State University
3. AWS Free Tier Documentation
4. PostgreSQL Documentation