ALT-CS Events Team Specification Doc

Iain Cline, Jake Kenslow, Alex Whipple | doc version a.6.0

# Introduction

Hack K-State uses a Django web app maintained by Dr. Weese to manage the event. Game Jam could use a similar website to run the Game Jam. Other clubs may want to run large events that could benefit from a website like Hack's. This project's purpose is to provide clubs and organizations a straightforward way to develop event web apps with Django. The web apps will provide all necessary functions to run events including event setup, registration, information, and administration. This document sets a mutual understanding between developers and stakeholders on what problem is being solved and how it is being solved.

# Project Overview

Factors affecting the development of our product are the needs of teachers and students as they are our major stakeholders of our product. For example, the goal is if a club needs a specific feature added to the application that would affect the development process of our app. The client of our product in theory should be universities that need a mutable website for setting up clubs and events. Currently our clientele is the K-State computer science department. Stakeholders for our project are Dr. Weese head of K-State hackathon, Dr. Bean head of K-State Game Jam, and presidents of clubs at K-state. There are two primary users of our application: those who registered for events and those who manage the events. The problem our application tries to solve is to create a modular application for different clubs to websites with their own customized features. The main features for our proposed application are student registration, event registration, event management, authentication, event information, and an event organizers dashboard. Our most important constraints are the needs of club organizers and users registering for the events.

# Devlopment and Target Environments

This project is being developed containerized with docker using poetry package management. This way the project can be simultaneously developed by the team members on either different machines, or different GitHub Codespaces, without worrying about redownloading or installing dependencies. Any databases or authentication servers used in development are separate from production servers and are meant specifically for development purposes.

The environment the project is currently targeted to be maintained in is the ALT-CS system repositories as a containerized package once it is completed in production. The over-arching main goal is to treat this as a Django package that can be installed in Django’s CLI and KSU CS event developers will be able to build their applications from.

# System Model

Each event will be its own Django web app. For example, Hack and Game jam will be separate Django web apps in their own repos. Django packages will provide common functionality across web apps. For example, registration, auth, informational pages, event countdown are all common functionality in the base package. Web apps will only import packages they need such as MLH Oauth or Discord integration. The Django packages and apps will follow the standard Django project structure and adhere to Django best practices regarding storing configuration in Django settings. Django allows any relational DB to be used, and we have chosen Postgres. The package architecture described is illustrated in the appendix as Figure 1.

# User Interaction

Main page

Should have login button that logs user in using an authentication method

Login button should change to logout button when logged in. When clicked, it logs user out.

Should have event countdown timer that displays time until next event to user

Should have appropriate navigation links to other sections: calendar, organizer dashboard (if organizer) and registration (if not registered)

Should have event description and information provided to user.

Organizer dashboard

Should allow organizer to see list of all users registered to an event and relevant information such as emergency phone numbers, email, relevant health information, and specific needs. User information should be editable or removable.

Should allow organizers to see time and location of events and the ability to edit information of said events.

Should allow organizers to edit and manage subevents within the events.

Should allow for users to see check in list of registered users.

Should allow for organizer to compile CSV data of all user and event information.

Registration page

Should collect registration information from user: name, birthday, school affiliation, email, phone.

Should allow user to submit information when necessary fields filled

Calendar

Should show any information set by event administrators.

# Functional Requirements

For both our functional and nonfunctional requirements, we spent time gathering information and requirements from our stakeholders, which include Dr. Josh Weese and Dr. Nathan Bean.

The functional requirements for this project should include:

* Providing an event developer package framework to build an event website from.
* Allowing event developers to choose their authentication system of choice (i.e. OAuth, CAS, etc.) for the web application.
* Provide means for an event database to keep and store event data and allow for multiple events and sub-events to occur.
* Functionality for event developers to add their own assets for their web applications, i.e. HTML and CSS templates.
* Dashboard for the event organizer for adding events, maintaining event goers, and viewing historical stats and information about past events.

These requirements can be checked if they are met by use of Django unit and integration testing. For example, unit testing can be used to see if an event or user model has been correctly generated and added to a test database, and integration testing can be used to test if the general functionality of the program still works when new features are added to the project.

# Nonfunctional Requirements

These requirements, also determined from our requirements gathering with Dr. Weese and Dr. Bean, are extra features that we think could be beneficial for the events package. These requirements include:

* Project is fully containerized and is able to run and compile in some capacity on first launch of the container.
* Event registration options such as QR generation for scanning into events.
* Options for discord integration for the event website.
* Event countdowns and timer clocks.

These nonfunctional requirements can be tested similarly to other features with integration testing to check if the core functionality of the project will still work correctly.

# Semester Goals

Minimum viable product:

Hack and game jam website based off common package. Needs auth, registration, event administration, informational sections, organizer dashboard. A demo app for future developers to see how the package is implemented. Django apps are built with the standard project structure and are extendable as any Django app. Apps should be extendable to future features.

Enhanced version:

Features of an enhanced ideal version include: QR code scanning, discord integration, countdown timer, past and future event viewing, analytics dashboard, multiple auth methods, minor accounts. These features would be part of the packages or as their own package (discord integration).

# Appendices

Figure 1A diagram of a company

Description automatically generated