Senior design conference evaluation criteria

- Technical accuracy
- Creativity and innovation
- Supporting analytical work
- Addresses project complexity
- Completeness
- Design and analysis of tests
- Quality of responses during Q&A
- Organization
- Time allotment
- Visual aids
- Confidence and poise

Changes from Design Review

- 12 minutes to present instead of 10
- Omit the conceptual model section
- Add a project demo
- Spend less time explaining use cases and activity diagram
- Add a testing procedure section
    - What testing you have done, what remains to be done
- Condense risk analysis and development timeline
    - Obstacles encountered (risk) and work left to be done (development timeline)
- Include lessons learned section in conclusion
    - 2-3 things about SE you've learned from the project

Introduction

This report details the various design aspects of the Alumni Engagement Recording System for Santa Clara University. These aspects include the several use-cases of the system, conceptual models, technologies chosen, and architectural design.

This design has three main objectives: create an event calendar with weighted event display, have interactive events for engagement data, and provide a system for SCU administrators to approve and edit events and to view collected data from the interactive events.

The first objective is to create an event calendar for alumni events that displays events with a weighted system in which more important events are displayed first. We plan to achieve this by automatically assigning weights to events based on several factors such as whether the event is official, if the event is recurring, and the type of event. SCU administrators will also be able to modify this weighting system as they seem fit.

The next objective is to have the events on the calendar be interactive to record engagement data. This includes attendance records and satisfaction scores after the event is over.

Lastly, SCU alumni will be provided with a system in which they can approve, edit, or remove events on the alumni calendar. This system will also provide access to the data collected from the interactive events.

This report details all the design choices we have made regarding these objectives.

Requirements

There are a multitude of requirements that our system must meet to be considered fully-featured and functional. The two sections of requirements are user-interface requirements and administrative control requirements.

The system must provide several functionalities to users. Users must be able to view upcoming alumni events chronologically with significant events shown first. Users must also be able to engage and interact with the events by indicating plans to attend, giving satisfaction scores after events, and requesting email reminders for events. Lastly, this must be achieved without the need for user logins. Users of this system should be able to use all the functionalities by only providing their email address.

The system must also provide a framework for SCU administrators to modify and approve events that show up on the alumni event calendar. Administrators need to be able to change event descriptions and information and approve and post unofficial events that are submitted by SCU alumni. This administrative tool will also need to be able to give all of the data recorded by the interactive events so that administrators can analyze and learn about user engagement.

Use cases

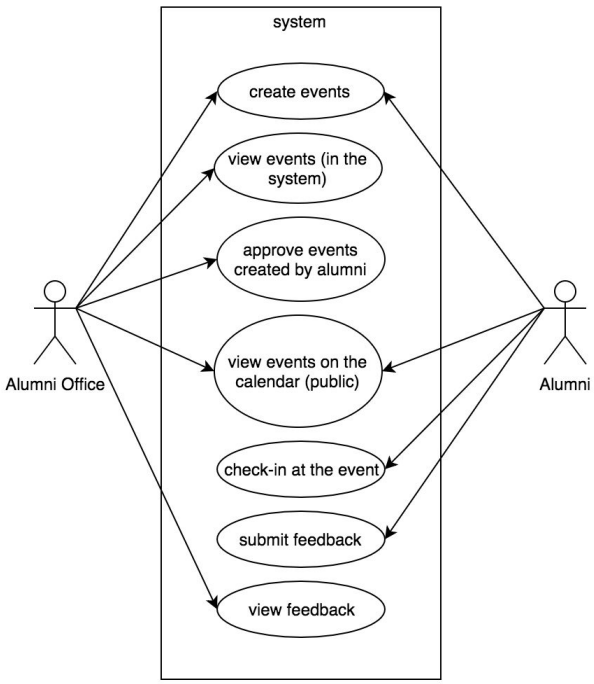


Figure 1: Case diagram of the system

With the system, the Alumni Office is able to create events and approve events created by alumni, while the alumni can use the system to create their own hosted events.

For the events that is created by the admin, it will be on the calendar automatically after creation. For the events created by the alumni, they will be on the calendar if they are approved by the admin.

For all events, the alumni can check-in using the system and submit feedback afterwards.

The admin can then use the feedback collected to decide what kind of events are more popular and therefore hold more of this kind of events.

Activity diagram

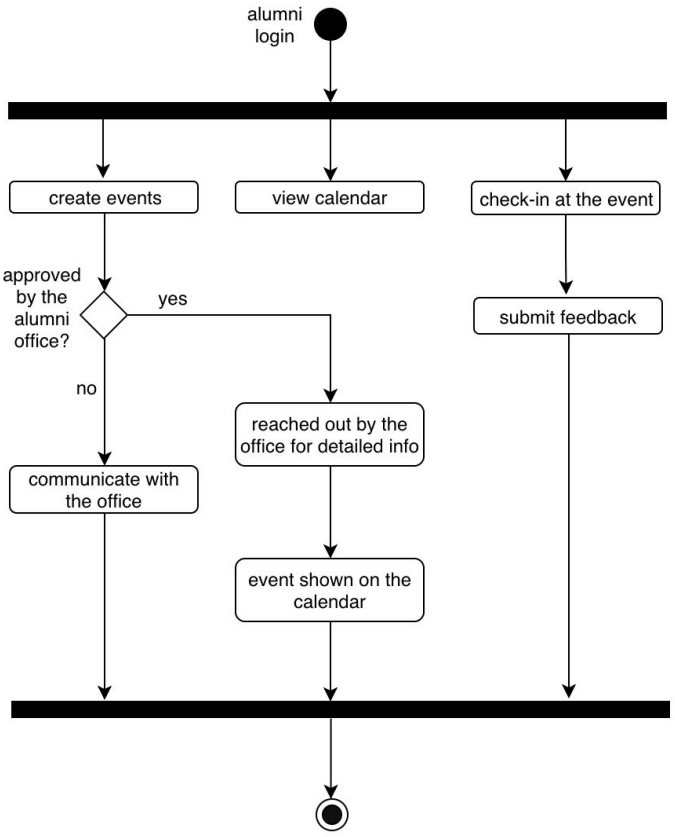


Figure 2: Activity diagram for alumni use

This activity diagram is what the alumni can do with the system.

Through the system, the alumni can create their own events as well as view the event calendar. When the events that the alumni create is approved by the admin, the admin will reach out to the alumni for more detail about the events. After that, the events will be posted on the calendar. If the events are rejected, the alumni can choose to communicate with the office privately to discuss.

When the alumni attend any event, they can check-in at the event using the system, and submit feedback afterwards.

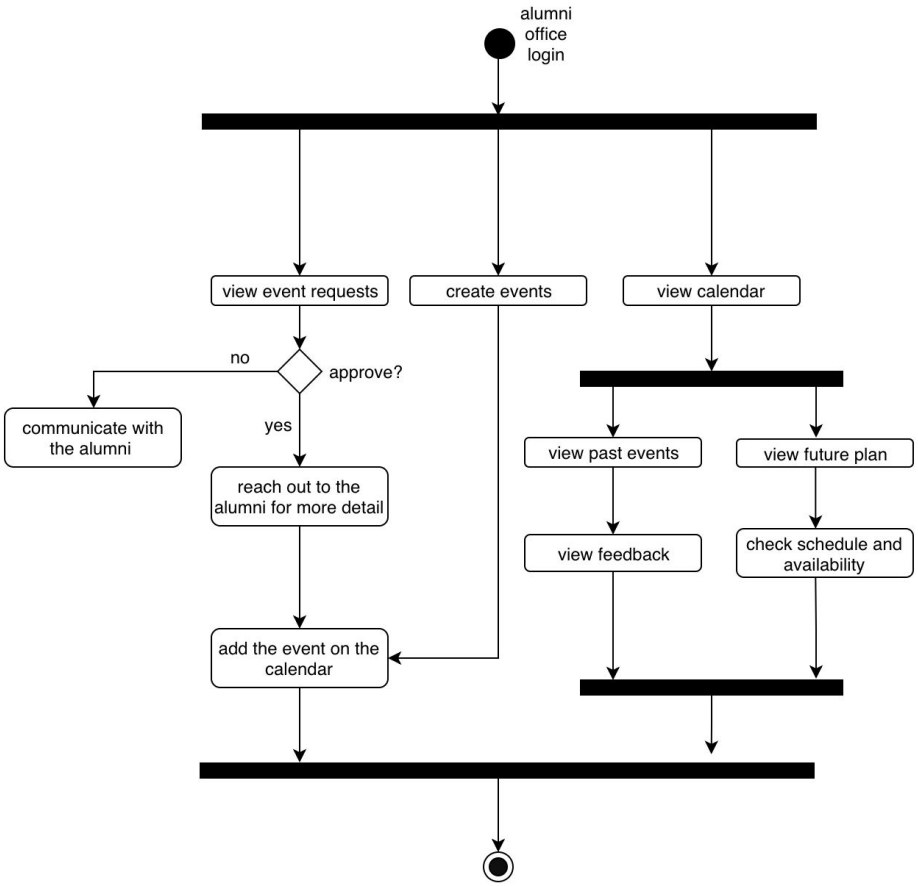


Figure 3: Activity diagram for alumni office use

This activity diagram is what the alumni office can do with the system.

Through the system, the admin can view event requests by the alumni, view the event calendar, and create official events.

When the admin decides to approve certain event, the admin will reach out to the alumni who created it for more information about the event. When both sides agree on all details about the event, the admin can then add the event on the calendar. When the admin decides to reject an event, they can communicate with the alumni privately.

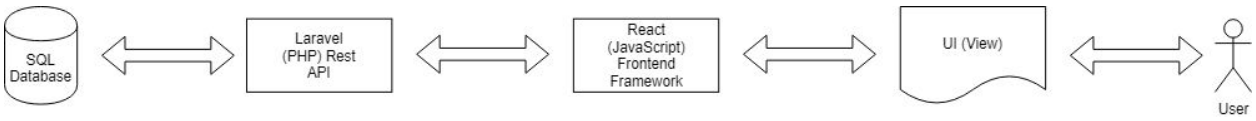
By viewing feedback of the past events on the calendar, the admin can learn what kind of events is more popular or immersive and then hold more of these events in the future. By view the calendar for future plan, the admin can check availability and therefore avoid conflict while adding events to the system.

Technologies used

Since we will be utilizing the Design Center to host our website, we do not have many technology choices to choose from. We will be using a SQL database and MySQL for database management since these are the only database technologies that the Design Center utilizes. For the backend of the server, we are forced to use the Apache server and PHP backend language that are already configured to work for the Design Center. We will incorporate a PHP framework, i.e. Laravel, to make backend development more structured and easier to implement.

The backend will server as an internal API for the frontend to query. The frontend of the website will be a single-page application implemented using a JavaScript framework, namely React. In addition, we will be using the Bootstrap framework to help style our website. Finally, we will use Webpack as our frontend asset manager to simplify the frontend development process.

Architectural diagram



Design rationale

We are forced to use SQL, MySQL, Apache, and PHP to implement the backend because other technologies are not supported by the Design Center. We are using the Laravel PHP framework to significantly simplify and bolster backend development, keep our backend code clean and easy to understand, and ensure that our internal API is REST-compliant. We are implementing an internal REST API and SPA frontend so that the initial load time for our website is fast, as many users will probably load the website over a mobile connection weekly or biweekly. We chose to use the Bootstrap frontend framework to make our site's UI mobile-friendly without having to waste time writing our own mobile-friendly code and styles. Finally, we decided to use Webpack as our asset pipeline manager so we can write modular and loosely-coupled code without having to worry about writing HTML code to reference all of the source files we create.

Testing Procedure (What testing you have done, what remains to be done)

Levels of testing

- Unit(single methods, procedures, modules, …)
- Functional (multiple units as a functional unit)
- Component (multiple functions)
- Integration/system (all components together)

Types of testing

- Acceptance (by clients before accepting)
- Conformance (product meets standards required)
- Configuration (different platforms)
- Performance (quantitative non-functional requirements)
- Stress (heavy loads to avoid catastrophic failure)

Other testing after unit testing

- Interface testing
    - Interfaces exposed by modules
- Usability testing
    - User satisfaction
- Robustness testing

How to generate test cases

- Intuition
- Specification
    - Black box

        - Based on specification
        - Ignores any information about code
        - Check whether a set of inputs produces the desired output
- Code
    - White box
        - Based on code
    - Grey box
        - Based on both specification and code
- Existing test cases
    - Regression
- history

Project demo

Obstacles encountered (condensed version of risk analysis)

Tasks left to complete (condensed version of development timeline)

Lessons learned