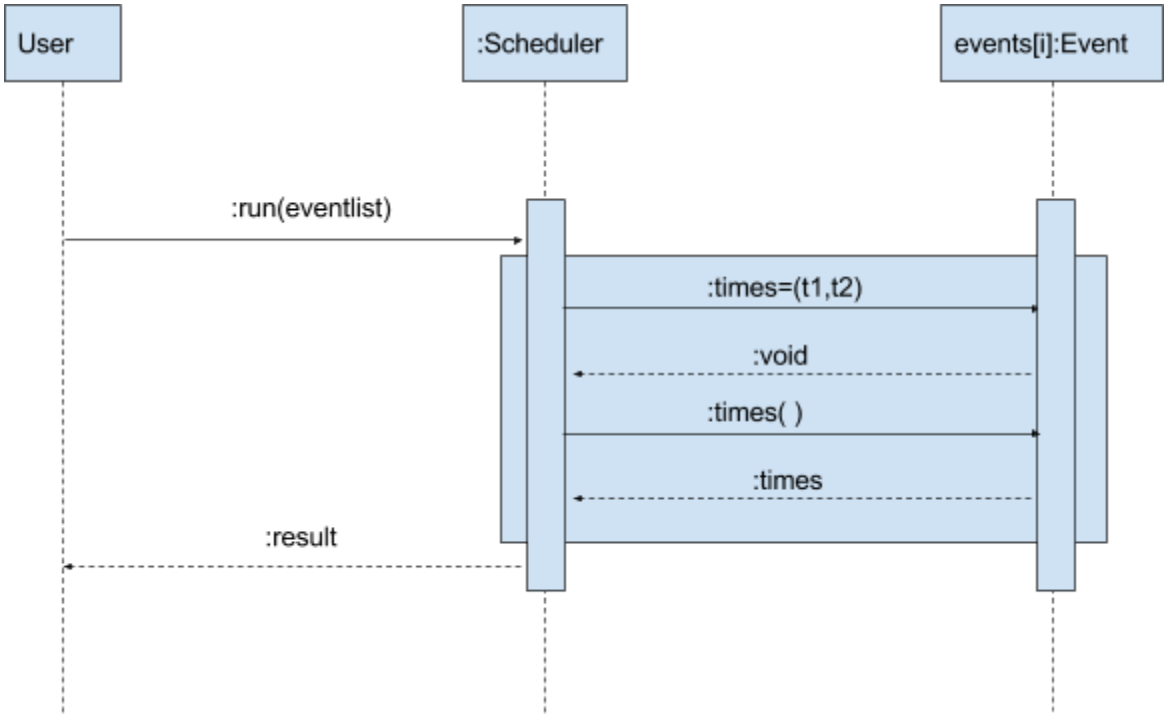
STAKEHOLDER REVIEW #1

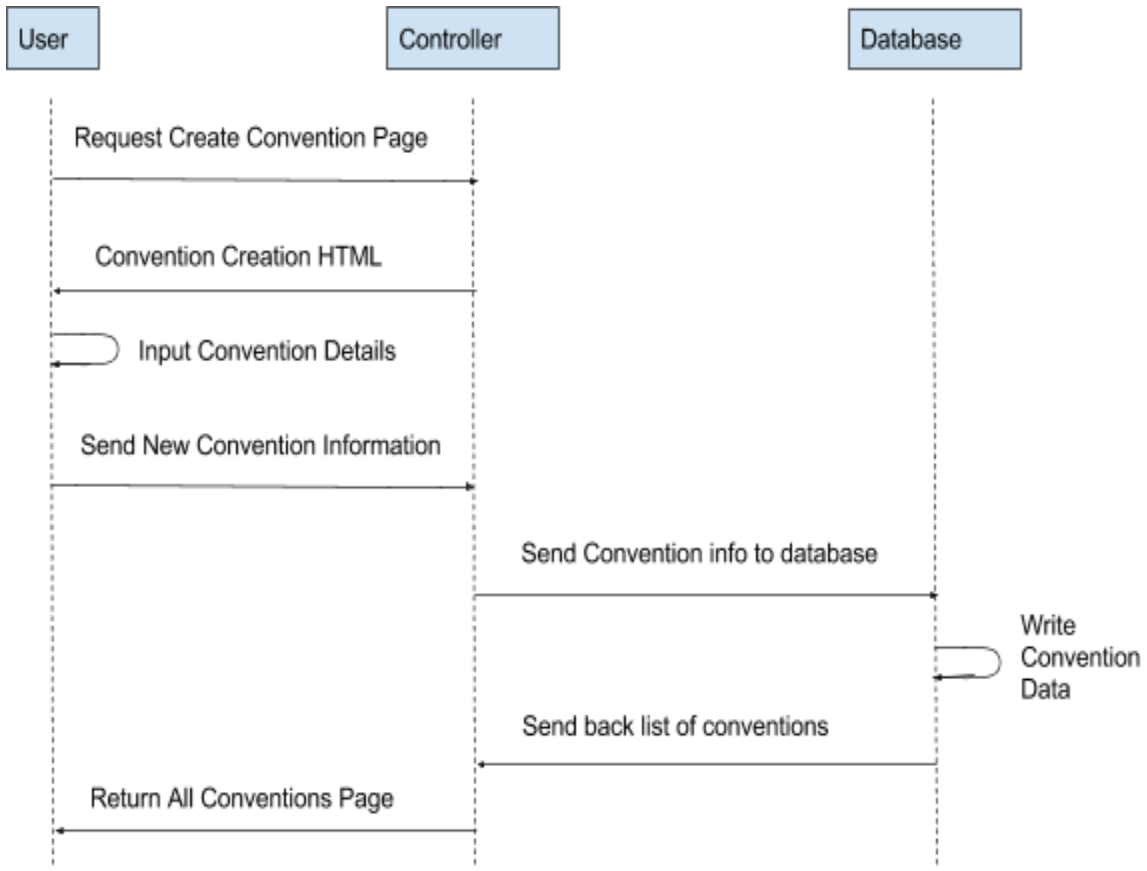
Lake Mountain Leprechauns

SEQUENCE DIAGRAMS

Use Case #1: GenerateSchedule

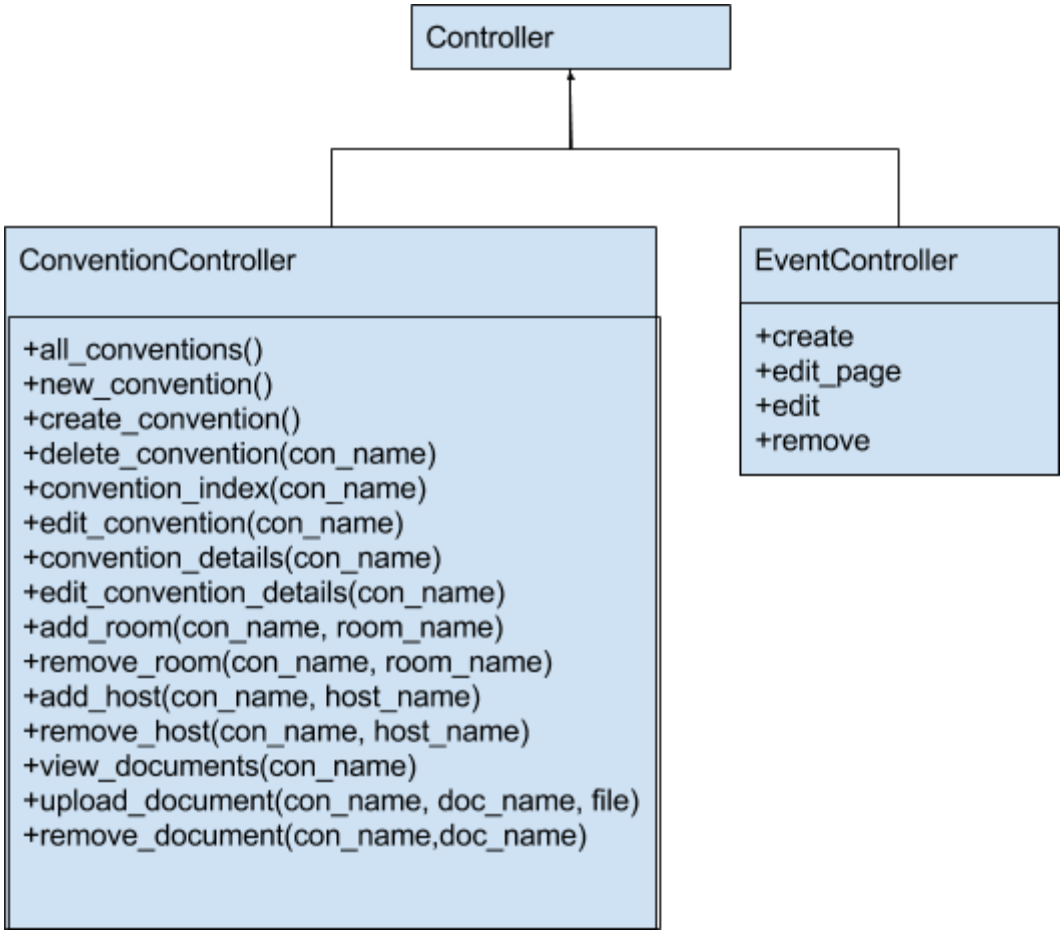


Use Case #2: CreateConvention

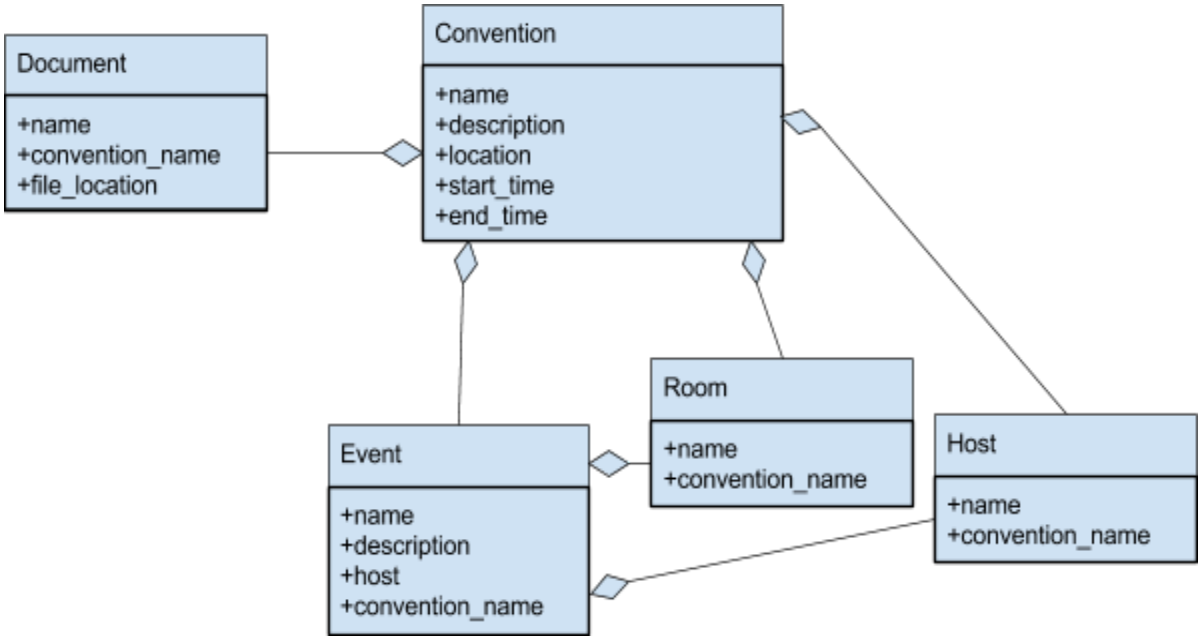


STATIC CLASS DIAGRAMS

Controllers

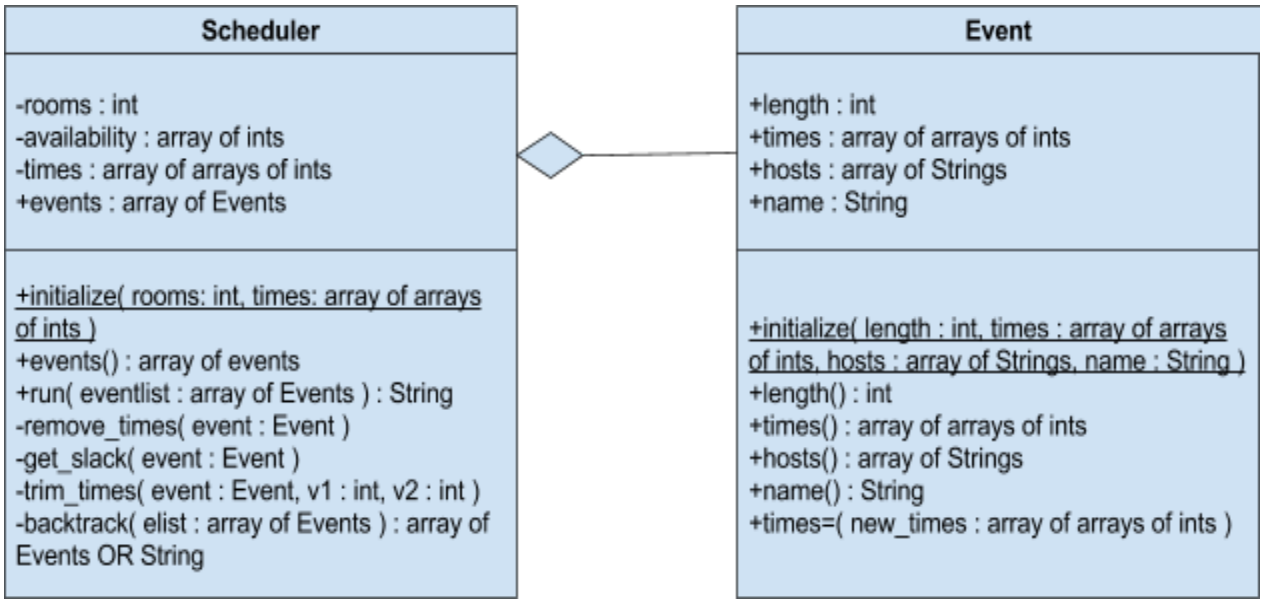


Database Objects



Scheduler

(Note: Scheduler not yet integrated with database and controllers and as such has some independent structuring.)



CRC CARDS

CONVENTION

Contains: *Convention Details* - Name, description, location, date & time, rooms, hosts, tags; *Convention Assets* - Events, documents, schedule

Responsibilities: Acts as an umbrella class storing information used in collaborators.

Collaborators: Event, Document, Schedule, Scheduler

DOCUMENT

Contains: Name, attributed Convention name, file location

Responsibilities: Handles misc. info file uploads that an Organizer wants to share via this product.

Collaborators: Convention

EVENT

Contains: Name, attributed Convention name, hosts, tags, times, length

Responsibilities: A data “nugget” used by the scheduling function; has useful constraints for this utilization. Has attributed name, times, tags, and hosts.

Collaborators: Convention

SCHEDULER

Contains: Data structures tracking Event availability

Responsibilities: Takes in Events and produces Schedules via a heuristic algorithm that uses rooms, times, hosts, and event length as constraints.

Collaborators: Events, Schedule, Convention

DESIGN APPROACH

This product is produced with deliberate object oriented design. Its end goal is to provide efficient and organized information collection and dissemination from a convention Organizer to any number of convention Attendees. This information includes Events, Documents, and a Schedule, all of which may be of varying quality and quantity. As such, the software approach to this product includes classes that allow for dependency inversion and that implement carefully limited and articulated coupling among them.

Architecturally, this software makes use of a Convention class, which largely acts as an overarching storage container and as a sort of “mediator” between the database, Attendee, and Organizer. It holds all of the “finalized” data that the Organizer wants to pass on to the Attendee in its Convention Assets (see **GLOSSARY**.) It also stores static data descriptive of the convention and common to all Assets - the Convention Details (see **GLOSSARY**.) An Organizer provides Documents to the database and to their Convention, making use of the class of the same name. They do the same for Events, which contain information such as times, titles, hosts, and tags. Potential tags and hosts are pulled from a static list provided by the Convention. All of the Event data is passed to the Scheduler, which parses it into a backtracking algorithm.The Scheduler takes Events from the Convention, and gives back an ordered container of Events that make up its Schedule. The Convention is accessible by the Organizer to edit, view, and finalize. Its Assets and Details are passed along to the Attendee, who views it as it has been parsed into a mobile Android application.

The product makes frequent and intuitive use of a common software design pattern - Model View Controller. It contains both a Front- and Back-End segment (see **GLOSSARY**) that make use of this architecture in their inherent code structure. The Back-End section of the product, a web application, is developed using the Ruby on Rails framework. All of our classes, data, and base objects (Events, Convention, etc. - our “models”) are written code that is used by “controller” files that pull information from the linked database and form it into an HTML “view”. Similarly, our Front End mobile Android application has data organized in Java “model” files (again, Events and such, all items displayed to the Attendees and pulled from the Back-End) that are handled by other Java “controller” files (referred to as “activities”) that link them to XML “view” files that define the application’s layout.

In its object-oriented design, this product also makes frequent use of Dependency Inversion principles. The Scheduler portion of the Back-End functionality is a good example of this. It is established with a set of dependencies. These take the form of constraints that it

requires in order to schedule. These dependencies are then fulfilled (or, “injected”) by Event objects and Convention Details passed to it from the database.

Contribution Summary

Michael Mortimer developed the web interface as well as one of the sequence diagrams and two of the static class diagrams. Maggie Borkowski wrote the scheduler as well as the other sequence diagram and one of the static class diagrams. Rachel King provided the CRC cards and the design approach as well as templating the mobile frontend. She also provided the glossary. Kyle Samson wrote the initial unit tests and provided the contribution summary and the current status report.

Status Report

Since our elaboration, we have made many steps to completing a functional beta of our application. We have thus far met all of the steps we describe in our work schedule. We have used the UI templates to design working forms to input conventions and their minimal details. The same has been done for the convention documents, events, hosts, and rooms. In addition to the forms, there are functional pages that display the documents, events, hosts, and rooms. In addition, a base implementation of our scheduler has be designed and prototyped. We have also created a simple interface that will strike the ground for our mobile front-end structure. We provided static class diagrams and sequence diagrams. We have also described our design approach and provided CRC cards of the major classes.

We have not run into many risks in between deliverables. We are proud to say that the only long standing risk is largely mitigated. This risk was that our team has an unfamiliarity with the tools that we are using. The tools in question are Android Studio, Ruby on Rails, and Git. The existence of our beta application is proof of our victory in mastering to the point of developing a functional prototype of our product.

The next big step we need to make is integrating the separate parts of the application. We will need the web application to provide the data for the mobile application to display. We also need to apply the scheduler’s functionality to our web application. We also need to create a system for users to make accounts and log in. The mobile application is admittedly bare bones but in the next few iterations it becomes more of a focus.

GLOSSARY

Attendee - A person attending a given Convention; the Front-End user.

Back-End - A segmentation of the project/product that encompasses the features accessed directly by Organizers.The Back-End segment manifests as a program that is hosted through a website. It allows an Organizer to create and organize a Convention, giving it Details and Attributes to provide to the Attendee.

Convention - A large, organized, and scheduled collection of Events hosted by an Organizer

Event - A small panel, talk or demonstration that occurs during a Convention.

Convention Asset - All combined, collective attributes of a Convention. Includes a Schedule, Events, and any Documents provided by the Organizer.

Convention Detail - Some defined attributes of a Convention, to be used elsewhere in scheduling and organization via this program. Convention Details include: Rooms, Event Hosts, and Event Tags.

Documents - Miscellaneous images and files that an Organizer may want to include in their Convention materials. For example, a map of the Convention space.

Front-End - A segmentation of the project/product that encompasses the features accessed directly by Attendees. The Front-End segment manifests as a program that runs on a mobile device. It allows an Attendee to "attend" a Convention and view its Schedule, a list of Events, and any Documents provided by the organizer.

Organizer - A person organizing a given Convention; the Back-End user.

Schedule - A collection of Events occurring at a given Convention, organized by Room and Time.

Schedule Generator - Application in the Back-End portion of this product that takes as input all Events provided by the Organizer, as well as Convention Details, and composes them into a Schedule.