Final report - VIF Tech

1. Members and Roles:

- Ajobiewe, Niyi E
- Anyiam, Nkemdi F — Product Owner (#0, 1, 2, 3, 4, 5)
- Chung, Insoo — Scrum Master (#0, 1, 3, 5)
- Nan, Nangga
- Zesch, Ryan Scott — Scrum Master (#2, 4)

2. Summary:

The website we have been building for the Visualization Industry Fair (VIF) serves as the first real career fair management website the event will ever have. For last year's fair, Nkemdi was chosen to create a static website (now only viewable at https://nkemdianyiam.github.io/VIF-2022-Site/) that had to be updated manually and had zero back-end functionality, and in years prior, the website was a small, unappealing subpage on the old College of Architecture website. The main customer need as per our client's wishes is to create the foundation for a brand new site where users can create accounts, sign up for events, manage profile data, and view useful information such as company schedules—all without relying on Google Forms, Qualtrics, and other outside applications (this was a big motivator for our client). The stakeholders are not only the VIF Committee, who will use the site to manage data more easily, but also the students and company representatives who will enjoy a more professional, streamlined career fair experience compared to the past fairs.

Our website meets the needs above with various features. Namely, we have a robust allowlist system that allows admins to control which emails and domains—and for which user types and companies—can create accounts and update certain site information, lets company representatives and volunteers to fill out built-in timesheets for open events, lets users view their own schedules to see who they have meetings with, allows users to update their profile data, automatically generates a block schedule showing when companies are holding booths at the virtual fair, lets users search/filter through companies attending the fair, and allows company representatives to view a list of all students along and their information with the options to email them to set up interviews outside of the scheduled meetings.

3. User Stories:

For a full detailed list of all user stories, please refer to **Appendix A**. This appendix includes story descriptions, point values, and completion status.

This project contains a total of 134 completed story points. These completed story points were distributed as follows:

- Ajobiewe, Niyi E (10/134)
- Anyiam, Nkemdi F (44/134)
- Chung, Insoo (35/134)
- Nan, Nangga (13/134)
- Zesch, Ryan Scott (32/134)

A breakdown of completed story points per iteration is shown in the chart below:

Iteration	I0 (Sept 23)	I1 (Oct 7)	I2 (Oct 21)	I3 (Nov 4)	I4 (Nov 18)	I5 (Dec 2)	I6 (Final)	Total
Ryan	0	3	3	5	8	10	3	32
Insoo	0	4	3	5	5	15	3	35
Niyi	0	0	0	6	0	0	4	10
Nan	0	3	0	6	0	3	1	13
Nkemdi	0	8	4	11	6	12	3	44
Total:	0	18	10	33	19	40	14	134

Points were assigned to stories by discussing point values and reaching an agreement. When new stories arose due to evolving customer requirements, points were assigned during scrum meetings. We ended up having approximately 45% of story points focusing on User functionality, 35% on Event/Meeting functionality, 10% on FAQ functionality, and the remainder on smaller tasks.

As we implemented our product, we continually gathered new requirements from the customers. Due to our very frequent communication with the customer, and continuous design discussion with them, we did not have to make any major revisions to stories once they were created. Based on customer feedback, a few stories were dropped entirely, as they were not needed features.

The main user stories we focused on accomplish the tasks listed below.

- Account management for different types of users (students, company representatives, volunteers, and administrators)
    - Allow for default user creation (students, volunteers, administrators)
    - Allow for company account creation, filtered on a dynamic allowlist, with email verification

    - Allow companies to maintain single a primary contact, who has abilities to edit the company's allowlist and update company properties
    - Allow for different privileges per user type
    - Stories related to account management took up approximately **45%** of all completed story points
        - This is a reasonable amount of work to put into account management, as account management is essential for maintaining a registration system
- Events & meetings
    - Create events that can take registrations and help admins connect students with volunteers and company representatives for hiring events.
    - Allow volunteers and company representatives to sign up for events and post their availability.
    - Allow students to sign up for events.
    - Allow administrators to connect students and {company representatives, volunteers} based on the latter's availability.
    - Stories related to events and meetings took up approximately **35%** of all completed story points
        - This is a reasonable proportion of points to allocate to event and meeting management, as this is the functionality which will allow users to actually participate in the career fair
- Content creation
    - Allow for administrators to maintain a dynamic contents page (FAQ, about, etc.), with rich text editing.
    - Stories related to content creation took up approximately **10%** of all completed story points
        - It makes sense that fewer points were allocated to content creation, as this only accounts for FAQ style information, which is relatively straightforward

Stories that didn't get implemented:

- File upload
    - As 1) this wasn't a client request feature, 2) in most cases URL or embed is enough, and 3) to focus on core functionalities, we did not work on this story.

4. Scrum Iterations:

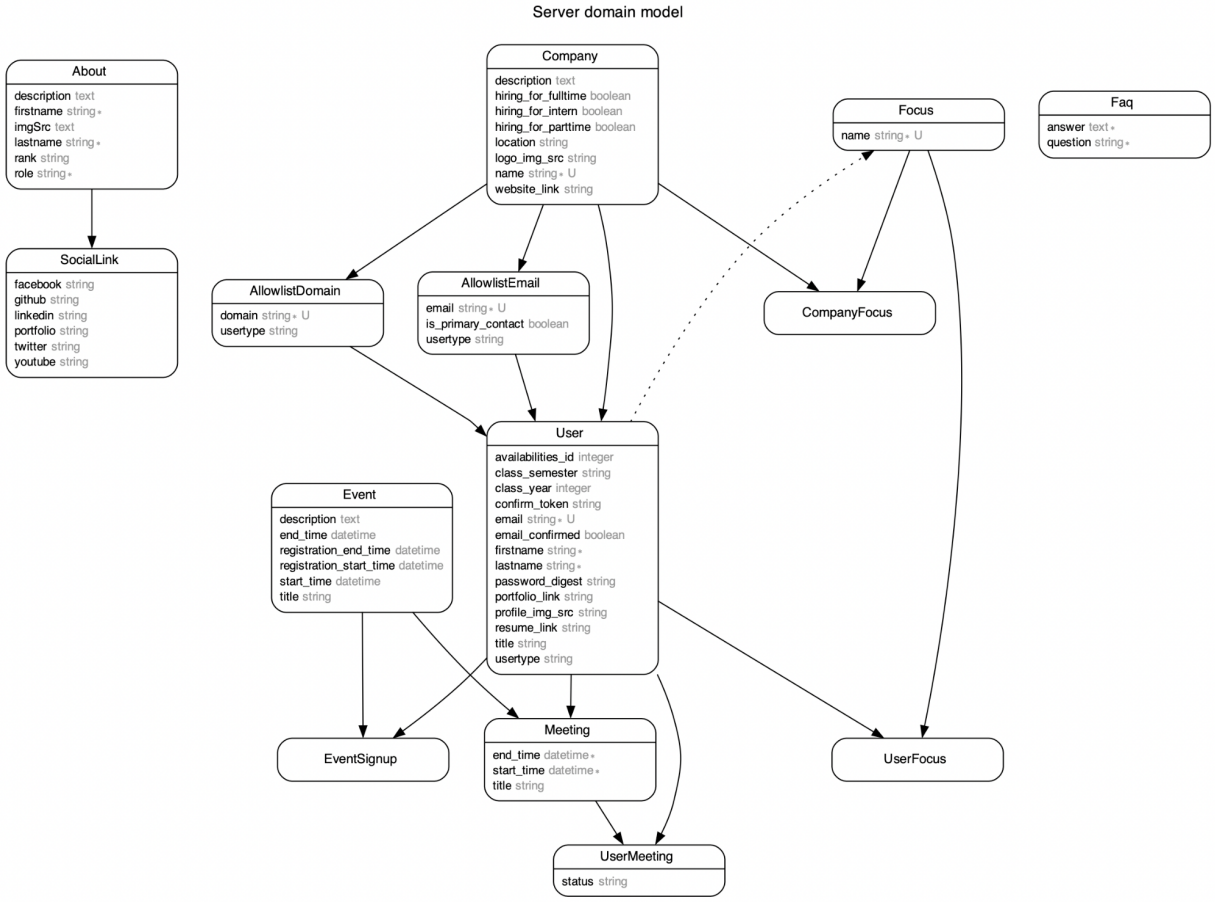
Appendix A contains a full detailed list of all user stories, which is organized by iteration number. The following list contains the major improvements which each iteration introduced:

- Iteration 0 (Sept 23)
    - Points completed: 0
    - Summary: This iteration focused on gathering requirements from the customer, and initial work on translating these requirements into designs for the product. Starter code for both frontend and backend were put into place as a starting point for future iterations.
- Iteration 1 (Oct 7)
    - Points completed: 18
    - Summary: This iteration introduced basic functionality to the project. In the backend, users are able to create accounts, and emails are sent to verify emails. Deployments of both frontend and backend were created, and work was done to make sure the separate frontend and backend can communicate correctly.
- Iteration 2 (Oct 21)
    - Points completed: 10
    - Summary: This iteration introduced allowlists for account creation, which are configurable by admins and company representatives. Additionally, meetings were added, allowing users to schedule times to mee during the career fair. In this iteration, the frontend was able to incorporate features from the previous iteration that were backend only, such as login and registration.
- Iteration 3 (Nov 4)
    - Points completed: 33
    - Summary: This iteration introduced primary contacts for companies. Additionally, company and meeting models were improved with new routes and attributes. An FAQ model was created, with appropriate routes. Frontend incorporated existing backend features, such as company creation, allowlist editing, and primary contact handling.
- Iteration 4 (Nov 18)
    - Points completed: 19
    - Summary: In this iteration, the user account gained new routes for password changing and account deletion. An availability model was created, in order to permit or deny meeting creation. Frontend incorporated FAQ creation and deletion.
- Iteration 5 (Dec 2)
    - Points completed: 40
    - Summary: In this iteration, the primary contact was changed to be unique instead of multiple. Emails are now compared case agnostically, and allowlists work more intuitively. An events model was added, in order to group meetings. A focus model was added, in order to group users

    - together based on interests. Frontend development incorporated new allowlist features, event display, and event registration.
- Iteration 6 (Dec 9)
    - Points completed: 14
    - Summary: Frontend was finalized, and all expected features are present. Backend received minor changes, as needed, such as introducing new fields for models. This document, and other final requirements, were created.

5. Design

5.1. Entity relationship diagram



5.2. Design choices

Again, our app's main purpose is to connect the students to hiring companies for career opportunities. Our design decisions all revolve around this and specific client requests.

5.2.1. Users and registration

We support four types of users through the **User** model. Although users differ in privileged actions, we decided that we should share a model as there are more common actions than unique ones. One of the common flows is our registration process. We wanted to filter out users not from the expected party. Hence we adopt *email filtering and confirmation* strategy. The flow requires that your email matches the expected pattern - provided by **AllowlistDomain** and **AllowlistEmail** models - and require users to confirm their email via link. As we share one User model for all user types the registration implementation shares a common pipeline. And once the user identity is confirmed, *usertype* attributes are used to provide access to privileged actions.

One thing to note is that company representatives are assigned to **Company** models. This association allows these users to perform actions as a representative of their company.

5.2.2. Connecting users

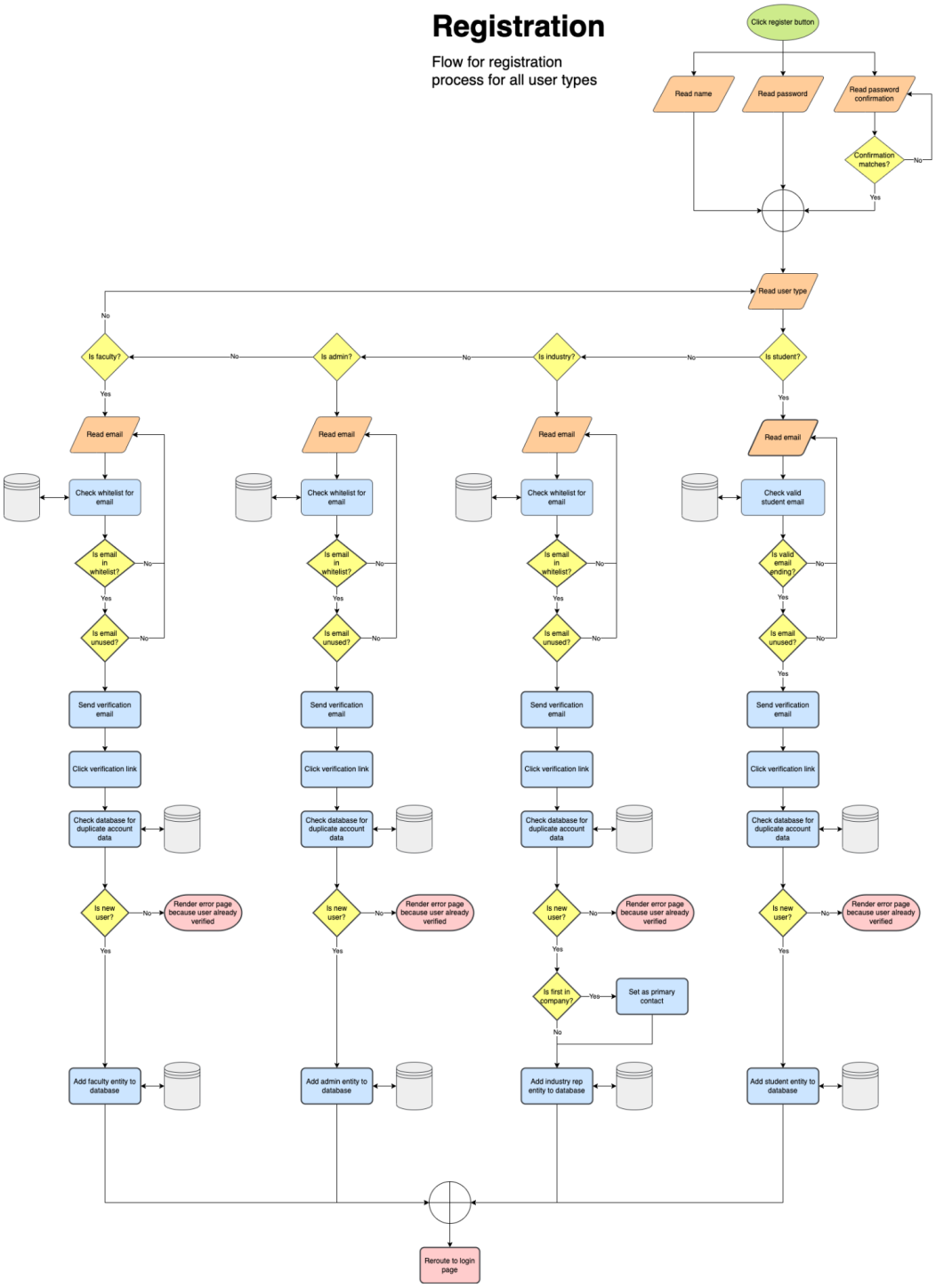
The admin users should be able to connect the students to volunteers for events like mock interview sessions and students to companies for sessions relevant to hiring events.
Events: The requirements from our clients were that there should be an event with start/end time, and also registration start/end time. Hence we created an **Event** model to encapsulate a group of activities. Users can choose to participate in these events by signing up. On signup, **UserEvent** association is created to keep track of the signup status. The privileged users: volunteers and company representatives can provide time availability. Instead of creating a separate model for availability, we just consider meetings with 0 invitees as available.
Focus: Focus instances can be created by admins. The users or the company instances can assign themselves to multiple focuses. Then, other users can view companies and users grouped by an instance of focus.
Meetings: Based on the signed up users' availability/focus, admins can connect the users via **Meeting** instances. These instances will be assigned a time span and multiple associated users, so the users can communicate. **UserMeeting** model keeps track of the invited users and the invitation status (accepted/rejected/canceled/pending)

5.2.3. Contents

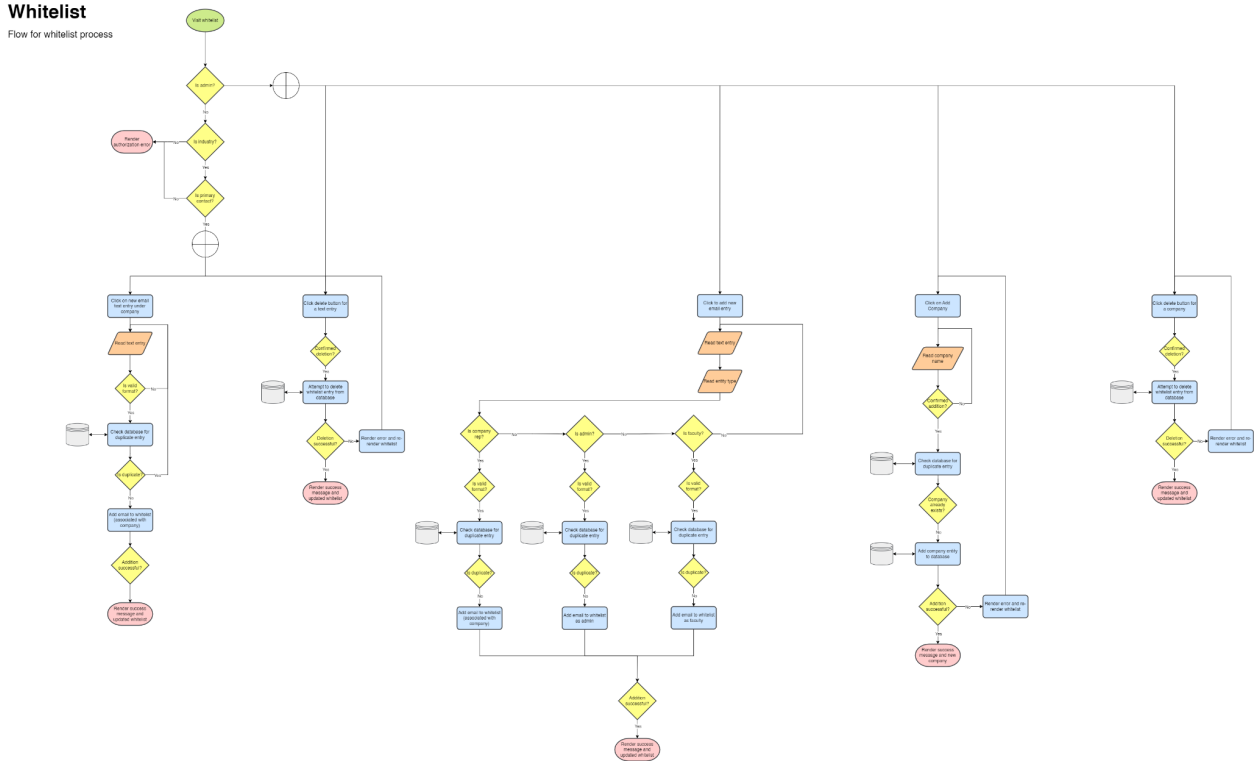
Administrators should be able to create/update/delete contents on the website - mainly the information on FAQs and About Us page. For these we created **Faq** and **About** models and we provide admin UI for create/update/delete the contents.

5.3 Additional design information

5.3.1. Flowchart for registration

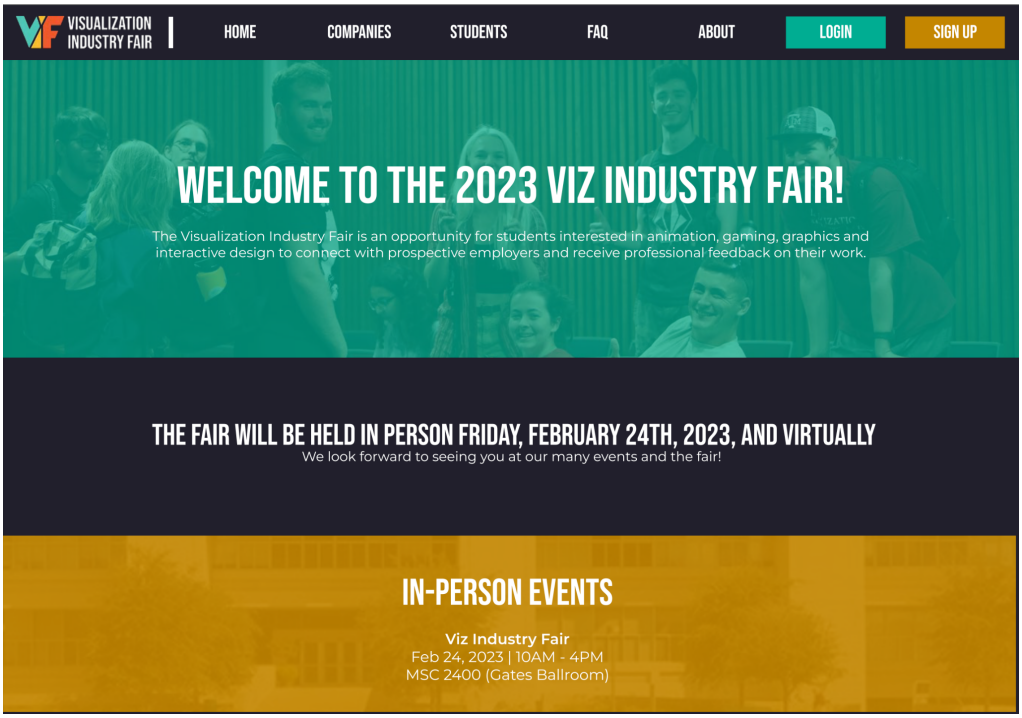


5.3.2. Allowlisting



5.3.3 Mock design

The mockups are currently being designed by visualization students on the VIF Committee, and we helped provide guidance, feedback, and sometimes instruction along the way. Below is a preview of the design. Link to the full design mockups can be found in Section 11. Note that the VIF Committee plans to continue the design of the mockups into winter break in preparation for the 2023 events.



5.3.4 Routing

We provide 113 routes - all base REST APIs for atomic transactions and high-level APIs for commonly used compound operations. We provide the list in **Appendix B**.

6. Testing

6.1. Our development/testing approach - XP/TDD

The clients had incrementally developed the requirements for the project for the duration of the project. This meant that there were modifications and additions to larger requirements. For example, while the event model should encapsulate the meetings and availability of signed up users, the requirement came at one of the later iterations. Hence we followed the principle of **extreme programming**[1]:

- Doing extensive code review:
    - We only merged code that was reviewed. Our full implementation is the result of 112 PRs most of them reviewed with exception of hotfixes.
- Unit testing of all code

[1] https://en.wikipedia.org/wiki/Extreme_programming

    - Our main testing approach was **TDD**, creating unit tests and creating relevant functions. The comprehensive cucumber tests came after each piece of code was tested (i.e. RSpec tests -> Cucumber tests).
    - Our code coverage maintained 90+% at all times.
- Not programming features until they are actually needed.
    - This was required as the customer requirement was not set in stone.

6.2. Benefits/problems of our approach

Benefits are that we maintained a high back-end percentage of coverage at all times. And this helped us reduce the amount of time required to communicate between the developers about "*why is this not working as intended?"* because we always had tests to reference. And as everything was unit tested, we didn't run into complex errors later on. Also in the early process of our implementation, we adopted pair programming to get used to the Rails' "convention over configuration" and cucumber testing. Pair programming helped all of our members to be on the same page and helped with the initial learning curve.

Problems we ran into was because the unit tests make some assumptions about the usage of each function, whenever the overall flow changed per client requests. The RSpec unit tests needed to be changed and not just the cucumber tests. This added to the implementation burden. However, we think that this is more of a problem that arises from changing requirements rather than the testing approach.
Another problem we ran into was an important lesson: Testing UI is *hard*—much harder than anticipated. As expected, the testing was very useful, but once the tasks became more complex as the sprints progressed, it ultimately became too time-consuming. However, with more practice and time working on good testing habits, it will no doubt become easier because the value of testing is pretty clear now that we have all experienced it.

6.3. Final coverage

6.3.1. Frontend (react):

- Test environment: Cucumber + Cypress
- Command:
    - npx cypress open (to see the awesome GUI)
    - npx cypress run (to run it in headless mode)
- Coverage (%): 45.42
- **NOTE:**

    - npm version = 8.5.3
    - node version = 14.17.0
    - Include 2 files in the /src directory called *.env.development.local* and *.env.test.local*, both with the following 2 lines:
        - REACT_APP_BASE_URL='http://localhost:3001'
        - SKIP_PREFLIGHT_CHECK=true
    - To view the coverage report, view the new *coverage* folder that gets generated in the /src directory called /coverage, go to /lcov-report/src, and open index.html

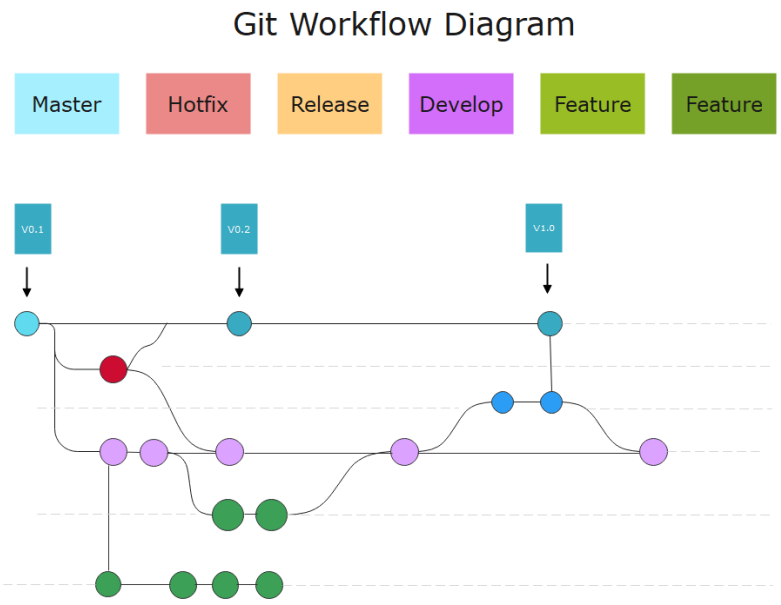
6.3.2 Backend (rails):

- Test environment: Cucumber + RSpec + SimpleCov
- # of tests:
    - Cucumber: 133 scenarios / 1230 steps
    - Rspec: 109 examples
- Command: *rake spec cucumber*
    - Report location: *server/coverage*
- Coverage (%): **93.22%** covered at **9.98** hits / line
- **NOTE**: In order for tests to run,
    - Place the included file `.env.test.local` inside the directory `/server`
        - File content:
            - `ORIGINS_URL="http://localhost:3000"`
            - `ADMIN_PW="pw"`
    - Then perform database migrations (`rake db:migrate ENV='test'`)
    - Then seed the test database (`rake db:seed ENV='test'`)
    - Make sure you are on `develop` branch and on `server/` directory
    - Run tests with `rake spec cucumber`

7. Configuration management

7.1. Git workflow

We maintained a *main* and *develop* branch and followed the basic git flow guidelines where each feature requires a *PR* until it is merged to *develop* and where each release merges develop branch to *main* branch.



One difference from the diagram was that our releases **did not merge into main** as our website URL is currently widely-used by the public (mainly students, representatives, and volunteers who plan to attend events). Hence, the main branch maintained a webpage that shows under construction, and we accumulated our work exclusively on the develop branch. Aside from the main and develop branches, we had 114 feature/hotfix branches corresponding to 114 PRs. Our sprint cycle followed the iteration cycle of the course. Hence we had 5 sprints and 5 corresponding releases that have been marked with a git tag. Our Heroku environment was set up to pick up the newest commits on the develop branch for automatic deployment.

7.2. Test/production Configuration Management

There were some configuration differences between test environments and deployment environments. For example, the local environment used sqlite3 DB while deployment DB utilized PostgreSQL. The differences were managed with Rails' dotenv gem. Further details are provided in Section 10.

7.3. Employed guidelines for PR merges

1. From your branch, execute: `git merge develop`
2. Create a PR from your branch to `develop`
3. Fix any merge conflict you see.
4. Run all tests including the tests written by others and make sure they all pass. If not fix them.
    a. `rake spec cucumber`
    b. Review files under `server/coverage` to see what you are missing and add more tests if required.
5. Get reviews from peers, bug people if you have to.
6. If your PR's been approved, merge it. New commit to will automatically be deployed on heroku.
7. If your PR includes DB changes, run below commands.
    a. `heroku run -a vif-tech rake db:migrate RAILS_ENV=production`
    b. `heroku restart -a vif-tech`
        i. optional, *only do this if you run into errors*

8. Client meetings

Iteration #0

- Meeting: Sep. 21st [link] (audio broken)
- Details:
    - Discussed initial iteration 0 stories.
    - Began discussion about how to manage who can register via emails.
    - Talked about generating block schedules for fair availabilities.

Iteration #1

- Meetings: Sep. 30th [link] (audio broken), Oct 5th [link]
- Demo: Oct. 7th part-1— [link] (audio broken), part-2(fixed audio)—[link]
- Details:
    - Sep. 30:
        - Proposed initial implementation idea of site allowlist where A) the admin would be able to add company groups with allowed emails/domains and B) company reps who are primary contacts would be able to modify their own company's individual allowlist.
        - Discussed idea of volunteer/student timesheets that admins would be able to see and use to generate volunteer-student pairs.
    - Oct. 5:
        - Walked through visual flow of allowlist for all user types
        - Discussed some UI ideas for the allowlist display
        - Showed the new under-construction page on the main VIF site
    - ***Oct. 7***
        - Demonstrated successful domain name transfer, redirection pages, SPA (single page application) routing capabilities, and rudimentary

registration form just to confirm registration confirmation emails work
    - Started another recording because audio was broken and discussed the Katelyn wanted to create UML diagrams
    - *Really* emphasized that Katelyn was happy with our progress

Iteration #2

- Meetings: Oct 14th (no recording), 19th (no recording)
- Demo: Oct. 21th [link]
- Details:
    - ***Oct. 21:***
        - Demonstrated improved registration where allowlist prevent non-present emails from registering
        - Showed successful registration for student account with allowed student domains
        - Demonstrated unsuccessful and successful login
        - Explained BDD/TDD process just for fun
        - Discussed how companies/representatives would specify their career interests (decided that reps would just have titles while the company groups as a whole would have interests)
        - Discussed best implementation of choosing interests overall (decided to do a set list that people can choose from)
        - Discussed initial concept of transferring primary contact privileges for company reps
        - Changed usertypes to admin, student, volunteer, and company representative, removing faculty since they're essentially just volunteers

Iteration #3

- Meetings: Oct 26th (audio broken) [link], Nov 2nd [link]
- Demo: Nov 4th [link]
- Details:
    - Oct. 26:
        - Discussed potential implementation for applying primary contact status (we decided to incorporate it into allowlist emails).
        - Discussed further that the list of interests (focuses) will be a set list that everyone will choose from.
        - Provided feedback/suggestions on mockup designs

        - Discussed what fields some user types should have (students should have portfolio link, resume link, class year, and interests)
        - Katelyn requested the ability for events to enforce registration deadlines
        - Discussed implementation plan for timesheet UI and initial idea for who can create meetings for mock interviews, portfolio reviews, and the virtual fair (the initial idea was that admins would pair people together for the former 2 while company representatives would be able to directly set up one-on-ones through the site with students)
    - Nov. 2:
        - Provided feedback for allowlist mockups
        - Discussed what mockups should look like for filling timesheets for the different events (idea was to have events timesheets on one page)
        - Discussed which fields should or should not be editable on profiles.
        - Discussed whether companies should be able to have multiple primary contacts or not (planned to have singular primary contacts)
        - Discussed what fields should be visible at registration depending on the usertype chosen (primary contacts will have to wait until after registration to apply the company group's focuses)
    - ***Nov. 4:***
        - Demonstrated the admin view for adding and modifying company allowlists (adding company groups works as well as adding primary contacts, person emails, and personal domains to each company group).
        - Demonstrated that registration (for people registering as company representatives) successfully establishes a 2-way association between users and companies so that allow lists have no impact on each other.
        - Gave feedback and suggestions on mockups (primarily on good color conventions).
        - Katelyn decided that having a whole system with representatives requesting interviews with students and having notifications for accepting/rejecting/pending meetings is too complex for this website—decided that it would be better to simply let representatives see a list of students and their info with the option of clicking their emails to schedule one-on-one interviews outside of the site (much more flexible and intuitive overall).

Iteration #4

- Meetings: Nov 9th (no link), Nov 14th (no link), Nov 16th (no link)
- Demo: Nov 18th [link]
- Details:
    - ***Nov. 18***
        - Demonstrated rich text content-management for FAQ page so admins can edit the FAQs freely, allowing the ability to add, edit, and delete them.
        - Summarized talk-points from the previous November meetings for the sake of the recording:
            - Company groups, students, and volunteers will all have “focuses”
            - While assigning students to volunteers, admins will see a dropdown next to each volunteer’s time availability and apply students that way.
            - Discussed UI for assignments page (all volunteers should show up on the same page so that it’s clear that the “Save Changes” button is updating the whole thing
            - Discussed idea for an automatically-generated block schedule for company booth times
        - Katelyn reiterated multiple times that it would be fine to hard-code event times since they are only updated once per year.
        - Decided that accounts will not be deleted every year.
        - Discussed rich text editor UI
        - Re-confirmed that the real UI design-work will occur over winter break
        - Discussed that there will eventually be a schedule for the physical fair (but that obviously won’t be done this semester because that information won’t be decided this semester)
        - Discussed idea to have Event entities to keep track of which users are planning to attend which event so that the admin assignments page knows which users to show.

Iteration #5

- Meetings: Nov 30th (no link)
- Demo: Dec 2nd [link], Dec 5th (no link)
- Details:
    - ***Dec. 2***
        - Demonstrated dropdown at registration where company representatives can specify which company they’re trying to associate with

        - Demonstrated updated primary contact system that allows transfers between company representatives (both by admins or by the primary contacts themselves)
        - Demonstrated that logins will be unsuccessful unless the registration email confirmation is confirmed
        - Demonstrated ability for volunteers to sign up for events and fill out timesheets to specify their availability
        - Showed that seed data will be used to populate details for the new Event entities
        - Demonstrated ability for admins to see who registered for events and then assign students to volunteers.
        - Discussed that registration will just disallow registration/timesheet modifications for events for normal users
        - Decided that more general events (like talks) will not be included as events. They'll just be displayed on the page or something.
        - Katelyn decided that the Event data should indeed be hard-coded because it's simpler to just change that code once per year.
        - Discussed plans for mobile mockups and other mockups (which the VIF Committee will continue to create and improve during winter break)
    - ***Dec. 5***
        - Showed Katelyn the fully-fleshed out timesheets and admins' ability to assign users to each other.
        - Discussed end-of-semester survey and confirmed for the umpteenth time that the plan is to flesh out the UI styling over winter break alongside the mockups and other new information that gets decided. (*We* do not need to keep beating you over the head with this since the TA already told us at the very first meeting that this was okay, but we *really* want to make sure our bases are covered).

9. Implementation

9.1. Repository structure

At the root level, we have 3 directories: *server*, *client*, and *documentation*. The following lists the directories and subdirectories

- server/: Rails source code
    - app/: Core rails classes that compose models and controllers are implemented here

    - config/: Configuration variables and route information are stored here
    - db/: DB migration files are generated and stored here. Also, in test environments, sqlite3 DB file is stored here.
    - features/: Cucumber tests
    - spec/: RSpec tests
- client/: React source code
    - cypress/: Cypress tests
    - src/: Core React components, Redux store, and helper classes are implemented here (all in TypeScript) along with Sass (SCSS syntax) stylesheets
- documentation/: Relevant document deliverables for each iteration.

9.2. Libraries

Rails Gems

- Rspec for unit tests
- Cucumber for acceptance level tests
- Bcrypt helps us generate secure passwords for our users.
- Simplecov generates interactive html files that show us what tests are missing.
- Pg supports our production database.
- Puma provides fast secure HTTP protocols.
- Tzinfo-data allows time zone data that are necessary for our interview and event scheduling.
- Bootsnap uses cache to improve our app performance.
- Rack-cors helps our app make cross domain AJAX calls.
- Email_validator tells us whether a user's email was valid.
- Active_model_serializers allows us to put arrays in a JSON response body.
- Database_cleaner helps us clean the test database before running each test.
- *Additional gems were utilized but above lists the most relevant ones.*

NPM Packages

- Axios provides an easy-to-use interface for making API calls
- React helps create modular components whose state can be easily managed
- React-Redux allows the separation of business logic from view logic
- Redux-Thunk facilities asynchronous functionality with Redux
- TypeScript allows typesafe code, which is a must-have for a project this large
- React-Draft-WYSIWYG provides a customizable rich text editor whose functionality can be integrated into React code
- Cypress allows for fairly robust integration testing with React code

9.3. Commands

9.3.1 Heroku deployment

Installing dependencies

- `heroku run -a vif-tech-dev bundle install`

Resetting and re-seeding DB

- `heroku pg:reset --app vif-tech-dev`
- `heroku run rake db:migrate --app vif-tech-dev`
- `heroku run rake db:seed --app vif-tech-dev`

9.3.2 Test commands

- See Section 6.3

10. Issues with Tools and Deployment

10.1. Heroku

We deployed our rails backend on Heroku (API only) and react front end on Netlify. Initially, when we were testing locally, we exclusively used sqlite3 as our DB engine. However we found out that a file-based engine like sqlite3, is not supported by heroku. To maintain the data persistence, we utilized PostgreSQL for the production environment. There were other minor differences such as the allowed URL of the requester for the test environment and the production environment - *for testing we used a localhost url and for production we used the url of the netlify deployment*. We initially managed the configuration manually but we later utilized Rails' ***dotenv*** gem to maintain the different set of configuration variables which lessened the confusion in our development process.

10.2. Netlify

Netlify is used to deploy both the development testing site (https://www.test-vizindustryfair.com/) through the *develop* branch as well as the real site (https://www.vizindustryfair.com/) through the *main* branch. The former uses a custom domain through Netlify; this was done just to make sure any CORS and cross-site-cookies matters were working. The latter hooks up to the vizindustryfair.com domain that is on HostGator (which we did very early in the semester with the VIF Committee members who maintain the site domain).

10.3. Github

We utilized Github for version control. As explained in Section 7, we only allowed tested and reviewed PRs to be merged to our code base.

- Number of PRs: 114
- Number of commits: 679

Initially, unfamiliar members found the learning curve for git concepts (commits, branches and PRs) to be steep. In the initial iterations we utilized pair programming to work through the learning curve and resolve issues that arose from merge conflicts. In the later iterations, we were able to stick to the git flow explained in Section 7 without much trouble.

10.4 AWS Cloud 9

We didn't utilize AWS Cloud 9 service. We instead used VSCode and LiveShare extension for remote pair programming.

11. Links

- **Website URL:**
  - Netlify front-end (custom domain): **https://www.test-vizindustryfair.com/**
  - Heroku dev backend (API requests only): https://vif-tech-dev.herokuapp.com/
  - **Note:** *We have probably reiterated this too many times to count, but just to do it one last time—the https://www.test-vizindustryfair.com/ front-end is a valid deployment, but it is has "test" because this site should only be viewable by us and the VIF Committee. Then, when all the VIF information (and UI) is finalized around January and ready to replace our under-construction page on the actual https://www.vizindustryfair.com site, the* main *branch will be updated.*
- Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2598999
- GitHub repo: https://github.com/tamu-edu-students/vif-tech.git
- Design mockups: https://vif-tech.slack.com/files/U042WRF14T1/F04EC6VRKCM/figma_mockup.pdf

12. Presentation and Demo

- Video URL: https://youtu.be/v772rzt3oWg

Appendix A - User Stories

The following is a complete list of all completed user stories. There are no incomplete user stories which we have created. User stories are ordered by the iteration in which they were completed. Story 2.4, for example, was the 4th story completed in the second iteration.

We began tracking which pull request corresponds to a story after iteration 3. In retrospect, we wish we had begun this practice from the start.

Iteration 1:

Story #1.1: Create a student account
As a student
I want to create an account
So that I can access the website features
Owner: Insoo Chung
Points: 3
Status: Complete

Story #1.2: Deploy backend on heroku
As a backend developer
I want to deploy backend to heroku
So that we can test the backend API
and automate the deployment process
Owner: Insoo Chung
Points: 1
Status: Complete

Story #1.3: Email Confirmation
As website moderator
We want to confirm users' email addresses
So we can prevent multiple accounts by a single person and verify the user has that email address
Owner: Ryan Zesch
Points: 3
Status: Complete

Story #1.4: Spa Routing
As any user

I want to navigate the website with minimal reloads
so that I can reduce data-usage
Owner: Nkemdi Anyiam
Points: 2
Status: Complete

Story #1.5: Redirection pages
I want to be exposed to clear redirections prompts
so that I can seamlessly navigate the page regardless of any interactions or errors
Owner: Nkemdi Anyiam
Points: 1
Status: Complete

Story #1.6: Basic client-side student registration
As a student,
I want to create an account through the VIF website
so that I do not have to use external tools for registration
Owner: Nkemdi Anyiam
Points: 2
Status: Complete

Story #1.7: Custom domain transfer
As any user,
I want to continue accessing the VIF site through the vizindustryfair.com domain regardless of the underlying nature of the site
so that I can always find the career fair information in the same place
Owner: Nkemdi Anyiam
Points: 2
Status: Complete

Story #1.8: Create a company account
As a company representative
I want to create my account
So that I am authorized to perform interview actions
Owner: Nangga Ga
Points: 3
Status: Complete

Story #1.9: Deploy to Netlify

As a front-end developer
I want to deploy the front-end to Netlify
So that we can test the client-side code and have temporary views for stakeholders

Owner: Nkemdi Anyiam
Points: 1
Status: Complete

Iteration 2:

Story #2.1: Add Meetings Feature
As an admin
I want to create and manage meetings
So that I can connect people participating in the fair

Owner: Insoo Chung
Points: 3
Status: Complete

Story #2.2: Add Allowlists Feature
In order to manage who can make an account
As an admin or company representative
I want to control access with an allowlist

Owner: Ryan Zesch
Points: 3
Status: Complete

Story #2.3: Allow user to specify user type at registration
As a user
I want to be able specify my user type at registration
so that I can use the features specific to my needs

Owner: Nkemdi Anyiam
Points: 1
Status: Complete

Story #2.4: Make back-end CORS "origins" configurable
I want to
As a
So that

Owner: Nkemdi Anyiam
Points: 1

Status: Complete

Story #2.5: Add login front-end
As a user
I want to be able to login/logout and maintain sessions
so that I can conveniently control my account utility
Owner: Nkemdi Anyiam
Points: 2
Status: Complete

Iteration 3:

Story #3.1: Correct HTTP code
As a programmer
I want my Rails backend to render correct HTTP code in a correct, consistent way
So that I can handle errors in a consistent way.
Owner: Insoo Chung
Points: 2
Status: Complete

Story #3.2: Company registration
As a company representative
I want to be assigned to my company on registration
So that I don't have to worry about adding them later on
Owner: Insoo Chung
Points: 2
Status: Complete

Story #3.3: Meeting status
As users of VIF website
I want my meeting status to reflect various conditions
So that I don't have to assume the status from state of how meetings are structured
Owner: Insoo Chung
Points: 1
Status: Complete

Story #3.4: Company Primary Contact
As an admin

I want companies to have a primary contact
So that I can restrict some actions to primary contacts
Owner: Ryan Zesch
Points: 3
Status: Complete

Story #3.5: Filtered Allowlists
As an admin or company rep
I want to see filtered allowlists
So that I can view my data in an organized way
Owner: Ryan Zesch
Points: 2
Status: Complete

Story #3.6: Add a “About” Contents Page
As VIF website user
I want to be able to visit the Abouts Page
So that I can get information about the VIF committee
As an VIF site admin
I want to be able to manage contents on the Abouts page
Owner: Niyi Ajobiewe
Points: 3
Status: Complete

Story #3.7: Admin add company to allowlist
As an admin
I want to be able to add company groups to the allowlist
so I can control which companies can attend the fair
Owner: Nkemdi Anyiam
Points: 1
Status: Complete

Story #3.8: Admin add primary contacts to company allowlist
As an admin
I want to be able to add primary contacts to a company's allowlist
so I can control which representatives get special privileges
Owner: Nkemdi Anyiam
Points: 1
Status: Complete

Story #3.9: Admin add personal email to company allowlist

As an admin
I want to be able to add personal emails to a company's allowlist
so I can control who can register under certain companies with their personal emails

Owner: Nkemdi Anyiam
Points: 2
Status: Complete

Story #3.10: Admin remove email/domain from company allowlist

As an admin
I want to be able to remove emails or domains from a company's allowlist
so I can restrict accounts or block registration

Owner: Nkemdi Anyiam
Points: 1
Status: Complete

Story #3.11: Admin company allowlist dashboard

As an admin
I want to be able to see all companies' allowlists
so I can view and control account/registration management for all company representatives

Owner: Nkemdi Anyiam
Points: 3
Status: Complete

Story #3.12: Specify company affiliation at registration

As a company representative
I want to be able to choose which company I am trying to sign up with
so that I can guarantee that I am not going to be affiliated with a different company erroneously using my email in their allowlist.

Owner: Nkemdi Anyiam
Points: 2
Status: Complete

Story #3.13: company rep creates meetings

As a company representative
I want to create meetings
So that I can meet with people online.

Owner: Ga Nan

Points: 2
Status: Complete

Story #3.14: only admin can create company account

As a developer,
I want to restrict “create company account” privilege to admin only,
So that other users will not create companies randomly.
As an admin,
I want to create a company account,
So that I can book-keep the companies attending fair and also only allow companies in
the allowlist.

Owner: Ga Nan
Points: 1
Status: Complete

Story #3.15: Add company model

I want to CRUD a company account
As a user (admin, company representative, student,...)
So that I can get access to more information (company details)

Owner: Nangga Nan
Points: 3
Status: Complete

Story #3.16: Content management

As a VIF website user
I want to be able to access the FAQ page
As a VIF website admin
I want to be able to access, create, update, edit, and delete contents on the FAQ page

Owner: Niyi Ajobiewe
Points: 3
Status: Complete

Story #3.17: Fix Deployed Cookies

As an admin
I want cookies to work correctly
So that I can use them to store information

Owner: Nkemdi Anyiam
Points: 1

Status: Complete

Iteration 4:

Story #4.1: User: Login Protection
As an Admin
I want login to be blocked by email confirmation
So that only verifier users can participate
Owner: Ryan Zesch
Points: 3
Status: Complete - PR#51

Story #4.2: Allowlist: unique company+email
As an admin
I want email-company allows to be unique
So that an email can join any list it's allowed to
Owner: Ryan Zesch
Points: 1
Status: Complete - PR#54

Story #4.3: User: Delete account protection
As a user/admin
I want to be able to delete my/a user's account
so that the user is no longer in the database
Owner: Ryan Zesch
Points: 2
Status: Complete - PR#55

Story #4.4: User: Change Password
As a user
I want to change my password
So that I can control what my password is
Owner: Ryan Zesch
Points: 2
Status: Complete - PR#56

Story #4.5: Add availability model
As a company rep or a volunteer
I want to add my availability
In order to help admin make informed decisions about scheduling

Points: 3
Owner: Insoo Chung
Status: Complete - PR#58

Story #4.6: Add high level API for user-availability association
As an an admin, a company rep, or a volunteer
I want high level API that gives me the overlaps of availabilities and meetings
So I can assess the impact of my availability updates/deletes
Points: 2
Owner: Insoo Chung
Status: Complete - PR#59

Story #4.7: Allow FAQ creation in UI
As an admin
I want to be able to type FAQ entries from the browser
so that I can manage the page's content without touching the code
Points: 2
Owner: Nkemdi Anyiam
Status: Complete - PR#53

Story #4.8: Add rich text editor for content-management
As an admin
I want to be able to use a rich text editor to make detailed edits to certain page contents
so that I can edit and format content beyond basic text editing.
Points: 2
Owner: Nkemdi Anyiam
Status: Complete - PR#53

Story #4.9: Allow FAQ deletion in UI
As an admin
I want to be able to delete FAQ entries from the website
so that I can manage the FAQ page without code modifications
Points: 1
Owner: Nkemdi Anyiam
Status: Complete - PR#57

Story #4.10: Allow FAQ editing in UI
As an admin
I want to be able to edit FAQ entries from the website

so that I can manage the FAQ page without code modifications

Points: 1
Owner: Nkemdi Anyiam
Status: Complete - PR#57

Iteration 5:

Story #5.1: Add mass invitee update for meeting models

As an admin,
I want to be able to add multiple invitees at once
So the workflow will be more efficient

Owner: Insoo Chung
Points: 3
Status: Complete - #66, #76

Story #5.2: Add event model

As an admin,
I want to be able to group meetings, availabilities, and users associated by a larger meeting
So that I do not have to go through every instances of meeting when I work on a new event

Owner: Insoo Chung
Points: 3
Status: Complete - #70

Story #5.3: Return invitee instances with meetings

As an admin,
I want to view invitees alongside the meeting instances
So that I don't have to perform an extra action of fetching invitees per meeting

Owner: Insoo Chung
Points: 1
Status: Complete - #80

Story #5.4: Relax restrictions for meeting create action

As a company representative or a volunteer
I want to create meetings myself
So I can hint admins about my availability

Owner: Insoo Chung
Points: 1
Status: Complete - #72

Story #5.5: Event update/delete deletes deprecated dependent meetings/availabilities

As an admin
I want to update or delete and event,
And I want meetings/availabilities that become deprecated to be automatically deleted
So that I don't have to care about them

Owner: Insoo Chung
Points: 2
Status: Complete - #77

Story #5.6: Company Rep Availability Fetch

As an admin
I want to fetch all reps with availabilities for a company
So I can view them all at once

Owner: Ryan Zesch
Points: 2
Status: Complete - #62

Story #5.7: Email Storage Casing

As an admin
I want emails to be compared case-agnostically
So permissions can be granted correctly to users

Owner: Ryan Zesch
Points: 2
Status: Complete - #71, 79

Story #5.8: Primary Contact Uniqueness

As an admin
I want companies to have single primary contact instead of multiple
So that there is a clear point of contact for each company

Owner: Ryan Zesch
Points: 2
Status: Complete - #71, 73

Story #5.9: Allowlist Permissions for Reps

As I company
I want all of my representatives to be able to view the allowlist
So that they can see all company members allowed

Owner: Ryan Zesch

Points: 1
Status: Complete - #71

Story #5.10: Primary Contact Transfer
As a company
I want to be able to transfer primary contact to a user allowed by email domain as well as by email
So that I do not have to allow existing users by email first
Owner: Ryan Zesch
Points: 2
Status: Complete - #71, 79

Story #5.11: Allowlist Cascade Deletion
As an admin
I want deleting a company to delete the users and allowlists of that company
So that I am not storing unneeded information
Owner: Ryan Zesch
Points: 1
Status: Complete - #75

Story #5.12: Focus model part1
As an admin
I want to CRUD a focus object
So that other users can associate focus objects with themselves
Owner: Ga Nan
Points: 2
Status: Complete - #84

Story #5.13: Focus model part2
As a student or company representative
I want to see the existing focuses
So that I associate focuses with me
Owner: Ga Nan
Points: 1
Status: Complete - #84

Story #5.14: Add company view for allowlist
As a primary contact for a company
I want to be able to view my company's allowlist

so that I can update the emails/domains for my company without waiting for admins

Owner: Nkemdi Anyiam
Points: 2
Status: Complete - #67

Story #5.15: Add UI/functionality for transferring primary contacts

As a primary contact or admin
I want to be able to transfer primary contact status to a colleague in the given company
so that privileges can be transferred to existing users with ease

Owner: Nkemdi Anyiam
Points: 3
Status: Complete - #67

Story #5.16: Add timesheet that generates a table based on an event's timeframe

As any user
I want to be able to see the time slots for an event in a table view
so that I may view my assigned meetings and availabilities in an intuitive form

Owner: Nkemdi Anyiam
Points: 3
Status: Complete - #86

Story #5.17: Add ability to modify availability for an event

As a company rep or volunteer
I want to be able to select my availabilities for an event through the timesheets and submit the changes at the same time
so that I can conveniently specify what time slots I can take for the event

Owner: Nkemdi Anyiam
Points: 3
Status: Complete - #86

Story #5.18: Add ability to register/unregister for an event through the UI

As any user
I want to be able to register or unregister for an event from the timesheet pages
so that I can decide my attendance without any forms or email correspondence

Owner: Nkemdi Anyiam
Points: 1
Status: Complete - #86

Story #5.19: Add user-focuses, company-focuses association

As a fair attendee,
I want to find out what people/companies are good at
Hence I want to associate focuses with users/companies

Owner: Insoo Chung
Points: 3
Status: Complete - #89

Story #5.20: Add event registration time

As an admin
I want to set a registration time for events to ease the administrative burden
Hence I will limit the users' ability to sign up to events outside of registration time

Owner: Insoo Chung
Points: 2
Status: Complete - #95

Iteration 6:

Story #6.1: Changed minimum length of FAQ question and answer

As an admin
I want minimum character length for FAQ questions and answers to be 1
So that I can have more flexibility with my entries for question and answers

Owner: Niyi Ajobiewe
Points: 1
Status: Complete - #90

Story #6.2: Added ability to create, edit, and delete social link and about content in one request

As an admin
I want to be able to create, read, update and delete about and social link contents on the same webpage
so that it can be easy for me to manage individual about content and their associated social link record

Owner: Niyi Ajobiewe
Points: 3
Status: Complete - #44

Story #6.3: Add descriptive attributes to existing models

As an admin
I want to store new values for users and companies

So there is more information to display about them
Owner: Ryan Zesch
Points: 2
Status: Complete - #105

Story #6.4: Final Report
As a student
I want to work on the final report
So that I can get a good grade
Owner: Ryan Zesch, Insoo Chung, Nkemdi Anyiam, Ga Nan
Points: 1 each
Status: Complete

Story #6.5: Final Video Presentation
As a student
I want to work on the final presentation slide video
So that I can get a good grade
Owner: Insoo Chung
Points: 2
Status: Complete

Story #6.6: Final Video Presentation
As a student
I want to work on the final presentation demo video
So that I can get a good grade
Owner: Nkemdi Anyiam
Points: 2
Status: Complete

Appendix B

B.1. Routes

All implemented routes are provided here. More details can be found in the full routes document.

Controller	Controller#Action	Verb	URI
Session	sessions#create	POST	/login
Session	sessions#destroy	POST	/logout
Session	sessions#is_logged_in?	GET	/logged_in
Users	users#find	GET	/users/find
Users	users#index	GET	/users
Users	users#create	POST	/users
Users	users#show	GET	/users/:id
Users	users#confirm_email	GET	/users/:id/confirm_email
Users	users#update	PUT	/users/:id
Users	users#destroy	DELETE	/users/:id
Users	users#update_password	PUT	/users/password(.:format)
Users	users#get_meetings	GET	/users/:id/meetings
Users	users#get_accepted_meetings	GET	/users/:id/meetings/accepted
Users	users#get_pending_meetings	GET	/users/:id/meetings/pending
Users	users#get_declined_meetings	GET	/users/:id/meetings/declined
Users	users#get_cancelled_meetings	GET	/users/:id/meetings/cancelled
Users	users#get_owned_meetings	GET	/users/:id/meetings/owned
Users	users#invited_to_meeting?	GET	/users/:id/meetings/:meeting_id
Users	users#add_to_meeting	POST	/users/:id/meetings/:meeting_id
Users	users#update_meeting	PUT	/users/:id/meetings/:meeting_id
Users	users#delete_from_meeting	DELETE	/users/:id/meetings/:meeting_id
Users	users#get_availabilies	GET	/users/:id/availabilities
Users	users#get_owned_and_avail_meetings	GET	/users/:id/meetings/owned/available
Users	users#get_owned_but_na_meetings	GET	/users/:id/meetings/owned/not_available
Users	users#get_invitations_avail	GET	/users/:id/user_meetings/available
Users	users#get_invitations_na	GET	/users/:id/user_meetings/not_available
Users	users#delete_owned_but_na_meetings	DELETE	/users/:id/meetings/owned/not_available
Users	users#delete_na_invitations	DELETE	/users/:id/user_meetings/not_available

Meetings	meetings#index	GET	/meetings
Meetings	meetings#create	POST	/meetings
Meetings	meetings#show	GET	/meetings/:id
Meetings	meetings#update	PUT	/meetings/:id
Meetings	meetings#destroy	DELETE	/meetings/:id
Meetings	meetings#get_invitees	GET	/meetings/:id/invitees
Meetings	meetings#swap_invitees	PUT	/meetings/:id/invitees
UserMeetings	user_meetings#index	GET	/user_meetings
UserMeetings	user_meetings#show	GET	/user_meetings/:id
AllowlistDomain	allowlist_domains#index	GET	/allowlist_domains
AllowlistDomain	allowlist_domains#show	GET	/allowlist_domains/:id
AllowlistDomain	allowlist_domains#create	POST	/allowlist_domains
AllowlistDomain	allowlist_domains#index	DELETE	/allowlist_domains
AllowlistEmail	allowlist_emails#index	GET	/allowlist_emails
AllowlistEmail	allowlist_emails#show	GET	/allowlist_emails/:id
AllowlistEmail	allowlist_emails#create	POST	/allowlist_emails
AllowlistEmail	allowlist_emails#index	DELETE	/allowlist_emails
Faq	Faq#index	GET	/faq
Faq	Faq#show	GET	/faq/:id
Faq	Faq#create	POST	/faq
Faq	Faq#update	PUT	/faq/:id
Faq	Faq#destroy	DELETE	/faq/:id
Companies	companies#index	GET	/companies
Companies	companies#new	GET	/companies/new
Companies	companies#create	POST	/companies
Companies	companies#show	GET	/companies/:id
Companies	companies#edit	GET	/companies/:id/edit
Companies	companies#update	PUT	/companies/:id
Companies	companies#destroy	DELETE	/companies/:id
Companies	companies#reps	GET	/companies/:id/users
Companies	companies#rep_availabilities	GET	/companies/:id/availabilities
Companies		GET	/companies/public
Focuses	focuses#index	GET	/focuses
Focuses	focuses#create	POST	/focuses
Focuses	focuses#show	GET	/focuses/:id
Focuses	focuses#update	PUT	/focuses/:id
Focuses	focuses#destroy	DELETE	/focuses/:id
Availabilities	availabilities#index	GET	/availabilities
Availabilities	availabilities#show	GET	/availabilities/:id
Availabilities	availabilities#create	POST	/availabilities

Availabilities	availabilities#update	PUT	/availabilities/:id
Availabilities	availabilities#destroy	DELETE	/availabilities/:id
EventSignups	event_signups#index	GET	/event_signups
EventSignups	event_signups#show	GET	/event_signups/:id
Users	users#get_focuses	GET	users/focuses
Users	users#get_focuses	GET	users/:id/focuses
Users	users#add_focus	POST	users/focuses/:focus_id
Users	users#add_focus	POST	users/:id/focuses/:focus_id
Users	users#update_focus	PUT	users/:id/focuses/
Users	users#update_focus	PUT	users/focuses/
Users	users#remove_focus	DELETE	users/focuses/:focus_id
Users	users#remove_focus	DELETE	users/:id/focuses/:focus_id
Companies	companies#get_focuses	GET	companies/focuses
Companies	companies#get_focuses	GET	companies/:id/focuses
Companies	companies#add_focus	POST	companies/focuses/:focus_id
Companies	companies#add_focus	POST	companies/:id/focuses/:focus_id
Companies	companies#remove_focus	DELETE	companies/focuses/:focus_id
Companies	companies#remove_focus	DELETE	companies/:id/focuses/:focus_id
Companies	companies#update_focus	PUT	companies/:id/focuses/
Companies	companies#update_focus	PUT	companies/focuses/
UserFocuses	user_focuses#index	GET	user_focuses/
CompanyFocuses	company_focuses#index	GET	company_focuses/
Abouts	abouts#index	GET	/abouts
Abouts	abouts#find	GET	/abouts/find
Abouts	abouts#create	POST	/abouts
Abouts	abouts#new	GET	/abouts/new
Abouts	abouts#edit	GET	/abouts/:id/edit
Abouts	abouts#update	PUT	/abouts/:id
Abouts	abouts#destroy	DELETE	/abouts/:id
Events	events#index	GET	/events
Events	events#show	GET	/events/:id
Events	events#create	POST	/events
Events	events#update	PUT	/events/:id
Events	events#destroy	GET	/events/:id
Events	events#get_attending_companies	GET	/events/:id/attending_companies/
Events	events#get_attending_companies	GET	/events/attending_companies/event_title=Some title
Events	events#get_company_meetings	GET	/events/:id/company_meetings/

Events	events#get_company_meetings	GET	/events/company_meetings/event_title=Some title
Events	events#get_availabilities	GET	events/:id/availabilities
Events	events#get_meetings	GET	events/:id/meetings
Events	events#get_users	GET	events/:id/users
Events	events#signup	POST	events/:id/signup
Events	events#signup	POST	events/:id/signup/:user_id
Events	events#signout	DELETE	events/:id/signout
Events	events#signout	DELETE	events/:id/signout/:user_id

B.2. Coverage report preview - full report is generated after running all the tests.