CSCE 606 Final Project Report

Spring 2023

Team Members

- Matan Broner - Product Owner
- Liam Berney - Scrum Master
- Chrysanthos Pepi
- Hanrui Chen

Client

Brazos Valley Council of Governments

Project

Contract Management System (CMS)

Project Summary

Our project was to implement a Contract Management System for Brazos Valley Council of Governments (BVCOG) from scratch. The client wants a platform which allows the user to create contracts, approve / review contracts, upload documents for each contract, send reminders of when contracts are about to expire. Furthermore, the user will be able to generate reports, review vendors and invite users. The client's needs highlighted the importance of Role Based Access Control. Users are classified into three tiers: Level 1, Level 2, and Level 3. Level 1 users are the platform's "administrators" and are in charge of changing platform settings and inviting new Level 2 or 3 users. Level 2 users have "read-write" privileges and can approve contracts. Finally, Level 3 users are "read" users with the least application rights.

The current implementation of the project supports all of the requirements, except for one due to lack of time. One huge factor of the completion of the project was that their top three stakeholders provided us with a requirements document at the beginning and we had weekly meetings in which we showcased new features with immediate feedback. The platform is deployed to Heroku and the client has already started testing it.

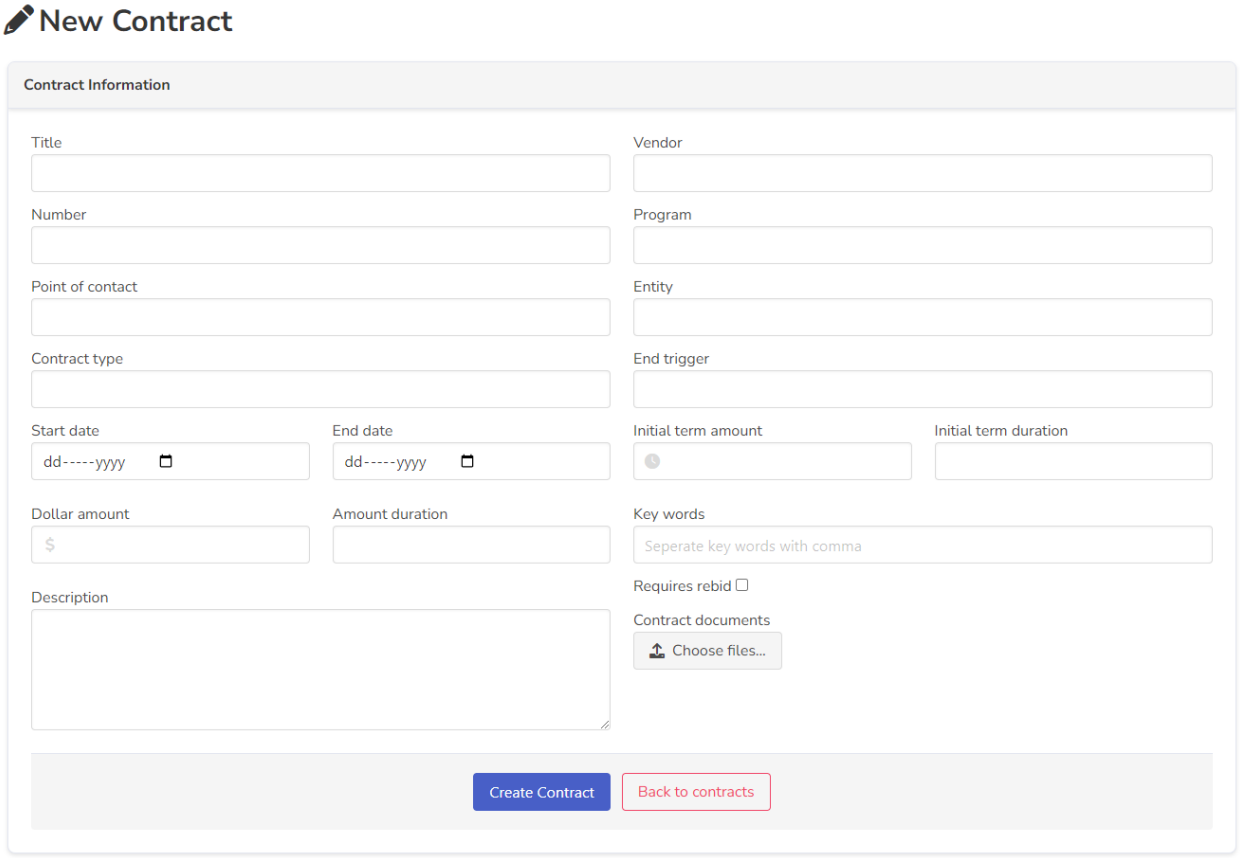
User Story Descriptions

Structure: User Story # | Implementation Status | Points

#1 | Implemented | 2

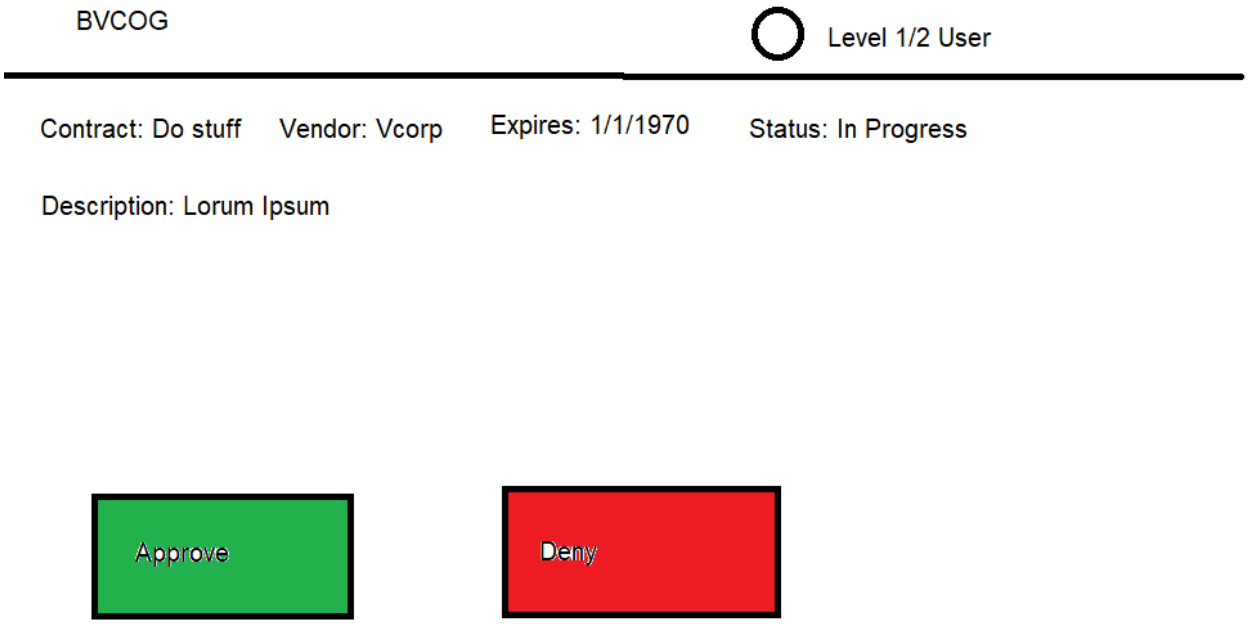
Feature: Add a new contract
As a platform user (of any level)

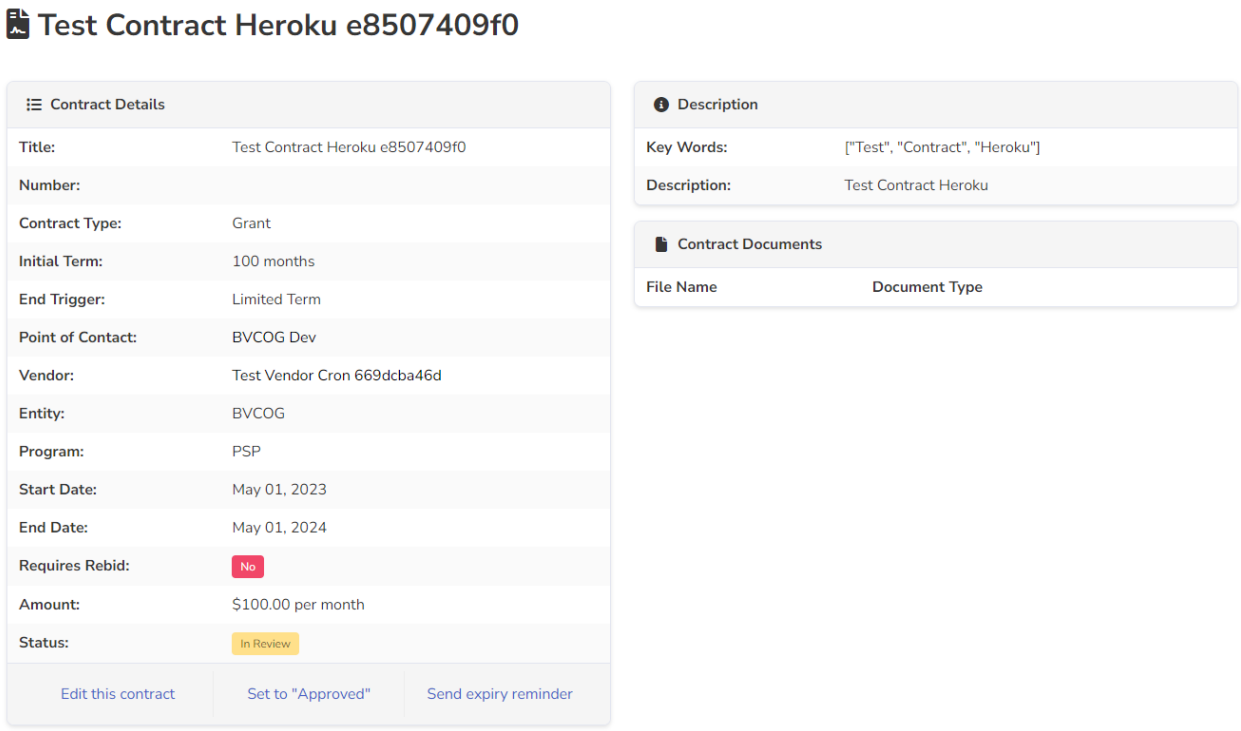
So that I can create an new contract for approval
I want to add a new contract entry to the database



#2 | Implemented | 2

Feature: Approve a contract
As a platform user (1 or 2 level)
So that a contract can move from “In Progress” to “Approved”
I want to be able to approve a contract through a single action





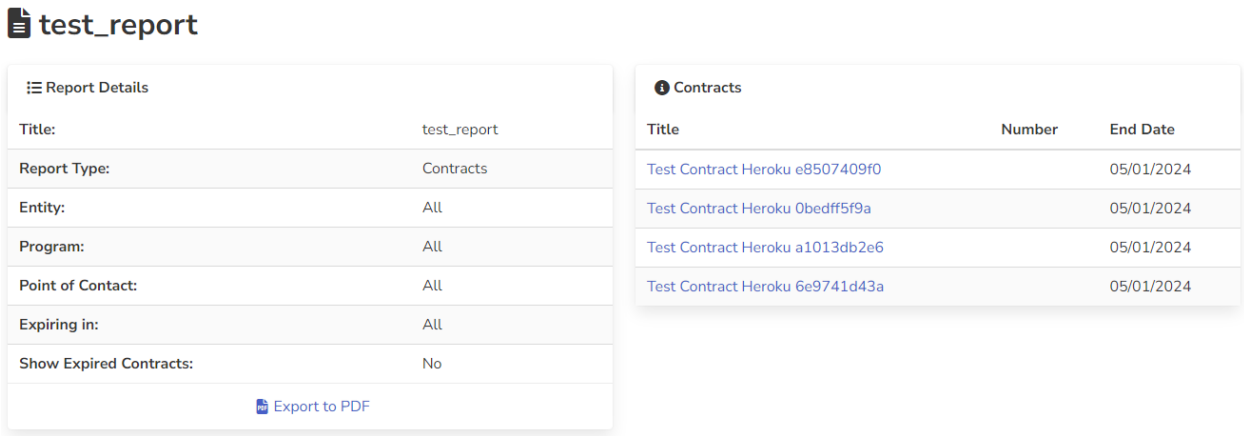
#3 | Implemented | 3

Feature: Email when contracts near expiry
As a platform user (of any level)
So that I can renew or extend important contracts

I want to receive email reminders for contracts in my entity that are close to expiry

#8 | Implemented | 2

Feature: Generate contract reports
As a platform user (of any level)
In order to see all contracts in my program
I want to generate reports for specific Entity, Program, and/or User



test_report

Report Details

Title:	test_report
Report Type:	Contracts
Entity:	All
Program:	All
Point of Contact:	All
Expiring in:	All
Show Expired Contracts:	No

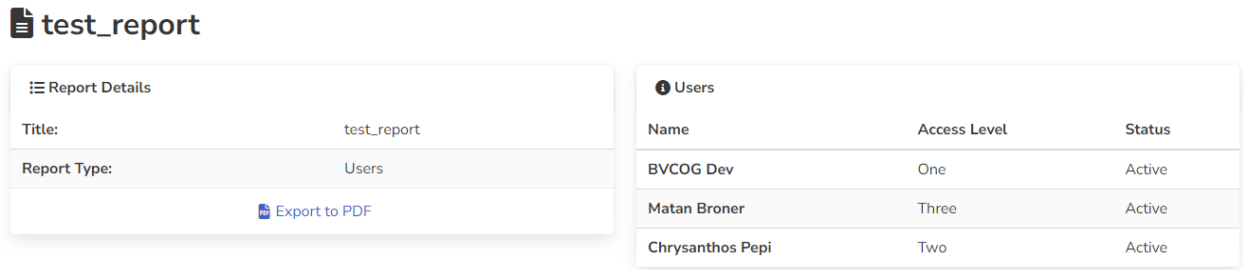
Export to PDF

Contracts

Title	Number	End Date
Test Contract Heroku e8507409f0		05/01/2024
Test Contract Heroku 0bedff5f9a		05/01/2024
Test Contract Heroku a1013db2e6		05/01/2024
Test Contract Heroku 6e9741d43a		05/01/2024

#9 | Implemented | 2

Feature: Generate user reports
As a platform user (of any level)
In order to see all user
I want to generate reports for all users



test_report

Report Details

Title:	test_report
Report Type:	Users

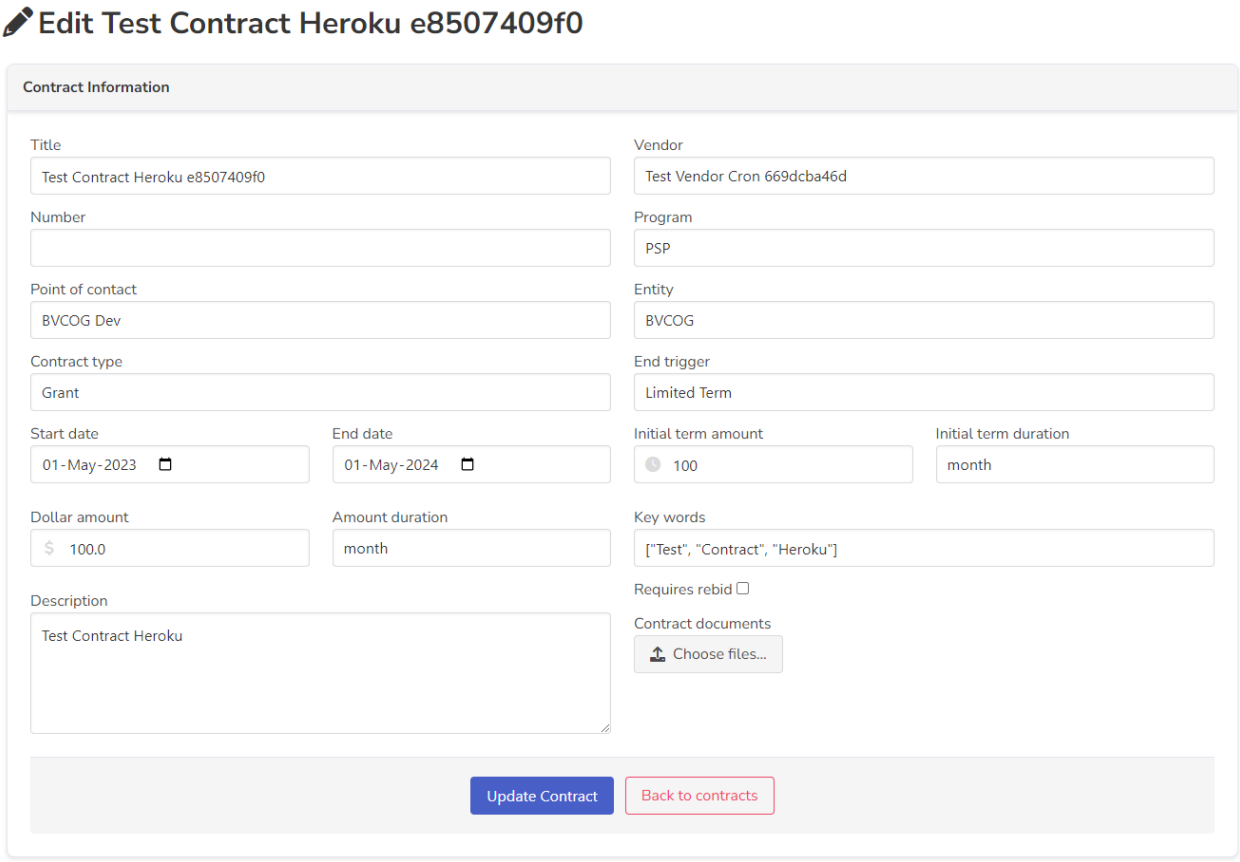
Export to PDF

Users

Name	Access Level	Status
BVCOG Dev	One	Active
Matan Broner	Three	Active
Chrysanthos Pepi	Two	Active

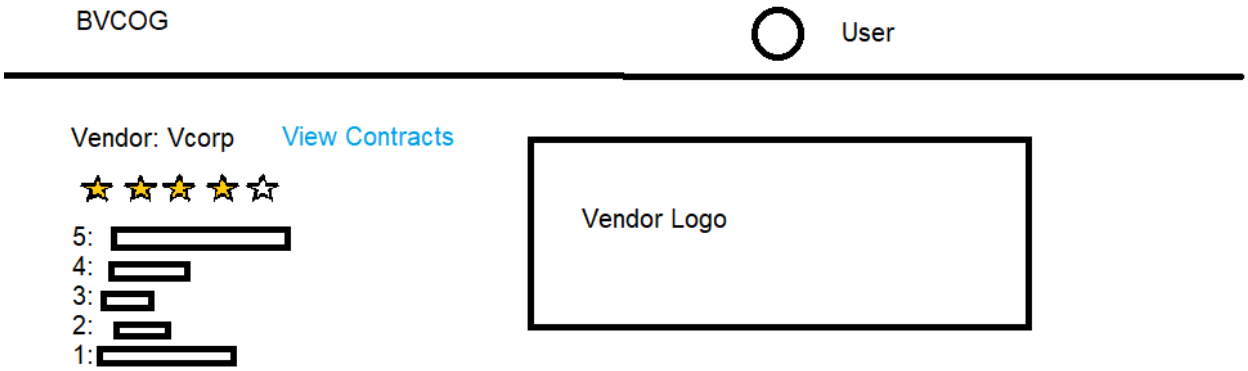
#10 | Implemented | 2

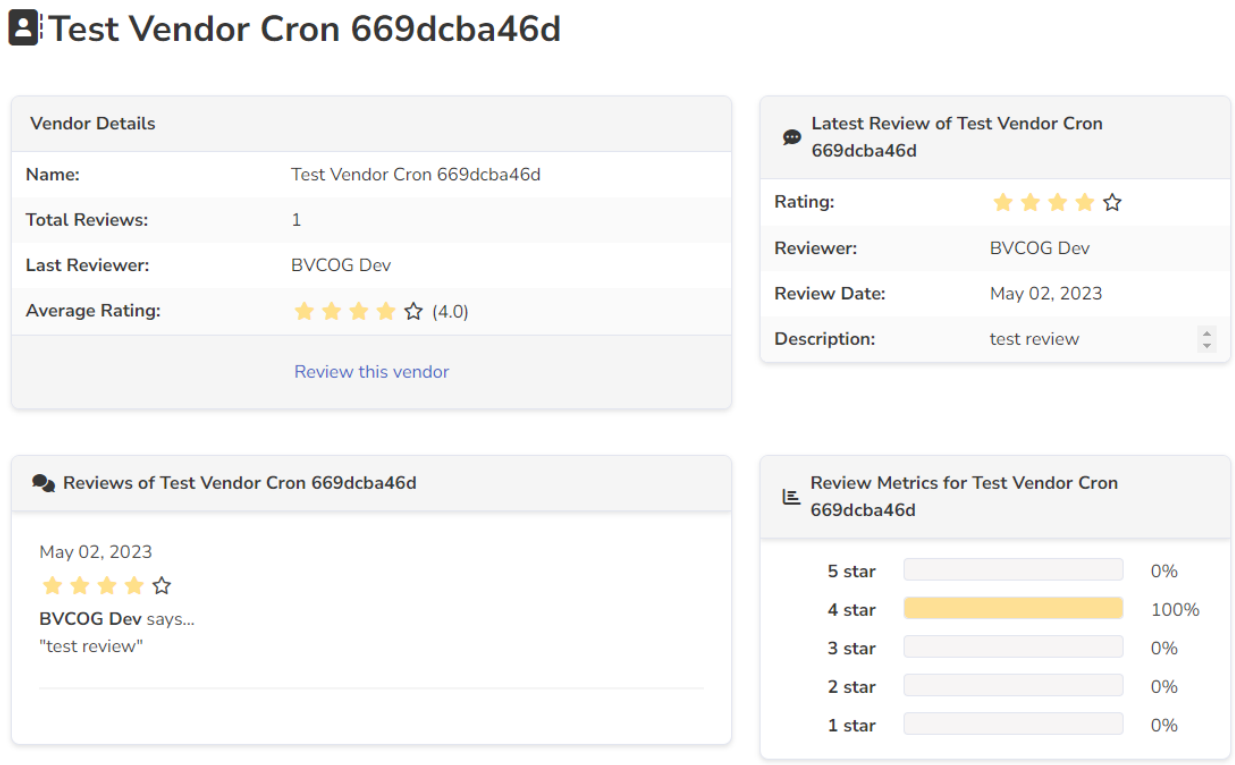
Feature: Edit a contract
As a platform user (of any level)
So that I can update a contract
I want to edit the contract fields



#11 | Implemented | ?

Feature: Amazon-style review system for vendors
As a platform user (of any level)
So that I can choose the best vendors for contracts
I want to have an amazon-style review system for vendors





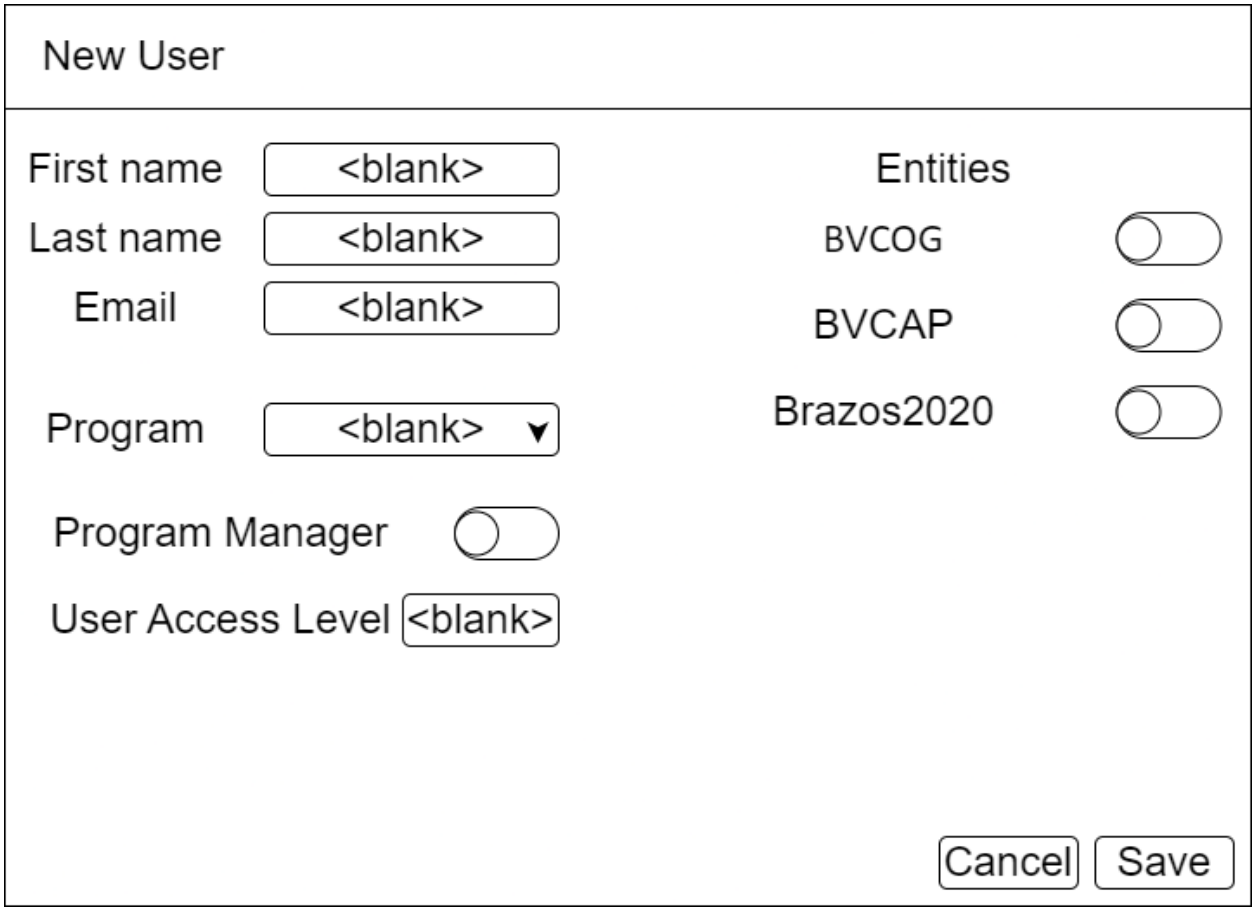
#12 | Not Implemented | ?

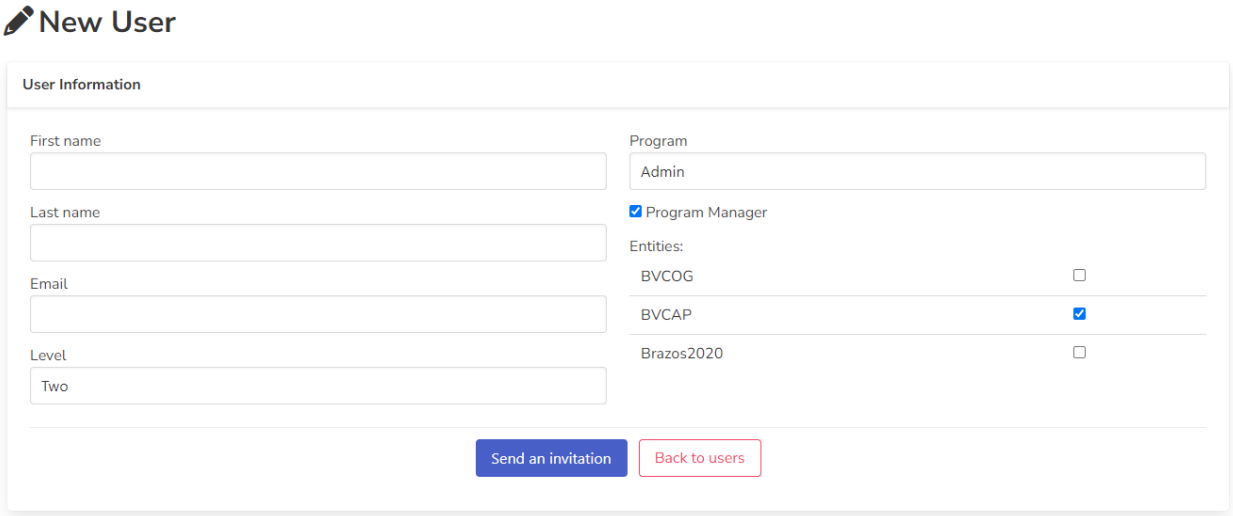
Feature: Send reminders for review

As a platform user (of any level)
So that I can review a vendor
I want to receive email notification at the end of the contract

#13 | Implemented | 3

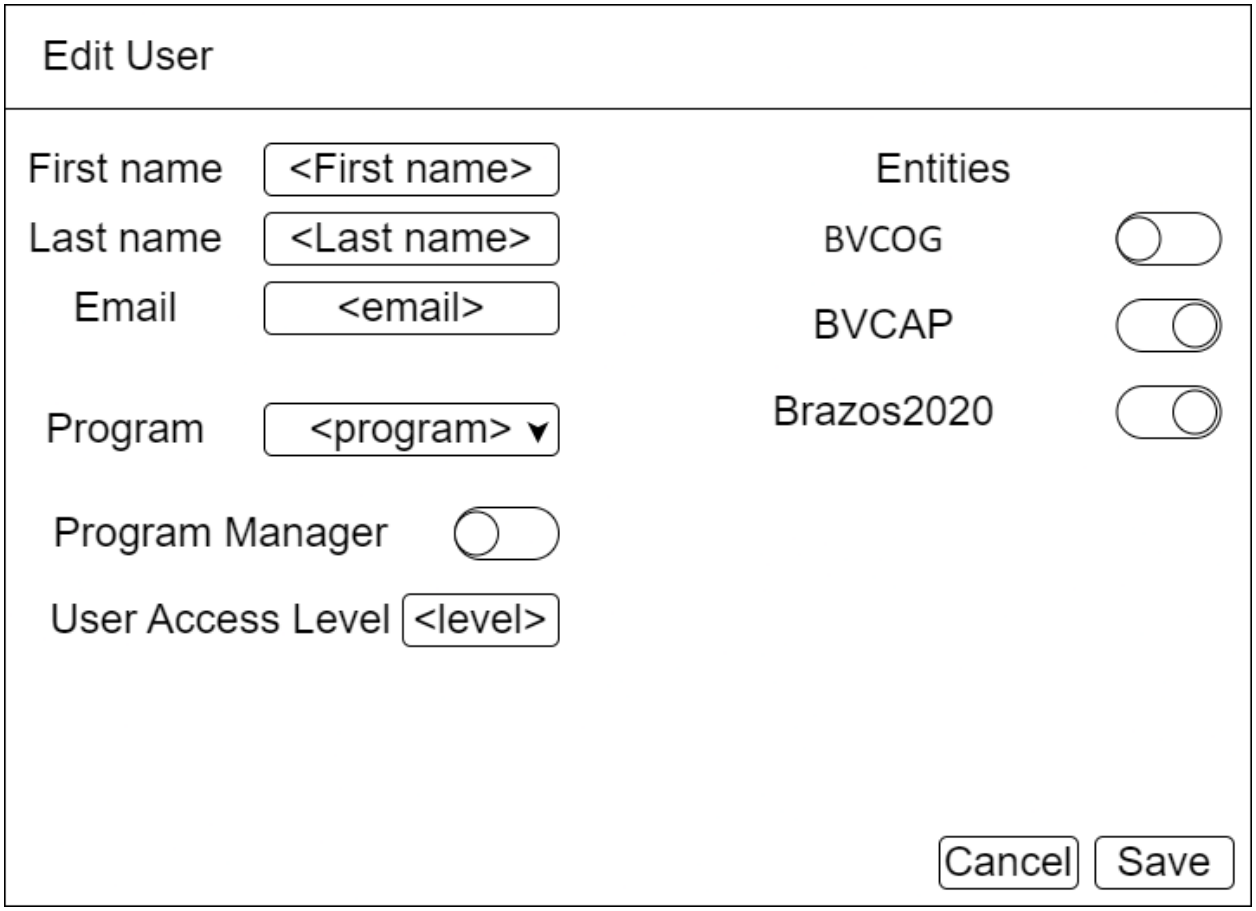
Feature: Invite a new user
As a platform user (1 level)
So that a new user can join the platform
I want to invite a new user by their email address and assign them a level

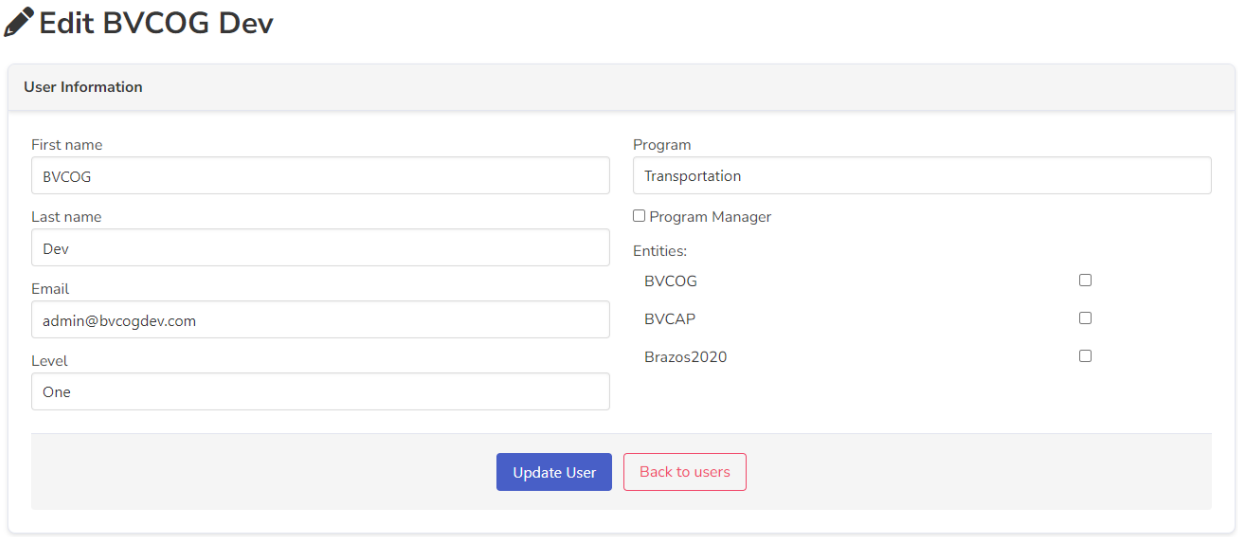




#14 | Implemented | 1

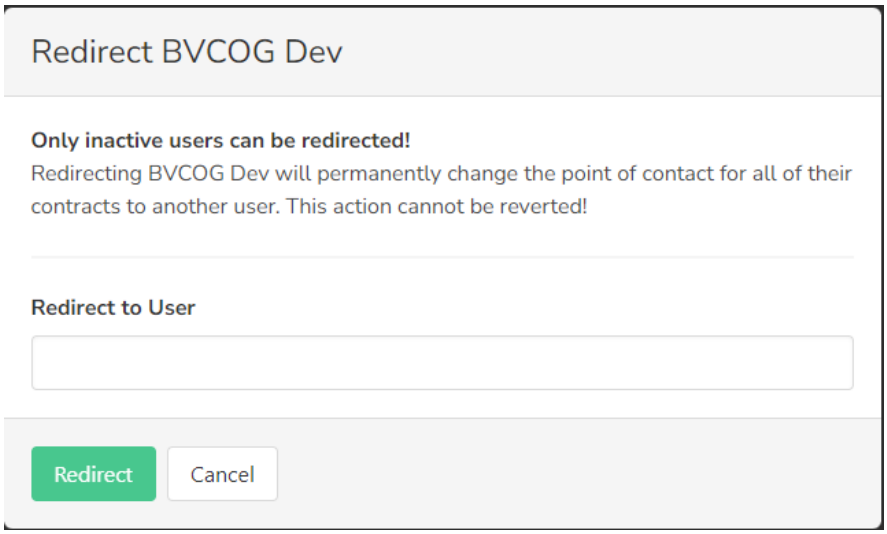
Feature: Edit user
As a platform user (1 level)
So that I can edit an active 2 or 3 level user
I want to view and edit the current values





#15 | Implemented | 2

Feature: Disable user
As a platform user (1 level)
So that I can disable a 2 or 3 level user
I want to select the user to disable and redirect the emails to another user



Scrum Iteration Summaries

Iteration 0:
Built lo-fi mockups of UI, set up meetings with customers, decided on roles, set up Pivotal Tracker and Github, Wrote up some initial user stories.
Points: 0
Iteration 1:

Planned and started to implement database models and schemas, set up github actions, got a heroku deployment going, created a pretty sidebar.
Points: 7

Iteration 2:

Added sign up / log in functionality with devise, created contracts pages, model, and controller.
Points: 20

Iteration 3:

Added mailer for reset/forget password, cleaned up UI, added tests and seed data for db tables, added users pages, added functionality for reports, initial cucumber tests, file upload, additional contract features:
Points: 35

Iteration 4:

Added automatic contract expiry emails, initial vendors UI/controller, fleshed out cucumber testing, modified automatic reports scheduling, button for sending expiry reminders, improvements to reports UI and generation, initial user invite/redirect/disable tools, initial OSO/Role based access control integration.
Points: 24

Iteration 5:

Finishing touches to project; Automatic reports of all contract data, amazon style review system for vendors, overhaul of testing to reach needed standards for rspec and cucumber, db seeding for production.
Points: 25

Meeting Dates Summaries

Our group met with 2-3 BVCOG representatives every Tuesday at 2pm. We gave ourselves a goal as a group that each week we would demo a big ticket feature in its early stages in order to collect feedback, show some refinements we made based on last week's feedback, and discuss our overall plans for the following week's demo.
For example, when we first developed the layout for displaying all contracts (ie. the contracts "index" page), we showed the initial layout of the page to get feedback, but we demoed it early since we were certain that the information the customer would want displayed in the table would change as soon as they saw it. This method of weekly meetings shielded us from too many last minute changes, as the customer got to be a part of the development process before features become too baked in and hard to change without refreshing our knowledge of older parts of the codebase.

BDD/TDD Process Summary

While we tried to follow a Test-Driven-Development process, we struggled early on due to a lack of experience with rails, and especially Rspec and cucumber. Testing early on revolved around Rspec testing, mainly on CRUD operations for our various models and controllers. As the project progressed, this testing became more fully featured. We mainly focused on writing tests that matched the specifications originally set by our clients, and during our weekly meetings with them improved on specific features. As the project progressed, more focus was put onto Cucumber testing, which had been behind. With vendors specifically, progress was increased by writing cucumber tests first based on the client's requirements, and coding to fit those cucumber tests. Overall throughout the project, because of our pressure to keep showing the clients new features, testing tended to lag behind development, but we made sure to thoroughly test our system before final submission to our client.

Configuration Management

At the time of writing: 13 Branches, 1 open, and 45 closed Pull Requests. No spikes were required. We had 2 main branches; Test, where we tested new features, and main, which we deployed into production. All other branches came off of test.

Heroku Challenges

Our application relies on storing uploaded files and auto-generated reports in the local filesystem, as this was the desire of our clients as opposed to storing files in a cloud managed service (ex. AWS S3). Given Heroku provides ephemeral dynos, the filesystem is not maintained across deployments and may even be temporarily wiped in between deployments when the dyno is reallocated. As such, we found it challenging to test our file storage features at times. Furthermore, this same issue occurred when trying to deploy CRON jobs. The only way to deploy these jobs on Heroku is through its scheduler. Given that we knew that our final production deployment would be on our client's local server, using Heroku's scheduler was a bad way to test the jobs we created with the "whenever" gem. We solved both of these problems by testing our production features that required the filesystem or CRON jobs on our local development builds and on an AWS EC2 instance.

Challenges With Tooling

We decided to work with GitHub and set up our machines to have the same Ruby version from day 1. Most of the issues we faced were mostly merge conflicts, which is common when each member is working on a different branch. Thus, each pull request before merge needed approval and the most critical one was merged first to the test

branch. Furthermore, we utilize GitHub Actions which were an extra evaluative step preventing us from merging a pull request if any test from rspec was failing.

Other Tooling

- Devise: For user authentication and password resetting
- Devise_inviteable: For user invites
- Kaminari: Table pagination
- Bulma_rails: Styling
- Polar OSO: Role based Access Control
- Whenever: Automated cron jobs

Repository Summary

We included a README into the repository that specifies the prerequisites and how to install and deploy the app.

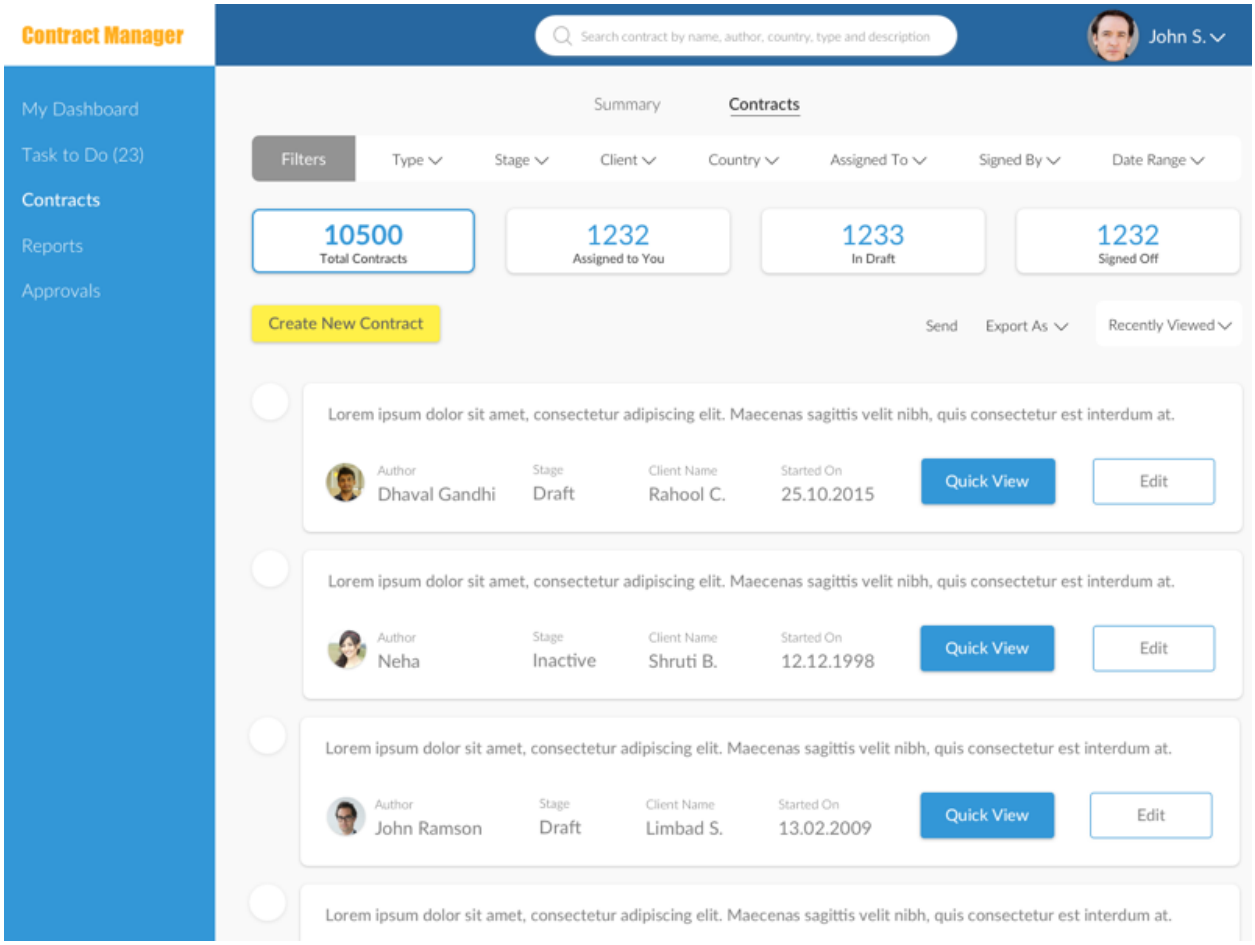
Links

- Pivotal Tracker
- Github Repository
- Heroku Deployment
- Presentation Video and Demo

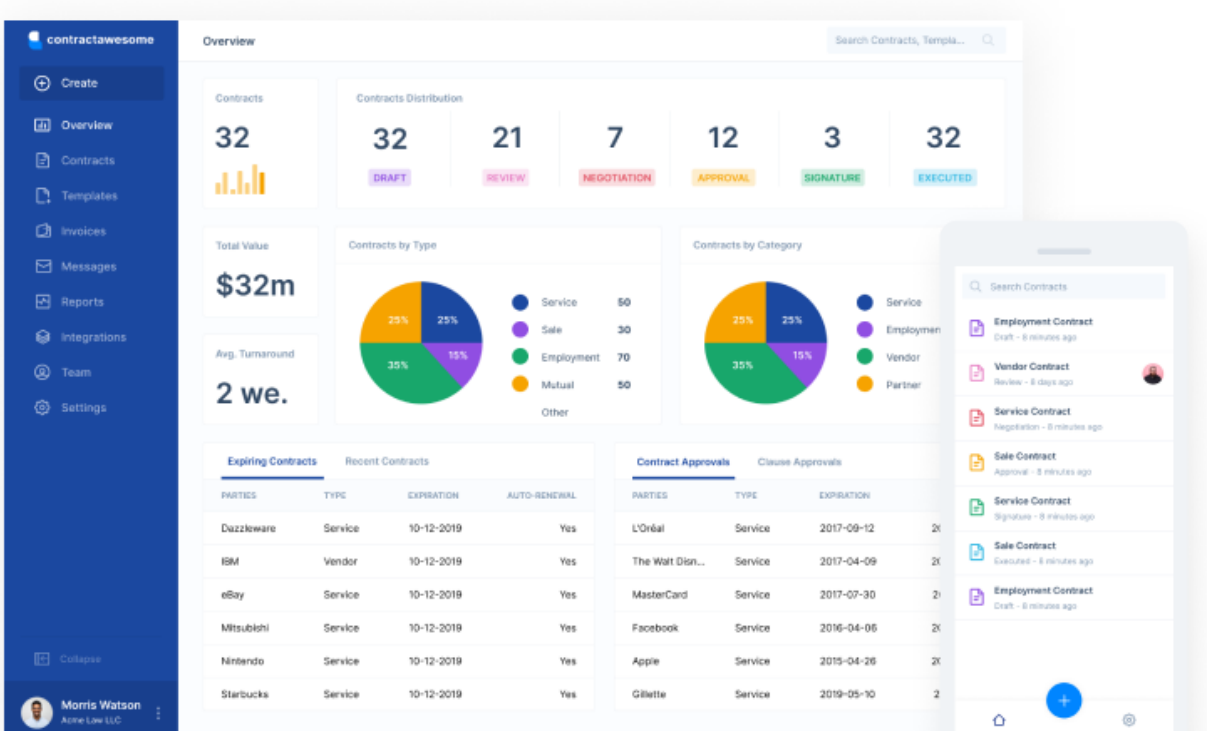
Features for future teams to implement:

- Report Tab: This tab would look a lot like the images below.,labeled "Contract UI 3 & Contract UI 1". Depending on the User's Level the Contract Total will fluctuate to only show what is within those Users limits. (ex. A Level 1 User would be able to see the grand total of contracts whereas a Level 3 User would only be able to see their total number of contracts for that program/entity). They think columns for Incomplete, Under Review, and Approved would also be necessary to show the user if they had forgotten to submit a contract for review, what contracts are still needing to be approved (hence under review), and the total amount of contracts approved (maybe not necessary). They believe a create new report button under these columns would be a viable spot. They also believe that adding an expiring contracts list to the report tab would be nice as it would just be another way to remind the user.

- Contract UI 1



- Contract UI 3:



- Approval Tab: This Tab will only be available to the Level 1 & 2 users and draws inspiration from the attachment labeled "Contract UI 1". It will only need a column for the total number of contracts needing approval. They like the idea of the way the bottom half is structured so they would like to go with that layout for the most part if possible. The quick view button should be labeled view instead. This button will allow the User to determine if the information is correct. From there they will also need an additional 2 buttons one for Approving and one for Save (to save changes without approving).
- A "Deny" button on the approval tab. In the case that the information is not complete, they would like to send it back to the submitting user with a comment.
- SSO Integration with devise; They'd like to use their own SSO system instead of using passwords, or possibly in addition to.
- Overhaul of the UI: They'd like the app to look a little more like the screenshots provided. Many changes would be needed to achieve this, such as including an image field for user creation.
- Entering Reviews at the end of a contract: After a contract ends, the point of contact will be asked through an email for a review of a vendor. Send up to 5 reminders every 4 days if the point of contact has not completed the review.