CSCE 606: Software Engineering
CASTNXT Final report

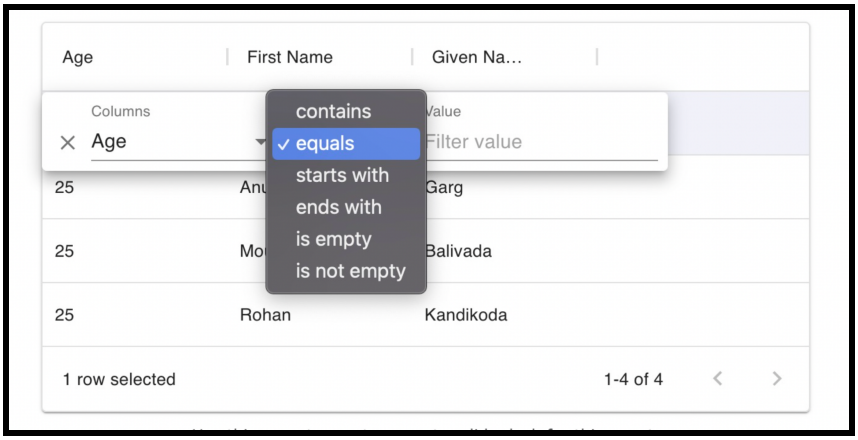
Summary:

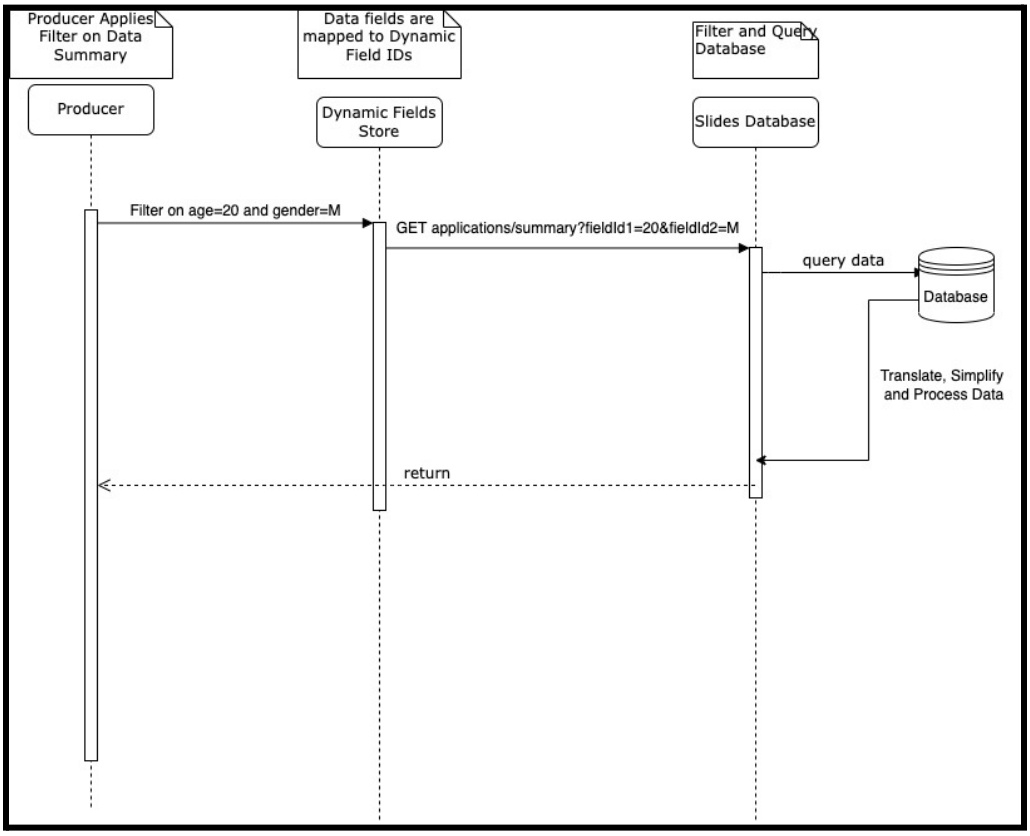
CastNXT is a web application that automates the operation of an event (a fashion show, play, or other unrelated event) that involves casting talent. Three personas are involved in this: Talent, Producer, and Client. A talent (user) can sign up for different events and keep track of the progress of their application. The producer (event manager) has the authority to pick the talent for an event, create client-facing talent lists, eliminate performers from the selection pool, and negotiate talent decks. The client can choose their top "N" preferences from the producer's talent card and then offer to negotiate based on those choices.

Our key point of contact is Tito Chowdhary owner of FashioNXT. Our stakeholders are FashioNXT as well as the models, event coordinators, and designers for the many events are all part of FashioNXT. Here, the customer's need is to streamline talent casting and client negotiations in order to automate the event management process. This includes creating lists of applicants for clients, shortlisting people for events, and back-and-forth negotiations with clients. CastNXT addresses this challenge by creating a common portal for all stakeholders that automates the event management process.

All User Stories:

1. **Feature: Filtering of attributes**
   As a producer
   I want to filter out the applicants for an event
   So that it will be easier for me to select the talent based on specific attributes.
   Points: 3
   Duration: Iteration 0 and 1
   Implementation Status: Completed
   Justification:
   In the legacy code, there was no option for filtering the applicants while in the reviewing phase, so we have created a table to enable this filtering feature for the ease of the producer. So, we have created the table with all filters applied to respective data type attributes. And on the click of the row in the table, the respective slide would be displayed below the table.





2. **Feature: Notification spotlight**
   As a user of the app I want to see a notification tab on the homepage of the web app.
   So that notifications about user registrations, new events, event due dates, etc are displayed
   Points: 3
   Duration: Iteration 0 and 1
   Implementation Status: **Icebox**, De-prioritized as Way Forward

Justification:
We have deprioritized this feature as per the client, since it involved UI restructuring and changes to DB.

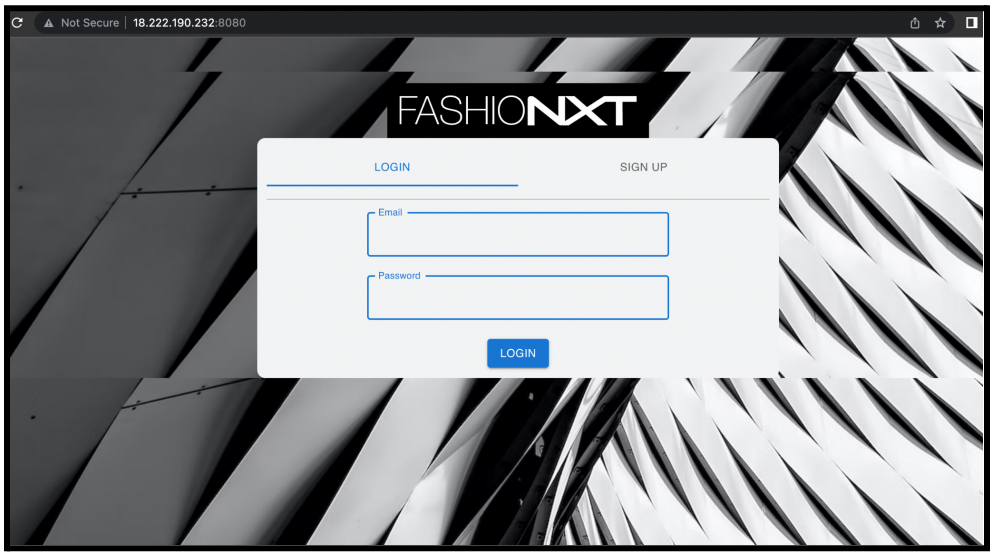
3. **Feature: Sign up email verification**
   As a producer I want to add an option of signing up through Gmail
   So that the email IDs can be verified as extra security measure before signing up
   Points: 0
   Duration: Iteration 0 and 1, implemented in 3.
   Implementation Status: Completed
   Justification/Changes to the story:
   As per client's requirement in the middle of the sprint, we have changed the story to have authentication based on normal email validation rather than a Gmail authentication. Started solution discovery and design analysis.

4. **Chore:** Setting up the infrastructure and running the previous web app on a local server and testing various features and functionalities developed by the previous teams.
   Points: 3 (Distributed amongst all team members. 0.6 per person)
   Duration: Iteration 0
   Implementation Status: Done
   Justification:
   Since this is a legacy project, we had to setup the existing infrastructure and we interacted with the previous team of Spring 2022 who worked on this project. We have been involved in understanding various features and functionalities in the app.

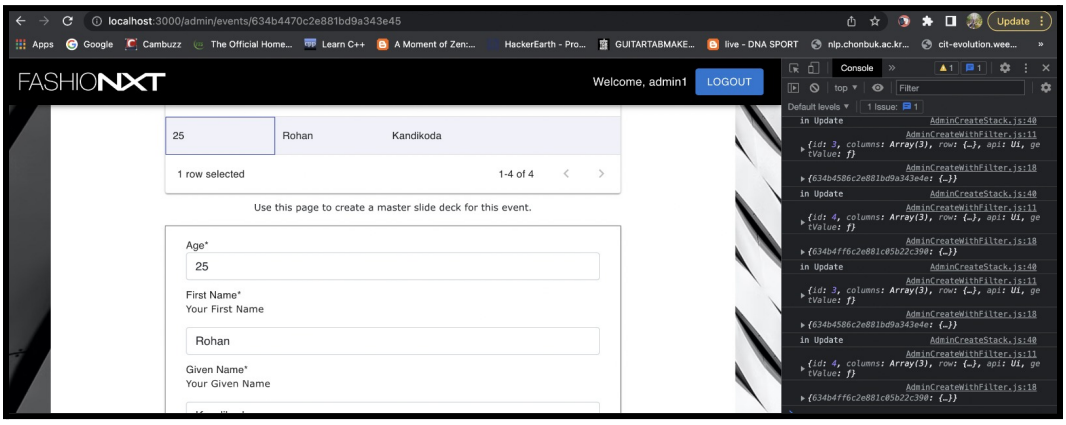
5. **Chore:** Debugging the app, discovering the bugs and trying to look at various improvements to be added to the web app.
   Points: 2 (Distributed amongst all team members. 0.4 per person)
   Duration: Iteration 0
   Implementation Status: Done
   Justification:
   This is to identify the bugs in the app. We had suggested various features and need to haves in the app to the client.

6. **Chore: AWS deployment**
   AWS deployment on EC2 instance for debugging.
   Points: 3
   Duration: Iteration 1

Implementation Status: Completed
Justification:
We have used AWS to deploy the app and test it. We have setup the whole code in AWS and have tested and run the app on AWS, resolving errors in between.



7. **Chore: Local deployment**
   Local deployment for running the app on localhost for debugging.
   Points: 3
   Duration: Iteration 1
   Implementation Status: Completed
   Justification:
   We have deployed the application locally for development purposes too.

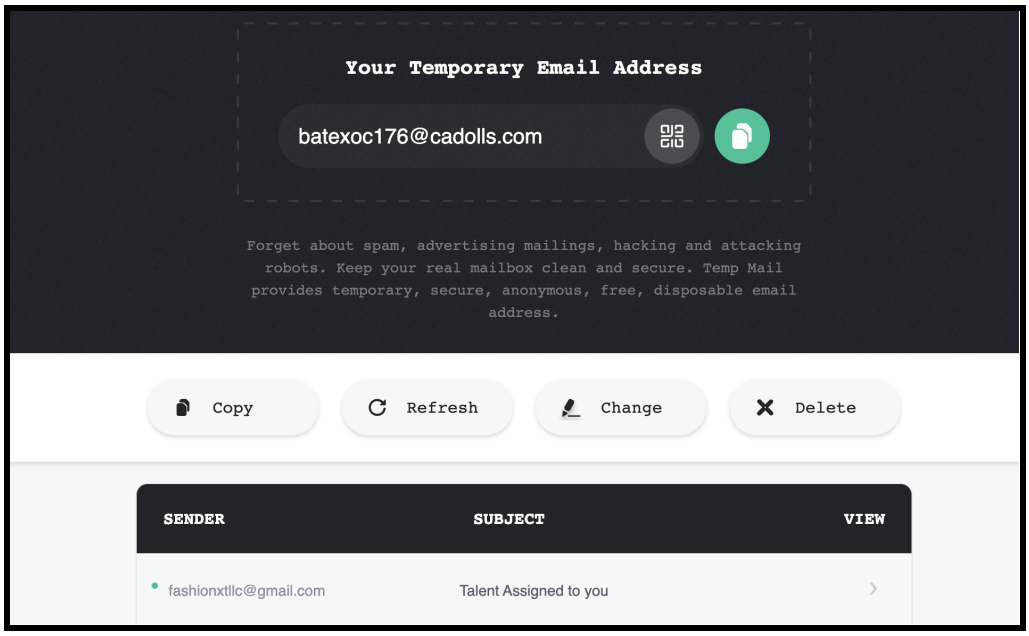


8. **Bug fix: Duplicate Column Names removal**
   Duplicate name display on event summary table.
   Points: 3
   Duration: Iteration 1

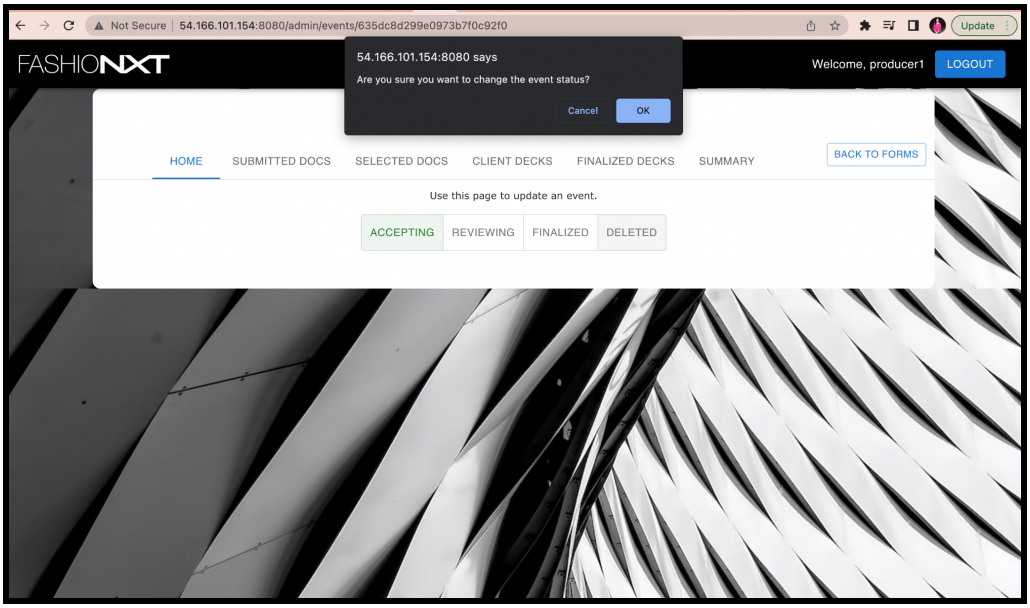
Implementation Status: Completed
Justification:
There were duplicate column names visible in the event summary table for the producer login which has been fixed now.

9. **Feature : Heroku Deployment Initial Steps**
   As a developer
   I would like to deploy the app on the Heroku server
   So that I can share the url with the client.
   Points: 3
   Duration: Iteration 2
   Implementation Status: Completed
   Justification:
   Started the Heroku deployment process.

10. **Feature : Client Assigned - Email Notification :**
    As a producer I want to send an email notification to the client when I assign a talent to them. So that the client gets an alert when a talent is assigned to them.
    Points: 3
    Duration: Iteration 2
    Implementation Status: Completed
    Justification:
    We have introduced an email notification when the producer assigns a talent to a client. The email will be sent to the email address of the client which was used to sign in the client dashboard.



11. **Feature: Delete Event**
    As a producer I want to edit or delete an already created event
    So that I have the option to edit or delete an event whenever I want to.
    Points: 3
    Duration: Iteration 2
    Implementation Status: Completed
    Justification:
    There was no option for the producer to delete the event from the list of events that are visible to the user. We have added the functionality to remove the event if the event is canceled due to any reason.



12. **Feature: Heroku deployment Database Changes**

    As a developer of CASTNXT

    I want to host the application on Heroku

    So that it is deployed on a public server and anyone can the app.

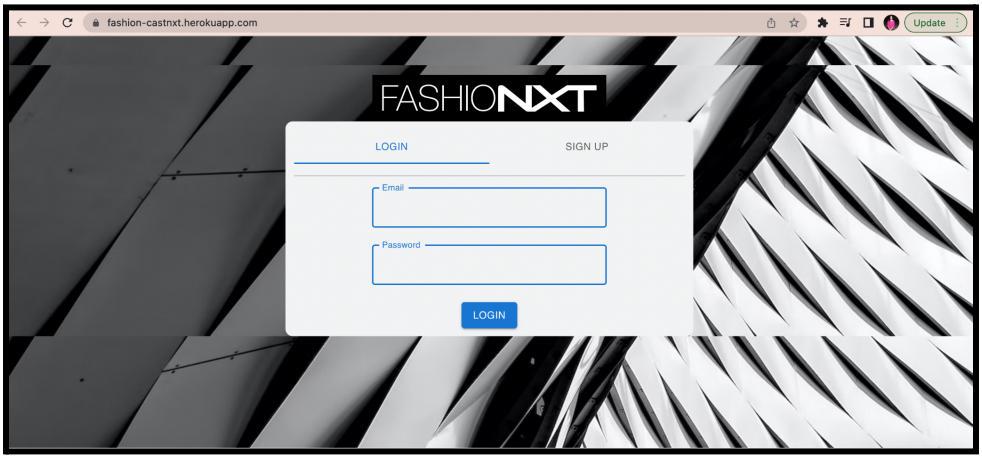
    Points: 3

    Duration: Iteration 2

    Implementation Status: Completed

    Description:

    We have deployed the application to Heroku cloud. We have created Mongo db cluster in Mongodb Atlas and connected it to Heroku deployed app. There were issues due to space and memory limitation aws. So we have upgraded tot2.small ram size.



13. **Feature : Add greater than, lesser than functionalities for the filtering feature**

    As a producer
    I want to provide filtering option to the client
    So that the client can filter the applicants based on values of their attributes.
    Points: 3
    Duration: Iteration 2
    Implementation Status: Completed
    Justification:
    Previously our application was not supporting data-type value based filtering. After
    making changes to way the data is being read and parsed we were able to provide more
    contextual value based filtering. Also we had identified many gaps with respect to how a slide was being curated after filtering. As part of this iteration we have simplified this experience and have fixed these integration gaps that existed.

14. **Feature: Signup Validation Code Implementation and testing:**

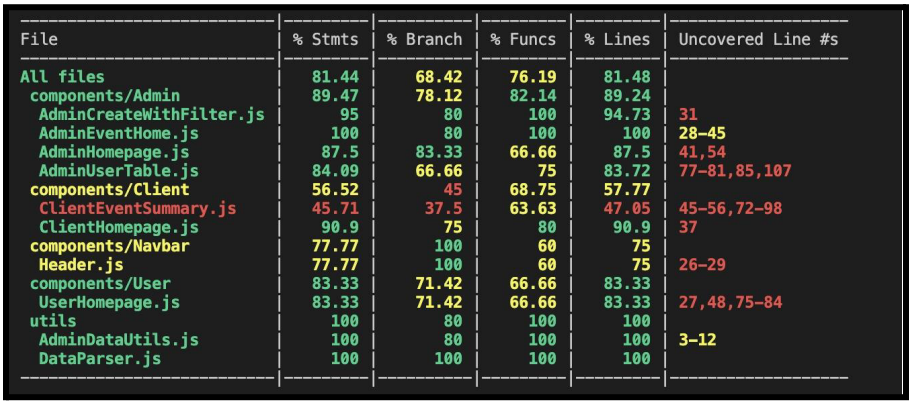
    As a product owner of the CASTNXT
    Once an email for verification has been sent, the user should be able to click the URL in Implemented backend APIs and integration with SendInBlue.
    Points: 3
    Duration: Iteration 3
    Implementation Status: Completed
    Description:
    A validation email is triggered when a user signs up for the first time. Users are not allowed to login in future unless they verify by clicking on the validation link in their mail. Once they click on the validation link, they are redirected to the login page.

15. **Feature: React js JEST testing framework integration:**

    As a developer
    I want to improve the code quality and allow coverage for Unit tests on the UI side.
    So integrated JEST unit testing framework for reactJS .
    Points: 3
    Duration: Iteration 3
    Implementation Status: Completed
    Description:
    We introduced and integrated test framework for writing test cases.

16. **Feature: React js unit tests - User:**
    As a developer
    I want to improve the code quality and the code coverage for the user login ui pages.
    So that code quality is ensured and many test cases are covered.
    Points: 3
    Duration: Iteration 3
    Implementation Status: Completed
    Description:
    We introduced jest framework for writing test cases. And have increased the coverage.

17. **Feature: React js unit tests - Admin**
    As a developer
    I want to improve the code quality and the code coverage for the Admin login ui pages.
    So that code quality is ensured and many test cases are covered.
    Points: 3
    Duration: Iteration 3
    Implementation Status: Completed
    Description:
    We introduced jest framework for writing test cases. And have increased the coverage.

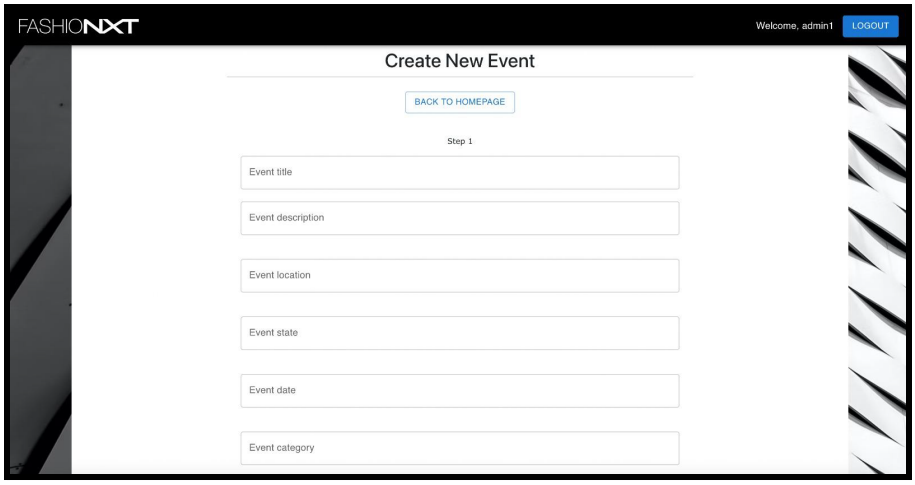


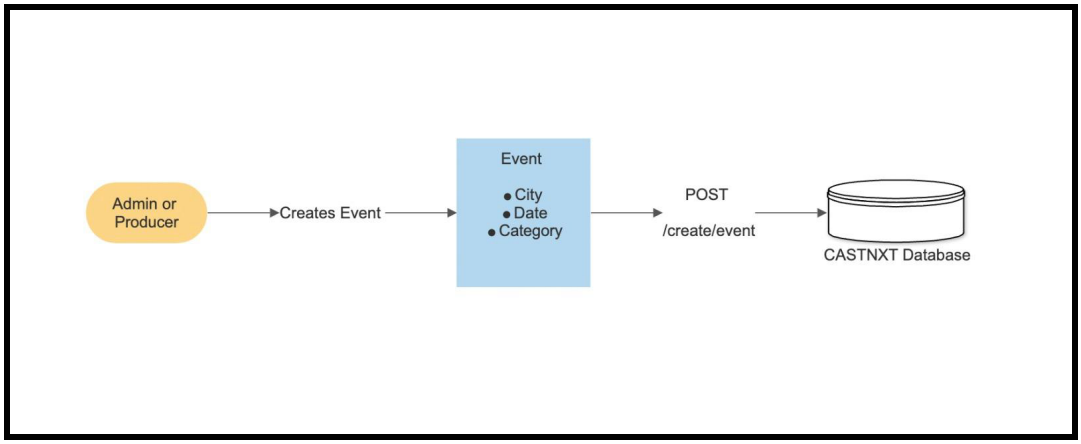
File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	81.44	68.42	76.19	81.48	
components/Admin	89.47	78.12	82.14	89.24	
AdminCreateWithFilter.js	95	80	100	94.73	31
AdminEventHome.js	100	80	100	100	28-45
AdminHomepage.js	87.5	83.33	66.66	87.5	41,54
AdminUserTable.js	84.09	66.66	75	83.72	77-81,85,107
components/Client	56.52	45	68.75	57.77	
ClientEventSummary.js	45.71	37.5	63.63	47.05	45-56,72-98
ClientHomepage.js	90.9	75	80	90.9	37
components/Navbar	77.77	100	60	75	
Header.js	77.77	100	60	75	26-29
components/User	83.33	71.42	66.66	83.33	
UserHomepage.js	83.33	71.42	66.66	83.33	27,48,75-84
utils	100	80	100	100	
AdminDataUtils.js	100	80	100	100	3-12
DataParser.js	100	100	100	100	

18. **Feature: React js unit tests - Client**
    As a developer
    I want to improve the code quality and the code coverage for the Client login ui pages.

So that code quality is ensured and many test cases are covered.
Points: 3
Duration: Iteration 3
Implementation Status: Completed
Description:
We introduced jest framework for writing test cases. And have increased the coverage.

19. **Feature: Creation of event metadata**
    As a producer of a fashion event
    I want to add event metadata to the events I create
    So that models can know more details about the event
    Points: 3
    Duration: Iteration 3
    Implementation Status: Completed
    Description:
    We added meta data for the events. Now, when the producer creates a new event, he/she is supposed to enter the meta data for the event like event location, state (venue), date of the event and the category of the event.





20. **Bugfix: Redirect from the user validation email link to the home page on the Heroku app.**
    As a user
    I want to get redirected to the login page after clicking on the validation email link in the Heroku app
    So that I can enter the login details again and enter the app.
    Points: 3
    Duration: Iteration 3
    Implementation Status: Completed
    Description:
    Once the user clicks on the validation email triggered to their mailbox. The redirection to the home page is not working on Heroku. Fixed this bug so the users can log in once sign-up validation is done.

21. **Bugfix : Heroku deployment - Mongo db integration fix issues:**
    There was a connection issue while connecting to the MongoDB atlas from the Heroku app. Resolve this issue so the users can sign up seamlessly. This built on the previous iteration's story and we were successfully able to deploy our application on the cloud!
    Points: 3
    Duration: Iteration 3
    Implementation Status: Completed
    Description:
    This bug is resolved now.

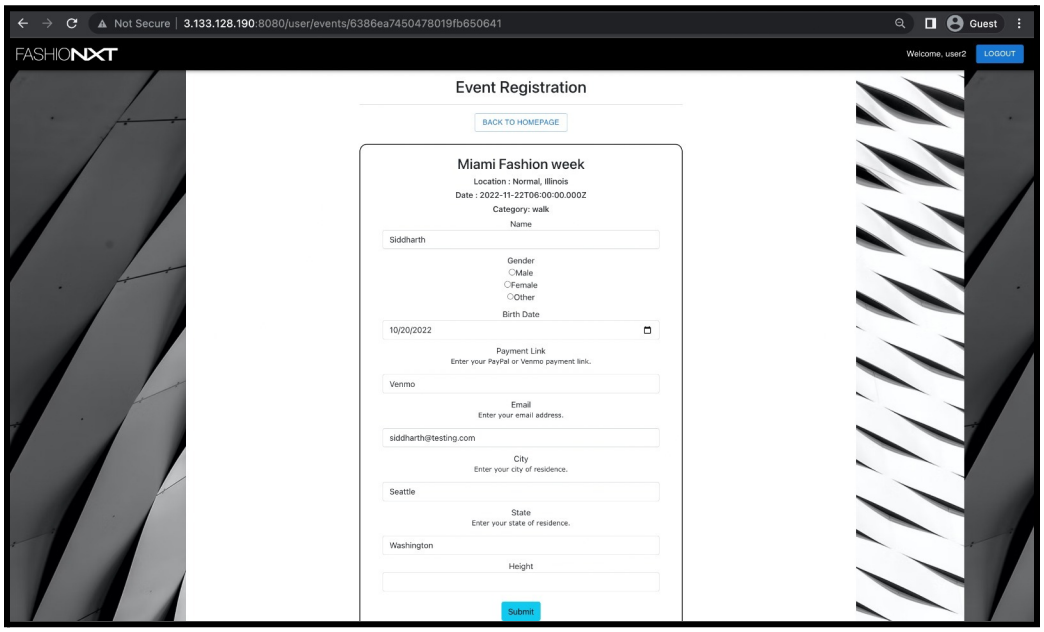
22. **Feature: Form Meta-Data - Admin:**
    As a producer
    I want to have fixed attributes/fields in all the forms
    So that the basic user details are captured always

Points: 3
Duration: Iteration 4
Implementation Status: Completed
Description:
To make life easier for admin privileged users, we have gone ahead and added some default fields for each event. This was done to support Talent MetaData that we collect in other stories and allow for auto-fill. Once an admin wants to create a new event, we add certain default fields which are referred to as the form meta-data.

23. **Feature: Talent Meta Data Backend - User / Talent**
    As a producer
    I want to have fixed attributes/fields in all the forms in the Database
    So that the basic user details are captured always
    Points: 3
    Duration: Iteration 4
    Implementation Status: Completed
    Description:
    We captured the talent meta-data such as name, date of birth, location, email, and payment link. Some of the fields were not present previously and doing so allows us to create an easier event registration flow for our talents.

24. **Feature: Talent MetaData and Auto form complete (Talent Meta-Data to UI Meta-data population)**
    As a user
    I want to have my details auto populated in the all the application forms
    So that I need not fill them redundantly
    Points: 3
    Duration: Iteration 4
    Implementation Status: Completed
    Description:

To increase event registration, we now intelligently rely on talent's meta-data to fill some basic fields. This decreases the time to register and thus gives us an opportunity to allow users to fill in more information quickly.



25. **Feature: Revamped the event meta data page**

    As a producer

    I want to have easier ways to fill the common meta data details

    So that I can have a standardized list of attributes rather than having inconsistencies. This was an extension to the above Form-MetaData story.

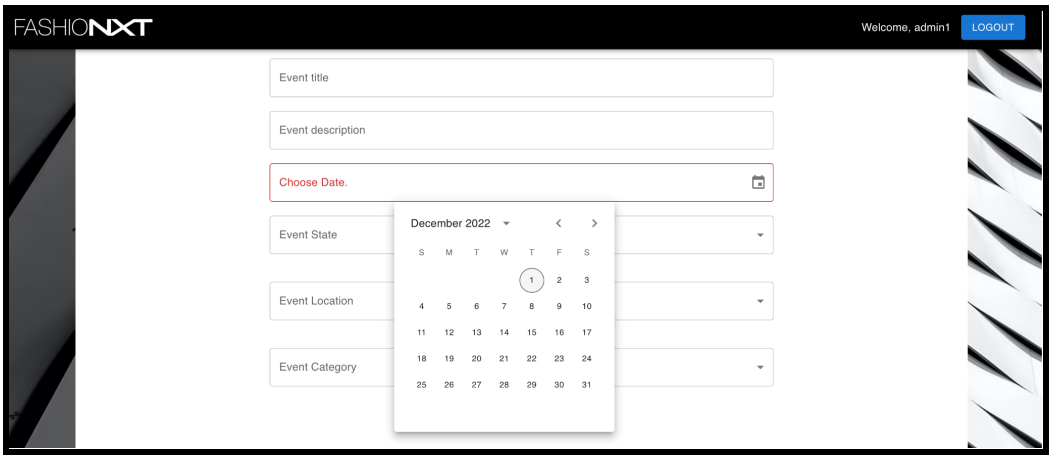
    Points: 2

    Duration: Iteration 4

    Implementation Status: Completed

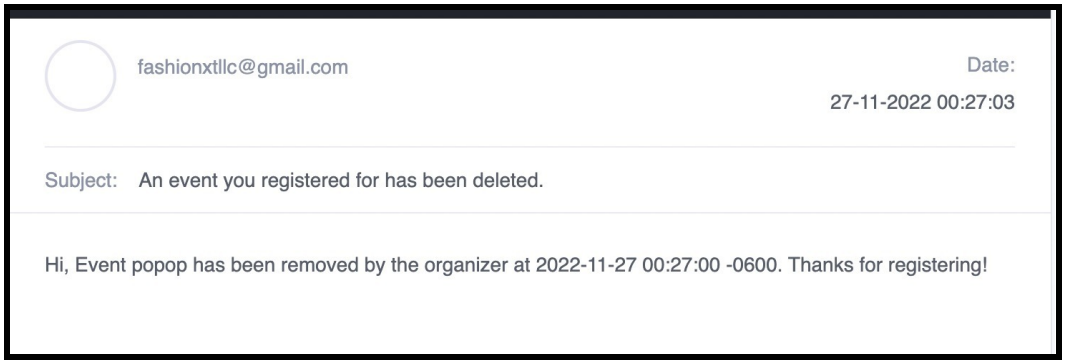
    Description:

    We have revamped the UI of the Admin create event page to include calendar view input for Date, dropdown for categories and state and city based location selection. This helps in storing the correct datatypes for these information about the event which will be helpful in the filtering of events.



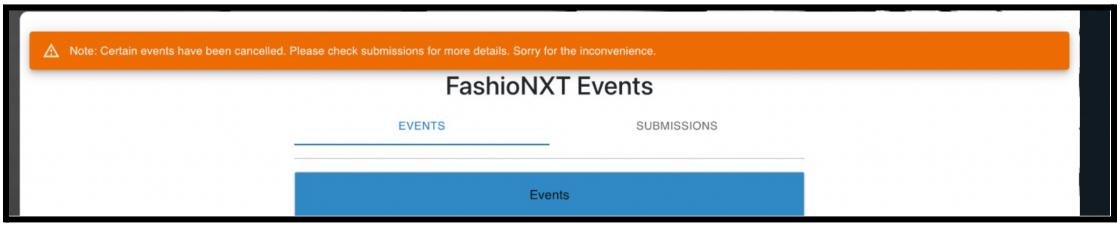
26. **Feature: Event Category based search for Talents**
    As a user
    I want to filter the events based on category
    So that I need not look at events that I am not interested in.
    Points: 3
    Duration: Iteration 4
    Implementation Status: Completed
    Description:
    To make the event search process simpler we have gone ahead and used the event meta-data to power our search. This has resulted in a more intuitive experience in which a talent can come onto our website and search events based on fields.

27. **Feature: Event Deletion Email for Talents**
    As a user
    I want to get notified via email when an event gets canceled
    So that I have the information and plan my schedule accordingly.
    Points: 2
    Duration: Iteration 4
    Implementation Status: Completed
    Description:
    In order to bridge the gap between admins and talents once an event has been canceled, we now trigger an email to notify registered talents to notify them of these unprecedented changes.



28. **Feature: Event Deletion Notification on Sign In page**

    As a user
    I want to get have a notification inside the app when an event gets canceled
    So that I do not miss the information about the event cancellation.
    Points: 3
    Duration: Iteration 4
    Implementation Status: Completed
    Description:
    Similar to the email, we now also inform the talent that said event has been canceled on their homepage. This is shown for 7 days.

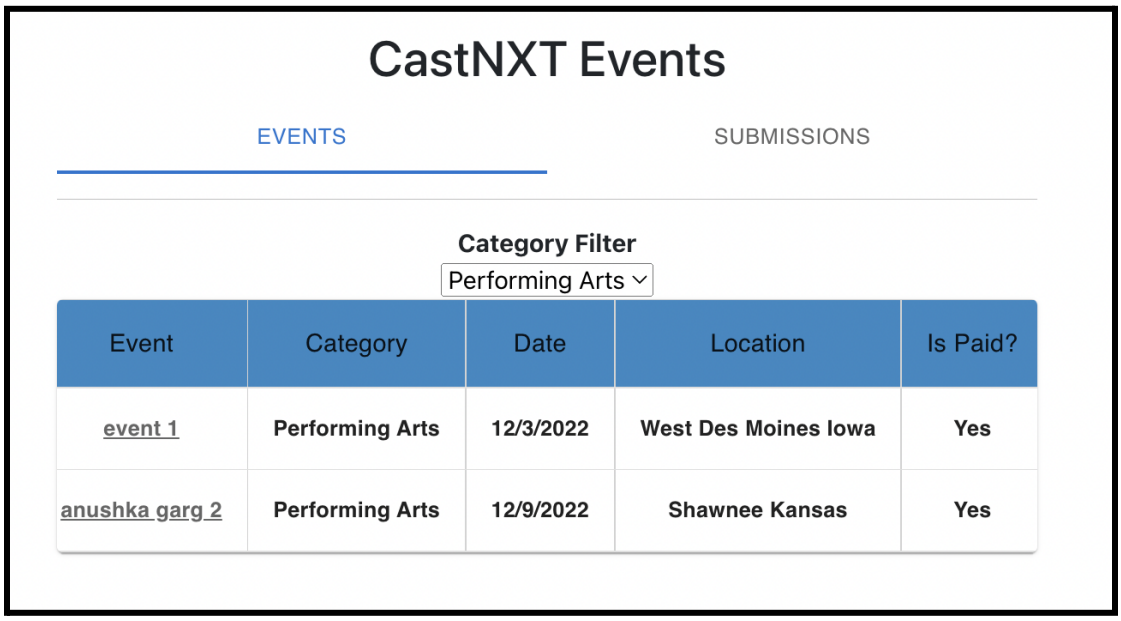


29. **BugFix: Image Column removal from event summary table**

    There is a bug in the event summary table in the producer login. Instead of having the image in the table of the talent, we get the image file path, which has to be removed.
    Points: 0
    Duration: Iteration 4
    Implementation Status: Completed
    Description:
    After working on this for some iterations, we were able to identify a fix for the problem and implement a solution that doesn't break any existing features and functionality.

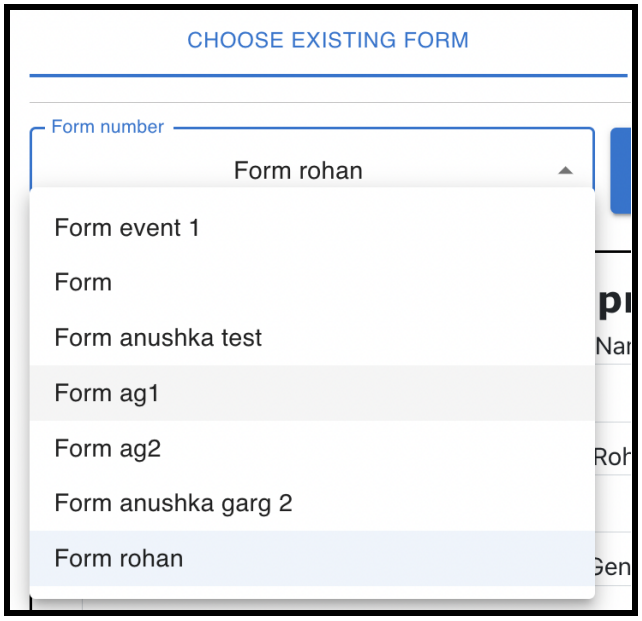
30. **Feature: User homepage event table rendering**

    As a user
    I want to look at the summary of all the events being posted
    So that I can have a quick view of what all events are being scheduled.
    Points: 2
    Duration: Iteration 5
    Implementation Status: Completed
    Description:
    When a user logs in, they see the events table. This table now contains additional fields Category, Date, Location and isPaid.



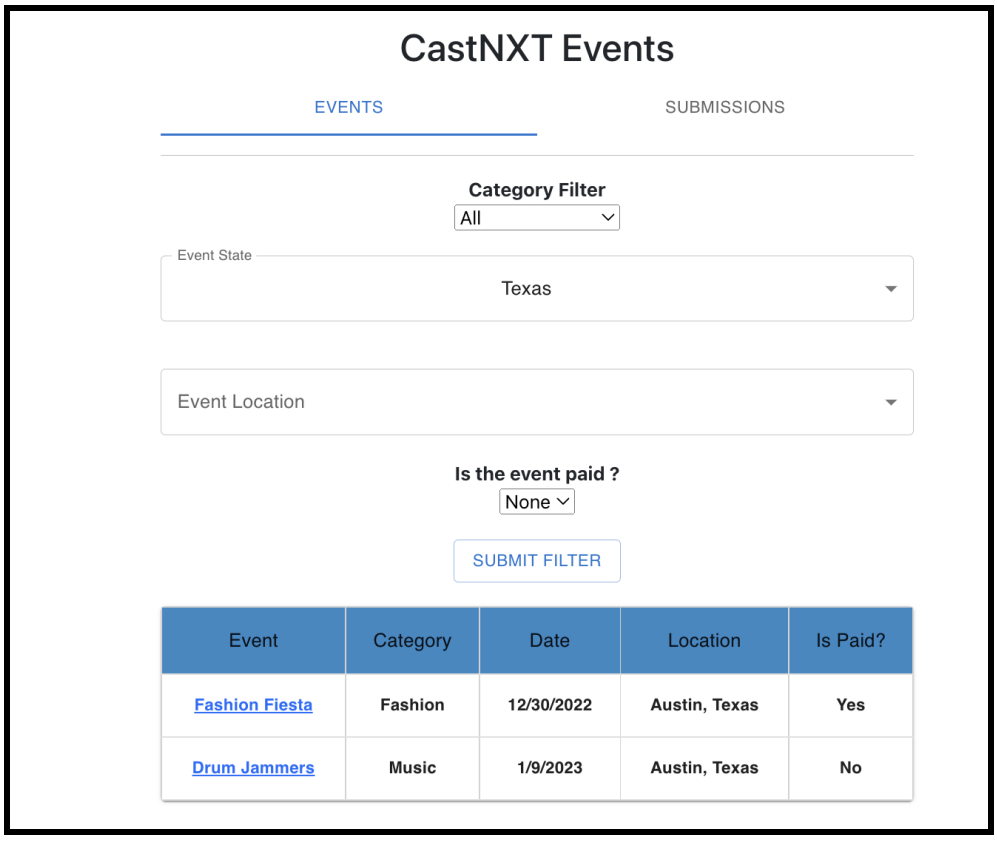
31. **Bugfix: Form template name should be given**

    When a producer creates a new event, for selecting form a drop down list of forms appears. This was previously used to show formId GUIDs. As a part of UI improvement, we have now fixed this to show the event name (with which form was initially created).
    Points: 1
    Duration: Iteration 5
    Implementation Status: Completed



32. **Bugfix: Change the title of the homepage to CASTNXT events**
    As an admin
    I want to have the correct title of the app on the homepage
    So that I can know which app I am working on
    Points: 1
    Duration: Iteration 5
    Implementation Status: Completed
    Description:
    Having consistency of the appname throughout the app.

33. **Feature: Location, Date Range and isPaid Filter for events**
    As a user
    I want to search for the events that are happening at a certain place
    So that I can reduce my search space and see only those I can register for.
    Points: 3
    Duration: Iteration 5
    Implementation Status: Completed
    Description:
    We have provided the users a functionality to filter based on the venue of the event. The user can just specify the state in which they want to attend the event rather than a specific city.

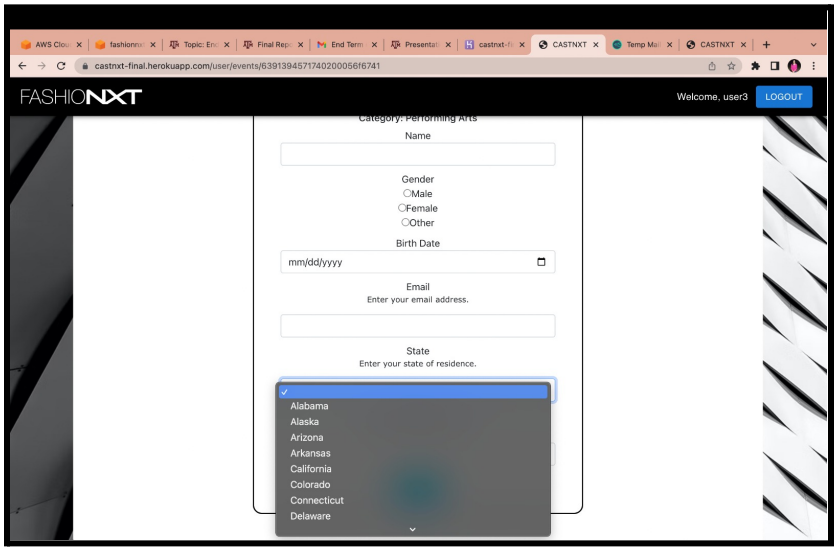


34. **BugFix: Filtering bug (change mathematical signs) for summary table**:
    As a producer
    I want to provide understandable labels in applicant filtering
    So that the clients can easily filter the applicants based on given labels.
    Points: 1
    Duration: Iteration 5
    Implementation Status: Completed
    Description:
    Previously, the filtering options were mathemcatical like >, <, >=, <=, etc which might be difficult for all the clients to understand. But now, we have replaced these with simple English labels like greater than, less than, etc.

35. **Bugfix: Payment link fix**
    There is a radio button option added in form while the producer is creating the form. If yes is selected, dynamic payment link field appears in form.
    Points: 1
    Duration: Iteration 5
    Implementation Status: Completed
    Description:

Previously, when the “Is the event paid?” question is being answered as “no”, even then a field is being populated for the users to enter a payment link which shouldn’t be asked.

36. **Feature: Adding data dropdown for states and cities for Form Metadata for user slide.**
    As a producer
    I want the users to select the city and state from dropdown list
    So that consistency is maintained.
    Points: 2
    Duration: Iteration 5
    Implementation Status: Completed
    Description:
    Previously, users had to enter their state and city, but now to avoid any inconsistencies, we have added drop-down lists for the users to fill from.



37. **Bugfix: First time sign-up app opening, app should go to login page**
    Upon sign up, user is not logged in for the first time, instead it gives a message to check email.
    Points: 1
    Duration: Iteration 5
    Implementation Status: Completed
    Description:
    Previously, when user signs up, the app opened up their dashboard and when they return again to login, that is when, user is stopped from accessing his page,

unless they are validated. But now, we are displaying an error to the user after signing up, asking them to validate first before logging in.

38. **Feature: CRM team API development**
    As a FashionNXT CRM admin
    I want to access the tables and information of CASTNXT users and events
    So that I can apply analytics on this data and gather insights.
    Points: 3
    Duration: Iteration 5
    Implementation Status: Completed
    Description:
    Added a flag for user sign-up to be controlled by CRM team. This will ensure that only the activated users are allowed to use the app. The information of CASTNXT users and events is being used by the CRM team for the analytics.

39. **Feature: Custom email for the producer to talent**
    As a producer
    I want to have a custom email functionality
    So that I can directly communicate with the talent regarding any queries.
    Points: 3
    Duration: Iteration 5
    Implementation Status: Icebox, Deprioritized due to CRM work.
    Description:
    This feature is a requirement by the client to make producer's communication with the talent easy.

40. **Quality Feature:** Ruby sanity quality checks
    As a developer
    I want to have better code quality
    So that the code is reusable and maintainable later on. This also helped us in doing some refactoring based on the reports generated from Code Climate.
    Points: 2
    Duration: Iteration 5
    Implementation Status: Completed
    Description:
    We have made Ruby sanity checks to ensure code quality. The motivation was to refactor code with some credence.

41. **Quality Feature:** Add some more javascript tests

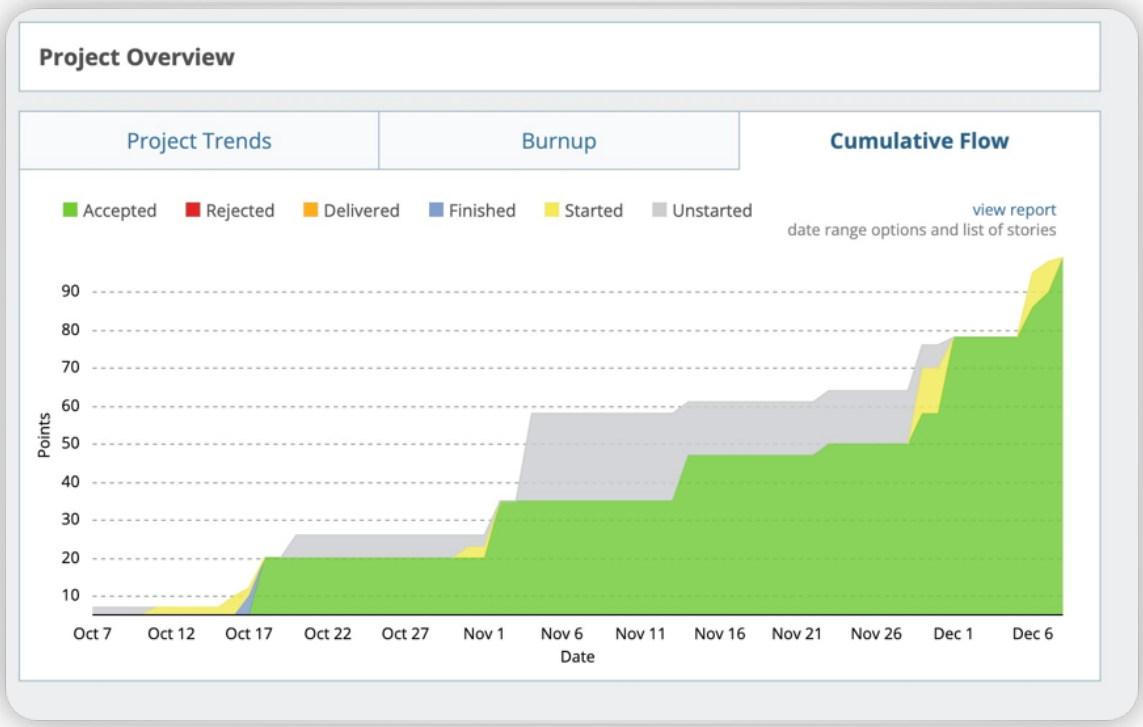
As a developer
I want to add more tests for the javascript files
So that we have more test coverage for the code
Points: 2
Duration: Iteration 5
Implementation Status: Completed
Description:
Added more java script test cases in the codebase.

42. **Chore**: **Final Report for Project**
    As a developer
    I want to document the coding, deployment and user stories
    So that the future teams can start integrating seamlessly.
    Points: 3
    Duration: After Iteration 5
    Implementation Status: Completed
    Description:
    Chore for final report.

43. **Chore**: **Final Demo Recording**
    As a developer
    I want to demo the web app's functionalities
    So that the future teams can start integrating seamlessly.
    Points: 3
    Duration: After Iteration 5
    Implementation Status: Completed
    Description:
    Chore for final demo.

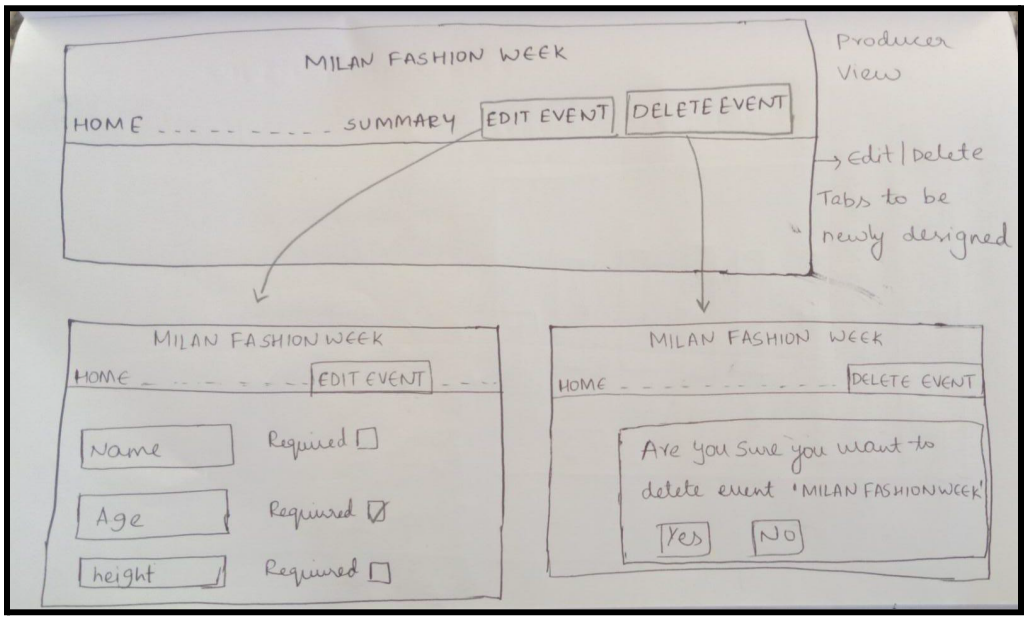
44. **Chore: Final PPT Presentation**
    As a developer
    I want to create a presentation related to the project
    So that I can explain the learnings and key features of the web app.
    Points: 0
    Duration: After Iteration 5
    Implementation Status: Completed
    Description:
    Chore for final demo.

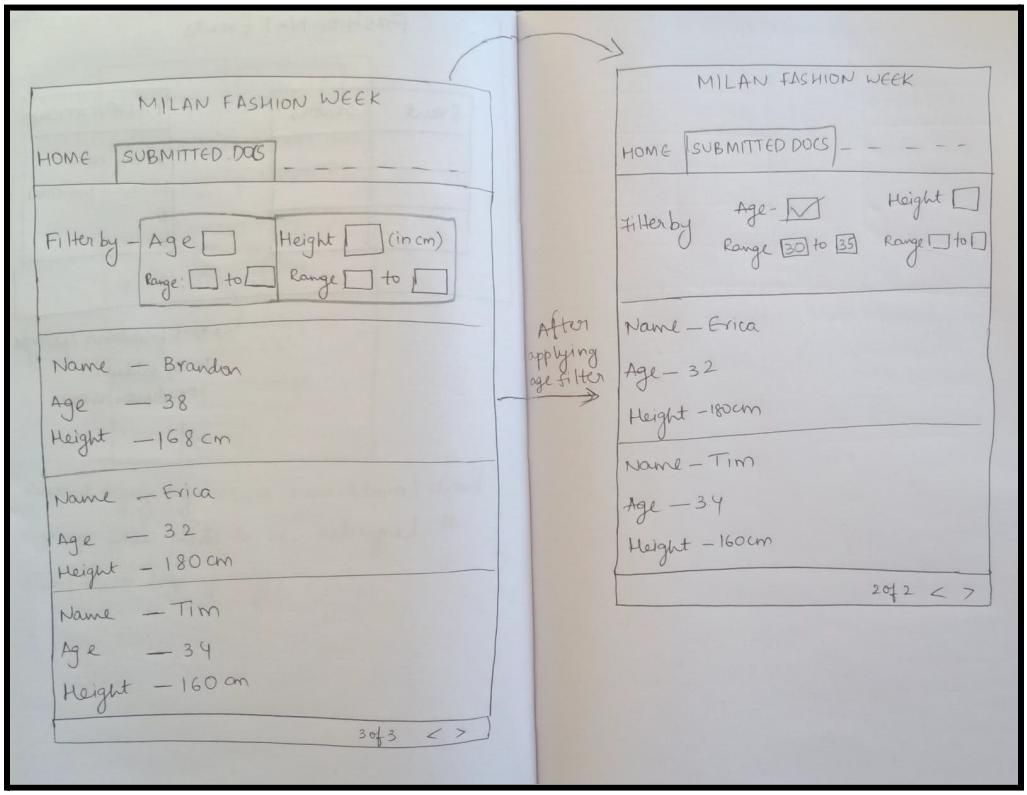
User story Burndown:



- Equal focus on backend, UI and quality.
- As requirements became clearer, code velocity increased.
- Close to 100 points of user-stories delivered.

Lo-Fi UI Mockups :





Understanding legacy code & Refactoring/Modification:

The current technology stack for the application makes use of Node.js for the UI and Rails for the backend. The backend is using Ruby on Rails framework. All the data is persisted in Mongodb.

Refactoring: We have used codeclimate tool to identify potential refactoring scenarios. We have refactored code according to codeclimate feedback. With this we have increased the readability of code by decreasing redundant lines of code.

Metadata: Previously there was no event and talent metadata for the fashion xt events in the database. We have made schema changes and introduced Talent and Event metadata. With this the talents can easily identify the events occurring in specific locations and dates. Also the talent metadata is stored in the database making the app more user friendly by auto-populating this data for second time users.

Quality Improvement: Previously there were no javascript test cases covering the ui code. We have introduced the JEST testing framework and increased the code coverage to 80%. We have also used TDD approach while doing backend development in Ruby. This helped us keep tabs on code and ensure there are not a lot of bugs and code smells. It also provided us with a continuous feedback loop that we could use to evaluate our code.

In addition to above improvements:

1. We have introduced many new features as per client requirements
2. Completed the unimplemented functionality from the previous semester

Team Roles :

We are team CoDevils : Shubham Parashar Rohan Gupta Kandikonda Mounika Balivada Siddharth Devulapalli Anushka Garg

The team roles have changed over the course of the project as per the instructor's advice in the beginning. For each iteration, there was a different scrum master and a different product owner. All of us, now, have an experience of being a scrum master and a product owner. The below are the roles for each iteration we had :

Iteration	Scrum Master	Product Owner
0	Rohan Gupta Kandikonda	Shubham Parashar

1	Anushka Garg	Mounika Balivada
2	Siddharth Devulapalli	Rohan Gupta Kandikonda
3	Mounika Balivada	Anushka Garg
4	Shubham Parashar	Siddharth Devulapalli
5	Anushka Garg	Shubham Parashar

Summaries of each Scrum Iteration:

Iteration 0:
We had interacted with the client about the expectations from the project. Majority of the iteration was to understand the legacy code and the functionalities which are already developed. We have finalized some of the requirements with the client and added those user stories in the pivotal tracker. We have added the user stories in expectation that these would be changed in the course of the project as per the priorities of the client.
Points completed: 5

Iteration 1:
We gave our progress regarding the understanding of the previous team's work and code base. We proposed some bug fixes and feature enhancements and discussed their scope and priority. Narrowed down to some important features as per client's requirements. We discussed requirements for filtering features and Data flow diagrams for proposed architecture. We discussed the notification feature in detail and decided to prioritize this over other features as per client's requirements. We have deprioritized sign up email verification to accommodate notification feature requests. We have started working on Heroku deployment, we had started designing and implementing filtering, and also designed the notification feature.
Points completed : 15

Iteration 2:
We gave our progress regarding the understanding of the previous team's work and code base. We gave a demo for newly developed features such as filtering and delete event. We proposed some bug fixes and feature enhancements and discussed their scope and priority. Narrowed down to some important features as per client's requirements. Discussed additional requirements and adornments for filtering features and Data flow diagrams for proposed architecture. Discussed further enhancements of

the Delete event, deprioritizing deleted events in the event list, and custom message for submitted participants. Discussed notification feature in detail and decided to prioritize this over other features as per client's requirements.
Points completed : 15

Iteration 3:
We gave a demo of the email Notification feature, where a client receives an email as soon as talent is assigned. We gave a demo of the signup validation feature. A validation email is triggered when a user signs up for the first time. Users are not allowed to login in the future unless they verify by clicking on the validation link in their mail. We proposed some bug fixes and feature enhancements and discussed their scope and priority. Narrowed down to some important features as per client's requirements. Discussed additional requirements and adornments for filtering features and Data flow diagrams for the proposed architecture. Discussed further enhancements of the Delete event, deprioritizing deleted events in the event list, and custom message for submitted participants.
Points completed : 15

Iteration 4:
We discussed features based on new fields that we plan to add in the database for Events, and Talents table. We also demoed features developed in the previous iteration where we have added some fields to the event table. Client had some feedback pertaining to developed features. We demonstrated the changes made to the talent table and how it has been implemented. This has helped in auto filling of fields for form submission. We showed the delete event notification and email in practice. Discussed the Heroku student account and changes. Discussed further enhancements of the search event features and also some more modifications to the final submissions table on the event side.
Points completed : 25

Iteration 5:
Event filters were demonstrated, Client suggested a better UI , all filters in one line along with the submit button.Client discussed the effort required for the EVENTNXT team to help the CRM team. Send an email about integration with the EVENTNXT team and adding client in cc. To be added in wayforward: Notification feature: addition, updation and deletion of slide from the deck, there should be some email notification to the client. To be added in wayforward: Payment Link feature should be there. All lessons learnt from the project were discussed with the client. We had suggested features and ideas for the future teams to develop. We have highlighted some of the technical issues and bug fixes which the future teams should be working on.

Points completed : 25

Meetings with the client:

Throughout the course of the semester, we met the client every week on Wednesdays on Zoom platform. General format of the meeting -

1. Discussing the work done in the week by the product owner
2. Showing demo of individual developer's work
3. Asking the client's feedback on the features developed so far
4. Client suggestions on the next week's work (prioritizing features)

Meeting 1:
Date : 22nd September 2022
Time : 7pm-8pm CT

Minutes of the Meeting:

1. Team introductions with the client
2. Client has given overview of what the project is about.
3. Discussed the legacy changes of the app over the course of last 2 semesters
4. Discussed the high level requirements.
5. Meet the previous team for the webapp setup.

Meeting 2:
Date : 28th September 2022
Time : 7pm-8pm CT

Minutes of the Meeting:

1. High level requirements of the web app were discussed by the client.
2. We gave our progress regarding the understanding of the previous team's work and code base.
3. Negotiation of feature development (prioritizing)
4. Discussed additional requirements and adornments for filtering feature and Data flow
   diagrams for proposed architecture.
5. Discussed a few bugfixes we have observed in the webapp and added them to backlog
6. Suggested some features which will be helpful for the client and the web app.

Meeting 3:
Date : 19th October 2022

Time : 7pm-8pm CT

Minutes of the meeting:

1. Validation of the signup needs to be done.
2. If a curated deck is given to the client, and then if a producer adds or deletes a talent from the deck, then an email should be sent to client informing them the change.
3. Sorting can be added to the filters and ranges also can be added to the filters.
4. Once a model is finalized, his/her details should not be visible on “Selected Docs” . This bug needs to be fixed.
5. Discussed further enhancements of the Delete event, deprioritizing deleted events in the
   event list, and custom message for submitted participants.

Features demo:

1. Filtering of attributes is shown.
2. Took inputs from the client regarding the filtering feature

Meeting 4:
Date : 26th October 2022
Time : 7pm-8pm CT

Minutes of the meeting:

1. Email notification feature is in progress.
2. Search for events based on location, category and dates is to be prioritized the most.
3. To look at the apps in the market which have this similar search feature.
4. To add talent metadata, so that whenever a talent wants to fill a form, their information is auto populated rather than filling the info multiple times.
5. Producers should be able to filter the applicants based on these meta data.

Features demo:

1. AWS deployment shown.
2. Heroku deployment defects discussed.
3. Delete event feature demonstrated.

Meeting 5:
Date : 2nd November 2022

Time : 7pm-8pm CT

Minutes of the meeting:

1. The same email service needs to be used by all teams. Contact other teams to know what service they have been using.
2. A feature to add multiple photos of talent by producer or client needs to be added to create a portfolio of the user.
3. Sizes of the files being uploaded by the talent is also important. Producer should decided whether they want low or high resolution files for their event.

Features demo:

1. Email notification when client is assigned
2. Signup validation for any user, producer or client.

Meeting 6:
Date : 9th November 2022
Time : 7pm-8pm CT

Minutes of the meeting:

1. The columns in the event summary table should be editable so that the users' payments links can be added or edited and hence the payments can be made easily.
2. Categories should include Fashion, Performing Arts, Music and others.
3. We have agreed to proceed with the payment links to be visible to the producer when they are accessing the Event Summary Table.
4. Work on filtering of events and delete notification is being started.

Features demo:

1. Event metadata UI and mockups shown and finalized.
2. Event metadata backend work is done.

Meeting 7:
Date : 16th November 2022
Time : 7pm-8pm CT

Minutes of the meeting:

1. For event data, we have agreed to add the dropdown list of states and cities in the US from the internet.
2. There has to be a certain feature for email communication between the producer and the talent. When a producer has certain doubts about a talent, then the

producer should be able to send a custom email to the talent from the dashboard itself.
3. Search bar functionality can have keyword searches too.
4. There is a bugfix related to the form template name. We will be working on that in the next iteration.

Features demo:
1. Heroku deployment is being shown to the client and the link of the web app is given.
2. Test cases for UI have been added by introducing the JEST framework for React tests.
3. Event metadata UI is being shown where the users can see the information of the event in the event details page.

Meeting 8:
Date : 30th November 2022
Time : 7pm-8pm CT

Minutes of the meeting:
1. We need to add event time when creating a new event by the producer.
2. We will be adding the state and city dropdowns for the user when they are filling the form.
3. If there is any change in the slide decks, then the client should be notified about the same via email.
4. Form meta data is being captured now. When the producer is creating a new event, in the application form, some field names are pre-populated and are compulsory to be present in any application form created.

Features demo:
1. Event metadata revamped with the dropdowns and calendar views.
2. Delete event email and notification feature is demonstrated.
3. Form metadata feature shown.

Meeting 9:
Date : 7th December 2022
Time : 7pm-8pm CT

Minutes of the meeting:
1. Talent meta data is being shown and the client suggested adding a profile page where this information is being displayed and should be editable.

2. The client has asked for a better UI of the filters, making them horizontal and intuitive.
3. We have to give the information about our database to the EventNXT team for their development and integration with the CRM team.
4. The client suggested we have a notification feature for updation, deletion and addition of a slide in the deck.

Features demo:

1. Event filtering based on category, location and isPaidEvent, is being shown.
2. Talent metadata is being automatically filled when a user is submitting another form.
3. Signup page now stops the person from entering the dashboard before validation.
4. Integration with the CRM team by creating a flag about the user activation.

BDD/TDD Process

Process:

There was no testing framework for frontend files in JS. So, we have introduced the JEST framework to write the unit test cases for frontend files. The majority of our work is written in Javascript. Almost about two-thirds of our code base is in JS, so we introduced that framework. We have added test cases to the Ruby files on top of the previous team's test files using their testing framework RSpec.

Challenges:

1. Since the majority of source code is in JavaScript, ReactJS to be more specific, it was important to add a testing framework that would make code readable.
2. When we started, we had a lot of legacy code, writing test-cases for 1000s of lines of code was hard and quite complicated
3. This meant that a proper code clarity was needed to add test cases for previously existing files.
4. Mocking user events was also a challenge, as we had to be accurate in our assertions and how events would occur on the webpage.

Benefits:

1. We had to make sure that the controller responded in accordance with each user's level of authorization because some of our endpoints were accessible to several user personas. Writing controllers that were aware of authorization levels

was made simple by having unit tests that were verified based on the authorization of each persona.
2. Unit tests helped front-end developers during integration by providing them with direction. Front-end developers were able to determine the parameters that were required to be sent to each endpoint by consulting unit tests.
3. Writing ReactJS tests gave us a very good picture of what had been done by the previous team and how it was implemented. Following TDD, helped us learn from previous mistakes and not repeat them.
4. Since ReactJS code is supposed to be performance-oriented, we were able to simplify code and reduce file size, eventually resulting in smaller bundle sizes.

Configuration management:

MongoDB (Backend):

In order to use MongoDB as the backend, 'mongoid' ruby gem was utilized. All the mongodb configuration is maintained in mongo id. yml for all the environments (development, test and production)

The deployment requires a connection to MongoDB Atlas. Heroku Container deployments do not allow MongoDB within containers (Due to this, we also do not have an idea about the persistence of data). To resolve this we made use of a Free MongoDB Atlas cluster for our database.

Spike:

1. Heroku deployment issues were observed due to mongo db network setting and database name issues. We have tried out with various configurations and configured heroku url to access mongo db cluster.
2. Experimented with JEST framework which was not existing before.
3. Finding bugfixes in the already existing legacy apps by experimenting and checking various features.

Number of Branches: 40 branches were created for different features, bugfixes and spike stories. And we had merged 35 PRs to main, in total.

Number of releases: We had 5 releases, one for each iteration.

Deployment Guide:

AWS local deployment steps:

Clone the repository:

```
git clone git@github.com:tamu-edu-students/CASTNXT.git
cd CASTNXT/web/CASTNXT
```

Install Mongo DB:

```
sudo tee /etc/yum.repos.d/mongodb-org-5.0.repo<<EOL
[mongodb-org-5.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/amazon/2/mongodb-org/5.0/x8
6_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-5.0.asc
EOL
```

```
sudo yum install -y mongodb-org
sudo systemctl start mongod
```

Install ruby:

```
rvm install "ruby-2.6.6"
bundle install
```

Install node and npm:

```
npm install -g npm@8.5.4
nvm install 16.13.0
npm install -g yarn
bundle exec rails webpacker:install
```

Start the application:

```
rails s -p $PORT -b $IP
```

Local deployment:

Local Deployment not done by prev team. Our team was able to locally run the code by following same steps as above. For Mongodb we have started mongo as a service in local and connected application to local mongo db.

In case there are issues with setup of Ruby, please refer to the links below:

1. https://stackoverflow.com/questions/10940736/rbenv-not-changing-ruby-version
2. https://dollardhingra.com/blog/install-ruby-old-versions-macm1/

Heroku deployment steps:

Clone the repository:

```
git clone git@github.com:tamu-edu-students/CASTNXT.git
cd CASTNXT/web/CASTNXT
```

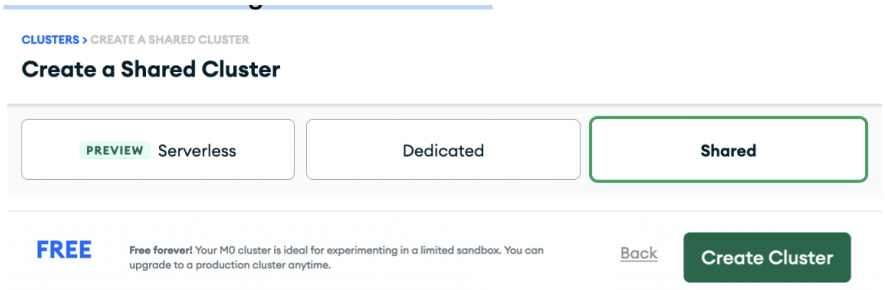
Connect to Heroku and Create a Deployment

```
heroku login -i
heroku container:login
heroku create -a <app
```

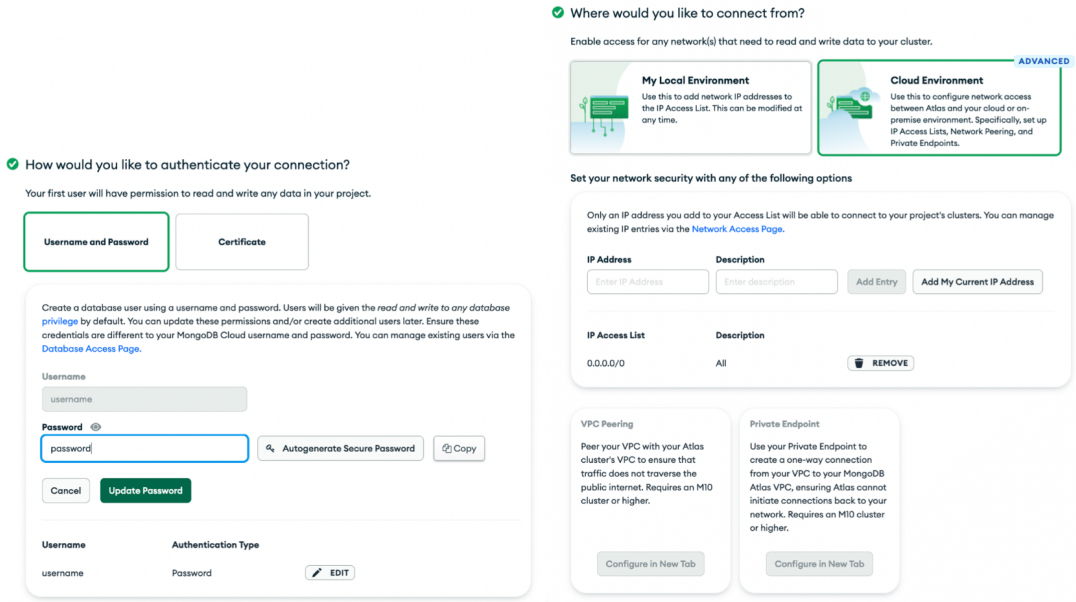
Push code into the Deployment

```
heroku container:push web -a <app_name>
heroku release web -a <app_name>
```

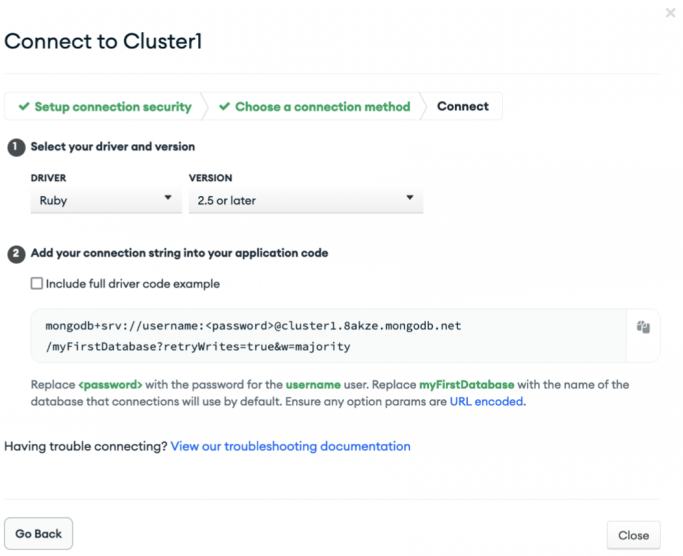
Create a Free MongoDB Atlas Cluster



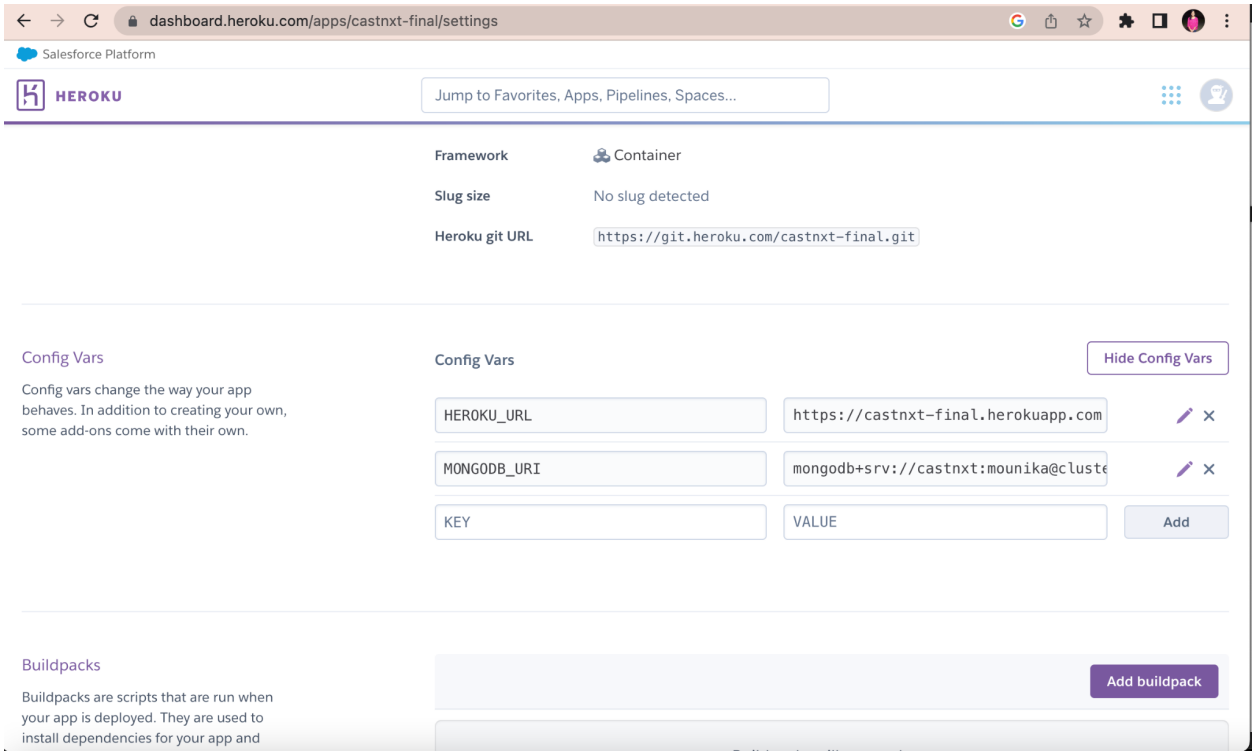
Configure the Free MongoDB Atlas Cluster under Security Quickstart



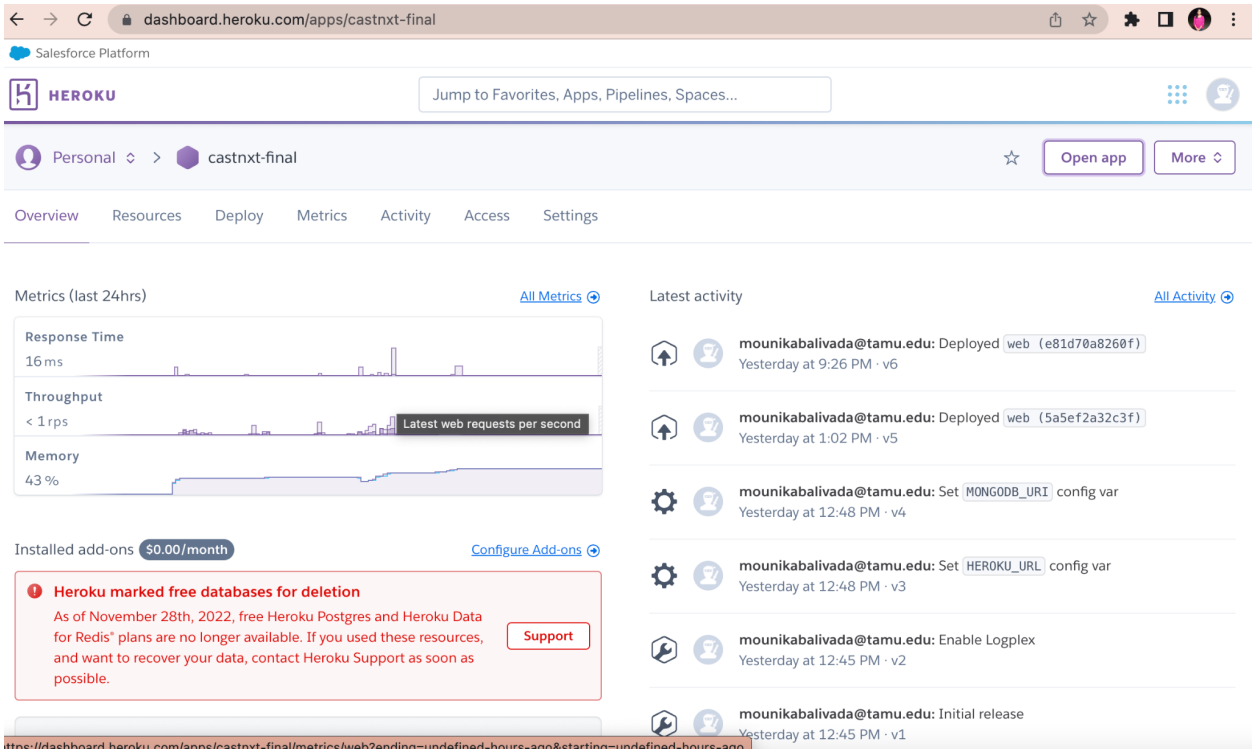
Get MongoDB Atlas Connection URL: Make sure to add the database name in the URL while configuring the Mongo DB URL.



Connect Deployment to MongoDB Atlas on Heroku.com



Open App to view the Deployment



Issues while doing Heroku deployment:

1.We faced connection issues from Heroku app to Mongo db cluster. The databasename is not included in the connection string by default.

2.From Heroku settings we have to add config variables for example MONGODB_URI

In our case, the connection string is :

mongodb+srv://castnxt:<password>@cluster0.9mfb0kl.mongodb.net/castnxttest?retryWrites=true&w=majority

3.To resolve this, the database name and password must be given in the connection string.

Issues in AWS cloud9 deployment :

AWS Free tier offers 10GB Storage and 1GB RAM. There are some issues that arise due to this:

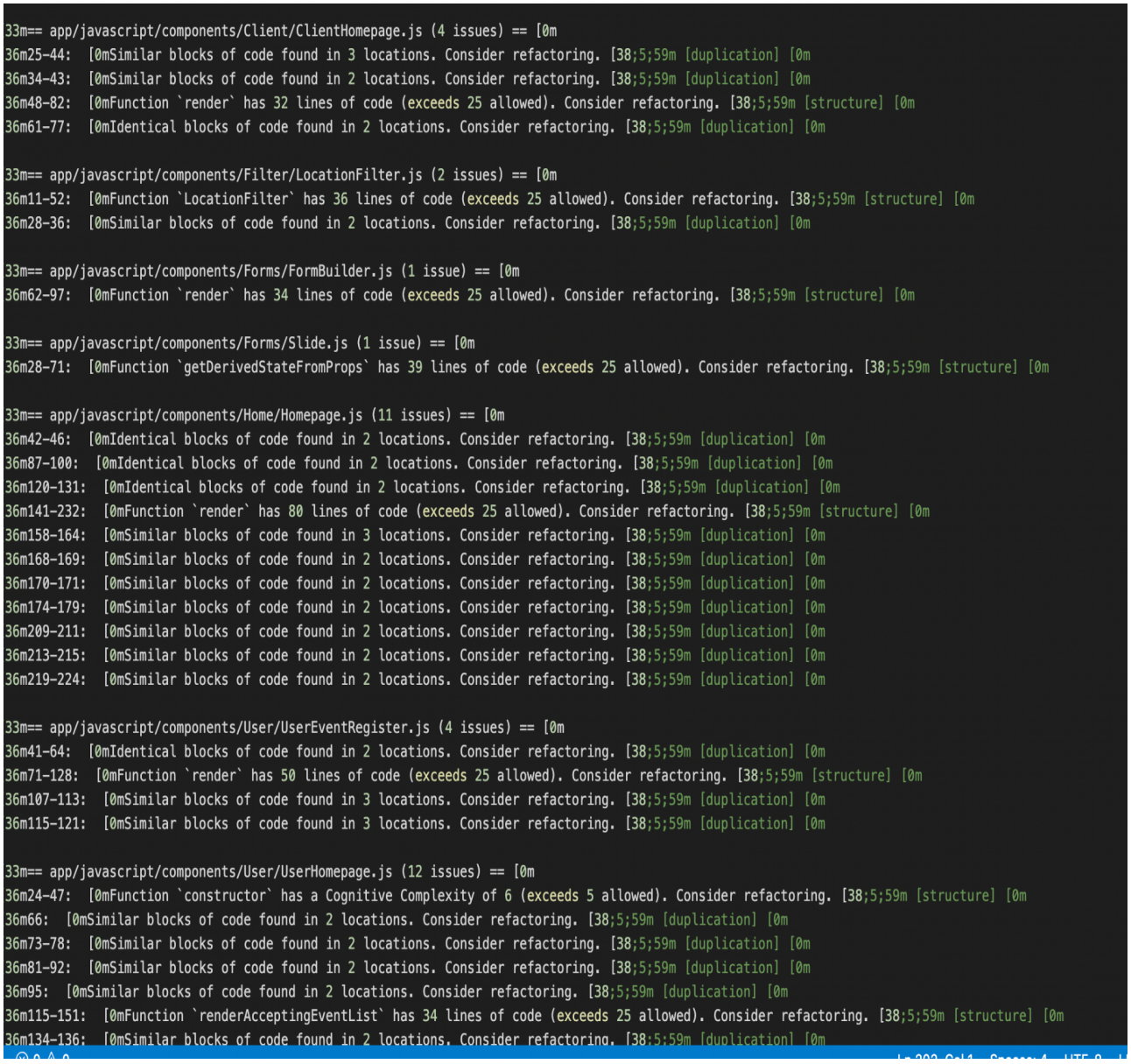
● Storage: This is not enough for both deployments. You will need to update your storage to 25GB before proceeding. (AWS has a Free Storage Limit of 30GB)

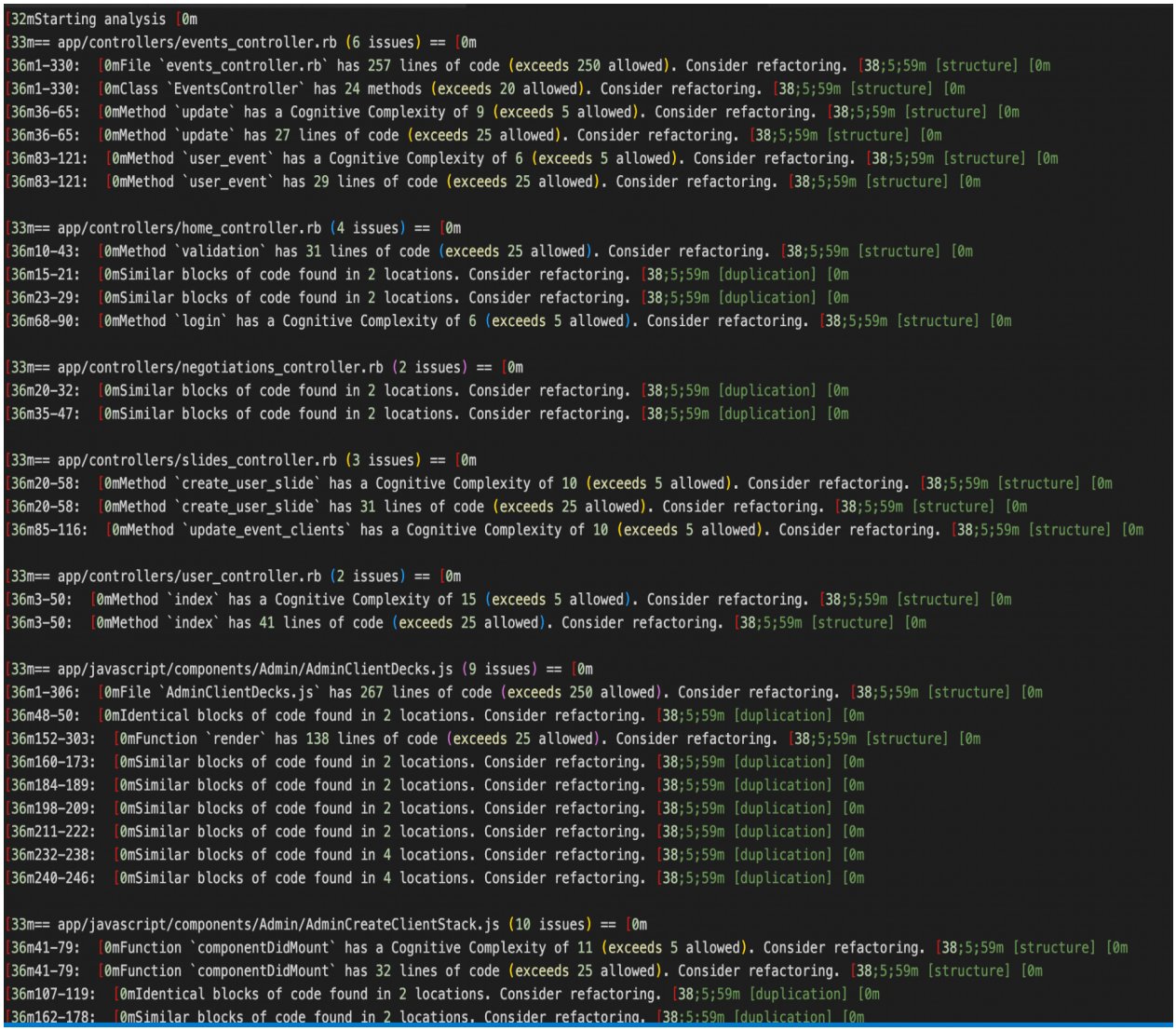
● RAM: This is not enough for the Heroku Container deployment. You will run into **insufficient heap storage** when containerizing the application. You will need to upgrade to **t3.small** (Create this instance only for deployment. It is **NOT FREE**). 1GB RAM from the Free Tier will be sufficient for the Local AWS deployment.

Gems/Tools Used

- **Code Climate:** We used CodeClimate to keep track of code quality issues in our code base. Codeclimate helped us in finding similar pieces of code across multiple Rails controllers or React components. This helped us extract similar code into a single function reducing the total number of lines and increasing modularity. We have refactored the code as per the code climate report.

Attached below feedback from codeclimate tool. Most of these are addressed by our team during Refactoring and Quality improvement.


33m== app/javascript/components/Client/ClientHomepage.js (4 issues) == [0m
36m25-44: [0mSimilar blocks of code found in 3 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m34-43: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m48-82: [0mFunction `render` has 32 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
36m61-77: [0mIdentical blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
33m== app/javascript/components/Filter/LocationFilter.js (2 issues) == [0m
36m11-52: [0mFunction `LocationFilter` has 36 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
36m28-36: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
33m== app/javascript/components/Forms/FormBuilder.js (1 issue) == [0m
36m62-97: [0mFunction `render` has 34 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
33m== app/javascript/components/Forms/Slide.js (1 issue) == [0m
36m28-71: [0mFunction `getDerivedStateFromProps` has 39 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
33m== app/javascript/components/Home/Homepage.js (11 issues) == [0m
36m42-46: [0mIdentical blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m87-100: [0mIdentical blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m120-131: [0mIdentical blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m141-232: [0mFunction `render` has 80 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
36m158-164: [0mSimilar blocks of code found in 3 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m168-169: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m170-171: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m174-179: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m209-211: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m213-215: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m219-224: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
33m== app/javascript/components/User/UserEventRegister.js (4 issues) == [0m
36m41-64: [0mIdentical blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m71-128: [0mFunction `render` has 50 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
36m107-113: [0mSimilar blocks of code found in 3 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m115-121: [0mSimilar blocks of code found in 3 locations. Consider refactoring. [38;5;59m [duplication] [0m
33m== app/javascript/components/User/UserHomepage.js (12 issues) == [0m
36m24-47: [0mFunction `constructor` has a Cognitive Complexity of 6 (exceeds 5 allowed). Consider refactoring. [38;5;59m [structure] [0m
36m66: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m73-78: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m81-92: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m95: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
36m115-151: [0mFunction `renderAcceptingEventList` has 34 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
36m134-136: [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m



```
[32mStarting analysis [0m
[33m== app/controllers/events_controller.rb (6 issues) == [0m
[36m1-330:  [0mFile `events_controller.rb` has 257 lines of code (exceeds 250 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m1-330:  [0mClass `EventsController` has 24 methods (exceeds 20 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m36-65:  [0mMethod `update` has a Cognitive Complexity of 9 (exceeds 5 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m36-65:  [0mMethod `update` has 27 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m83-121:  [0mMethod `user_event` has a Cognitive Complexity of 6 (exceeds 5 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m83-121:  [0mMethod `user_event` has 29 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m

[33m== app/controllers/home_controller.rb (4 issues) == [0m
[36m10-43:  [0mMethod `validation` has 31 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m15-21:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m23-29:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m68-90:  [0mMethod `login` has a Cognitive Complexity of 6 (exceeds 5 allowed). Consider refactoring. [38;5;59m [structure] [0m

[33m== app/controllers/negotiations_controller.rb (2 issues) == [0m
[36m20-32:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m35-47:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m

[33m== app/controllers/slides_controller.rb (3 issues) == [0m
[36m20-58:  [0mMethod `create_user_slide` has a Cognitive Complexity of 10 (exceeds 5 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m20-58:  [0mMethod `create_user_slide` has 31 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m85-116:  [0mMethod `update_event_clients` has a Cognitive Complexity of 10 (exceeds 5 allowed). Consider refactoring. [38;5;59m [structure] [0m

[33m== app/controllers/user_controller.rb (2 issues) == [0m
[36m3-50:  [0mMethod `index` has a Cognitive Complexity of 15 (exceeds 5 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m3-50:  [0mMethod `index` has 41 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m

[33m== app/javascript/components/Admin/AdminClientDecks.js (9 issues) == [0m
[36m1-306:  [0mFile `AdminClientDecks.js` has 267 lines of code (exceeds 250 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m48-50:  [0mIdentical blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m152-303:  [0mFunction `render` has 138 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m160-173:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m184-189:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m198-209:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m211-222:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m232-238:  [0mSimilar blocks of code found in 4 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m240-246:  [0mSimilar blocks of code found in 4 locations. Consider refactoring. [38;5;59m [duplication] [0m

[33m== app/javascript/components/Admin/AdminCreateClientStack.js (10 issues) == [0m
[36m41-79:  [0mFunction `componentDidMount` has a Cognitive Complexity of 11 (exceeds 5 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m41-79:  [0mFunction `componentDidMount` has 32 lines of code (exceeds 25 allowed). Consider refactoring. [38;5;59m [structure] [0m
[36m107-119:  [0mIdentical blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
[36m162-178:  [0mSimilar blocks of code found in 2 locations. Consider refactoring. [38;5;59m [duplication] [0m
```

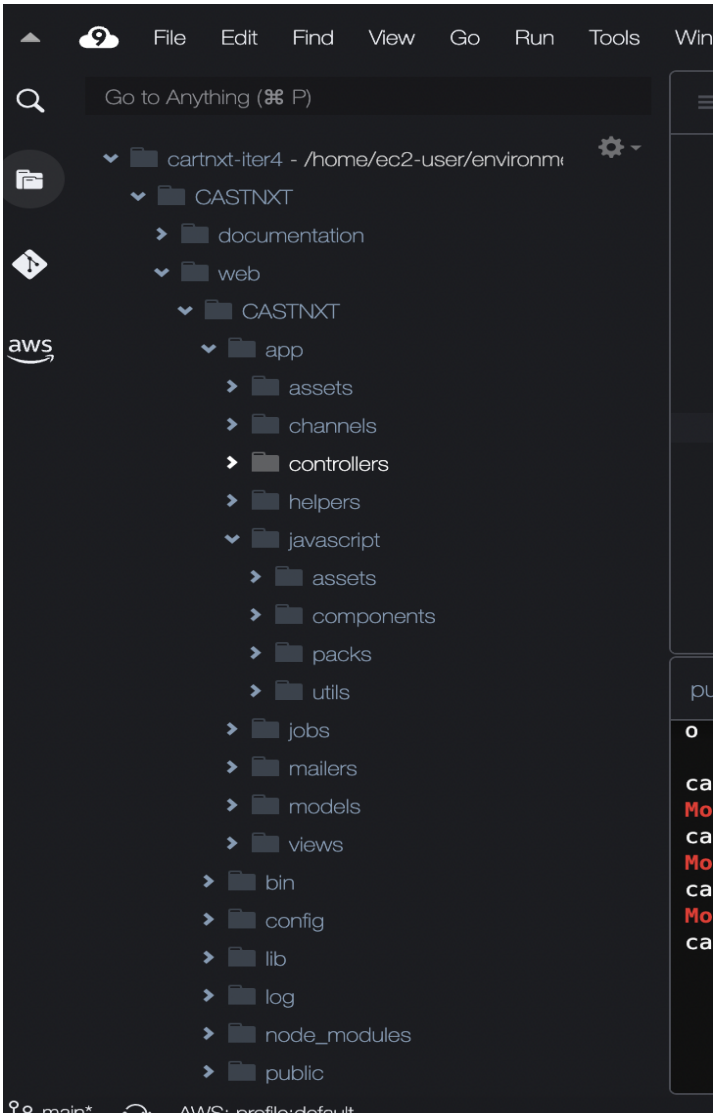
- **JEST framework:** Quality improvements: There were no javascript test cases in the legacy project. We have introduced the jest framework and improved the code coverage to 80%. The reports are included below in this document.
- **React material UI:** We used Material UI's React component library in the front end of our application. This was of much use to us since all components in this library were by default responsive to various screen sizes. This library also ensured that our design language stayed consistent throughout the application.

Final Github repo status :

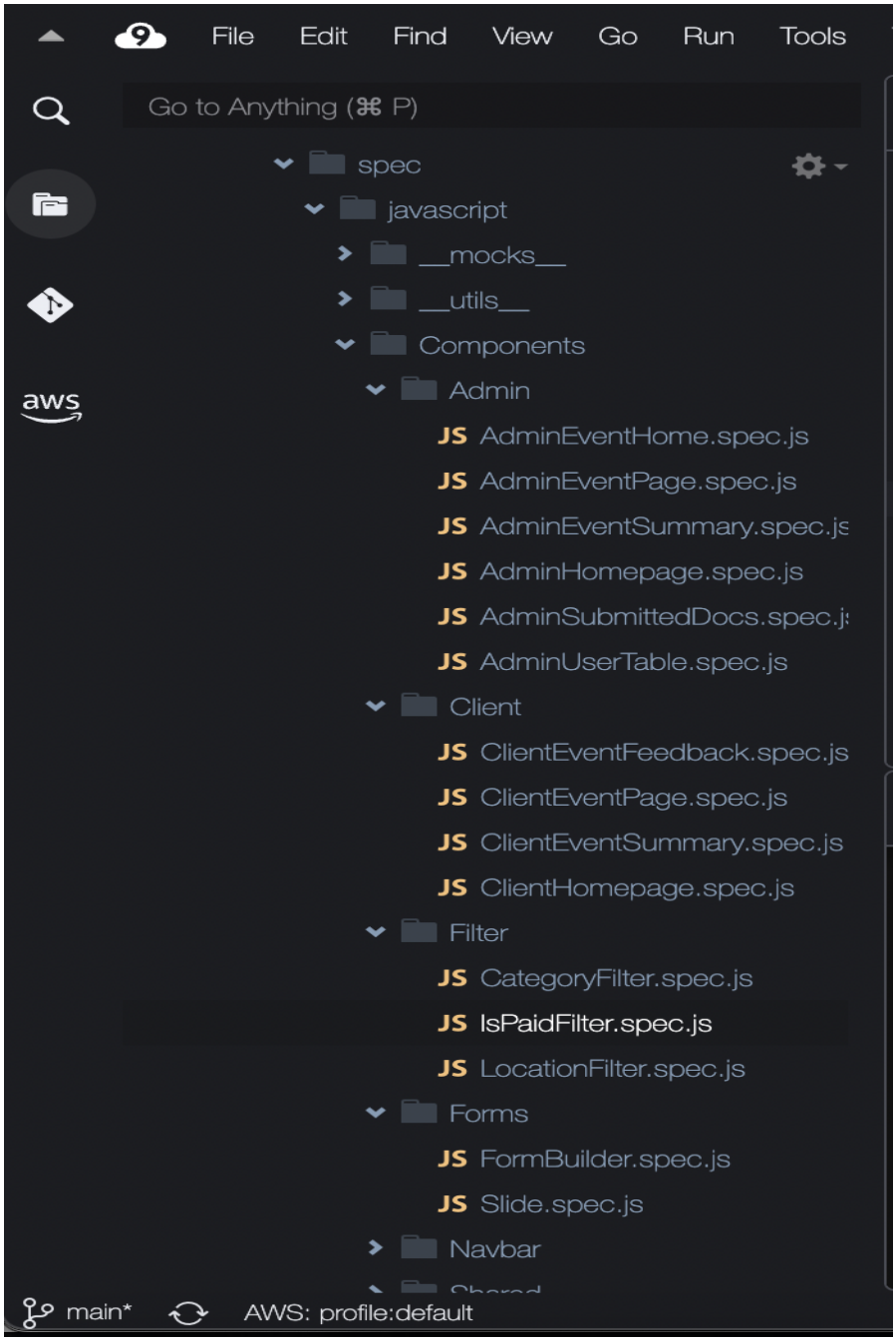
We have pushed all the code changes including Rspec files and other changes to our GITHUB repository.

Repo contents and deployment scripts:

1. The repo uses ReactJS framework for UI and Ruby on Rails framework for backend.



2. Jest framework is introduced to add javascript spec files. Ruby tests are also written to cover the backend part.



3. Heroku deployment is being done manually using heroku CLI. The commands are given in the Deployment guide section.
4. Mongdb.yml files have all the database configuration for production, development and test environments.

Pending Items and Future Outlook:

1. Color coding on slide tab: The producer or the client should be able to color code the talents in the summary tables for their future reference about types or categories of clients.
2. Ability to update password using the 'Forgot/Reset Password' feature.
3. Remove the selection of user/talent/producer radio buttons when signing up. A regular user who signs up on the web app should only be taken to the talent dashboard. An admin (event manager) can create other admin and client profiles. Client profiles with default passwords can be sent to clients via email for them to reset and use - they will be redirected to the client dashboard. Admin adding other admins will create a profile taking them to the admin dashboard.
4. Bug-fix: Removing a talent from the initially curated admin deck (first level of filtering submissions) should remove the talent from the assigned client decks which are being negotiated on UI.
5. Provide support for previewing PDF uploads. Currently, the application supports previewing only image uploads
6. Mobile View: Optimize the application for viewing on mobile devices.
7. Edit Event – Give the producer the ability to edit a previously created event.
8. Automatic Billing dashboard - When the client wants to pay the talents, there should a billing dashboard generated where the client can edit the payment amounts and links and click on Pay Now to send the money to the talent.

Important Links:

Pivotal tracker - https://www.pivotaltracker.com/n/projects/2599656

GitHub - https://github.com/tamu-edu-students/CASTNXT

Heroku - https://castnxt-final.herokuapp.com/

Presentation and Video Demos:

Presentation and Demo Video - https://youtu.be/PVcpLw3_sP4

Presentation link-
https://tamucs-my.sharepoint.com/:p:/g/personal/anushkagarg_tamu_edu/Efy2j5Wx94RGg3DHXSwwhzABcBfsTAJGz2VbhXSGtP7-5Q?e=glfvnl