FashioNXT: Event360

PlaNXT | CastNXT | EventNXT

Final Report

Table of Contents

1. Summary

Events360 serves as the central platform for three specialized apps developed by the event planning company FashioNXT: PlaNXT, CastNXT, and EventNXT. Each app is tailored to a specific aspect of event management and operates independently with its own database. The key feature linking these apps is the Event360 CRM system, which allows users to access all three apps through a single login. Additionally, Event360 offers administrative tools that enable monitoring and management of the system's backend, providing a streamlined experience for both users and administrators.

- *PlaNXT*

PlaNXT is an innovative online application that is a part of FashioNXT and intends to streamline the setup and teardown processes for events. It allows users to create and manage multiple floor plans with varied dimensions, offering the flexibility to place and time items like chairs according to specific event needs. Inheriting a functioning feature-rich application from the previous team, our team was entrusted with the responsibility to implement a successful sign-out feature that would enable the user to sign out from all the apps including Events360. This semester, our primary goals as directed by our client, were to set up the environment and the heroku application without any conflicts, along with ensuring that the downloadable data is readable, uploading an existing floor plan, and to implement a save button to download this plan to a local device.

By the end of the semester, the team was able to deliver the features as requested by the client. Additionally, the team focused on enhancing the user interface, functionality, testing, and overall usability of the previous team's application. We're confident that we have left the app in a strong state, well-prepared for future teams to further develop and refine.

- *CastNXT*

CastNXT is an application designed for producer, client and talent to do negotiation for events. A producer could create an event as well as its form. The form is similar to a google form for talent to submit their information. And the producer will receive those submissions and decide whether to move on or not. If the producer wants to move on with the talent, he can assign

those talents to the clients and chat with the client in the comment section. The producer can also add talent for an event from his side for convenience.

- *EventNXT*

EventNXT is an event and RSVP management app tailored for event organizers, which allows them to create an event, upload box office data, manage seating categories, invite guests, and send out referral emails. The app facilitates tracking purchase information from box office spreadsheets and offers direct connections to ticketing websites. Users can send referral emails, with rewards managed in the app, and QR codes for guest identification. As part of the project handoff, we received two applications from the previous semester (Fall 2023) EventNXT Team: *EventNXT_old_app* and *EventNXT_new_app*. EventNXT_old_app is operational with some missing functionalities such as seamless integration with Event360 (CRM), Referral rewards feature, Importing guests from 3rd party ticketing websites such as Ticket Master or EventBrite etc.This meant that we had to work with the Event_new_app, which had many missing features compared to the old application i.e., RSVP email functionality, seating summary feature, importing guests from box office data, referral rewards management, and box office sales features.

By the end of the semester, the team was able to deliver all the features requested by the client, which included a successful sing-out feature. Additionally, the team focused on enhancing the user interface, functionality, testing, and overall usability of the previous team's application. We're confident that we have left the app in a strong state, well-prepared for future teams to further develop and refine.

2. Description of all user stories

Sprint 1:

1. User Story: Environment/Heroku Setup for PlaNXT

a. Points: 3
b. Status: Completed
c. Summary: Accomplished setup and deployment of the legacy PlaNXT app locally and on Heroku

2. User Story: Environment/Heroku Setup for EventNXT

a. Points: 3
b. Status: Completed
c. Summary: Accomplished setup and deployment of the legacy EventNXT app locally and on Heroku

3. User Story: Environment/Heroku Setup for CastNXT:

a. Points: 3
b. Status: Completed
c. Summary: Accomplished setup and deployment of the legacy CastNXT app locally and on Heroku

Sprint 2:

1. User Story: Successful Sign Out Feature:

a. Points: 4
b. Status: Completed
c. Summary: The signout feature that logs out the user from the PlaNXT app and all other apps under Event360 was successfully deployed.

2. User Story: Tests for the signout feature:

a. Points: 2
b. Status: Completed
c. Summary: Tests for the signout feature were written and passed successfully

3. User Story: Readable downloadable data feature:

a. Points: 1
b. Status: Completed
c. Summary: The readable downloadable data feature was successfully implemented.

4. User Story: Tests for the readable downloaded data:

a. Points: 1
b. Status: Completed
c. Summary: Tests for the readable downloaded data feature were written and passed successfully

5. User Story: Save Floorplan data feature:

a. Points: 2
b. Status: Completed
c. Summary: The feature that allows the PlaNXT user to easily analyze downloaded data was successfully deployed.

6. User Story: Tests for save Floorplan data feature:

a. Points: 2
b. Status: Completed
c. Summary: Tests for “Save Floorplan data” feature were written and passed successfully

7. User Story: Write test case for upload csv for floor plan

a. Points: 3
b. Status: Incomplete
c. Summary: As a PlanXT developer, So that I can ensure the platform's capability to integrate diverse floor plan formats, I want to test the "Upload Floorplan " feature for its ability to successfully upload and recognize CSV files, confirming the feature's functionality and reliability for user needs.

8. User Story: Upload existing floor plan

a. Points: 5
b. Status: Incomplete
c. As a user of the PlaNXT platform, So that I can efficiently reuse and modify existing floor plans for my events, I want the "Upload Existing Floor Plan " button to function correctly, allowing me to import and edit floor plans without encountering errors or usability issues.

Sprint 3:

1. **User Story:** Successful Sign Out Feature:
   a. Points: 2
   b. Status: Completed
   c. Summary: The signout feature that logs out the user from the EventNXT app and all other apps under Event360 was successfully deployed.

2. User Story: Tests for Sign Out Feature

a. Points: 1
b. Status: Completed
c. Summary: Rspec tests were written and passes

3. User story: Upload Manage Guests

a. Points: 5
b. Status: Completed
c. Summary: The feature was implemented and works as intended. Took quite a bit of time to get the parsing worked.

4. User story: Upload Manage Guests Tests

a. Points:3
b. Status: Completed
c. Summary: The feature was implemented and works as intended. Took quite a bit of time to get the parsing worked.

5. User Story: Fix destroy guest button when upload manage guests

a. Points: 1
b. Status: Completed
c. Summary: Fixed the issues when removing guests that it would break the code.

6. User Story: Work on some UI for upload manage guests

a. Points: 1
b. Status: Completed
c. Summary: Originally the upload button was two separate buttons and now it is one button and bypassed a page it would redirect you to when deleting a guest.

7. User story: Sending out RSVPs

a. Points: 3
b. Status: Completed
c. Summary: The feature was implemented and the intended guests received the RSVP emails.

8. User story: Tests for RSVPs

a. Points: 1
b. Status: Completed
c. Summary: Tests were written and successfully passed.

9. User Story: Update "Committed Seats" from RSVP

a. Points: 3
b. Status: Completed
c. Summary: The feature was implemented and the guest was able to commit their desired number of seats through the RSVP email.

10. User Story: UI for Box Office Sales

a. Points: 2
b. Status: Completed
c. Summary: User Interface work was done to the Box Office Sales table.

11. **User Story:** Update Committed Guests in "Manage Invited Guests"
    a. Points: 3
    b. Status: Completed
    c. Summary: The feature was implemented successfully and the number of seats that were committed by the guests were updated in the “Manage Invited Guests” table

12. **User Story:** Fix last semesters code coverage
    a. Points: 5
    b. Status: Completed
    c. Summary: Improved code coverage from last semester. Started with 62 percent.

13. **User Story:** Parsed Box Office Data
    a. Points: 5
    b. Status: Completed
    c. Summary: The feature was implemented and the parsing was done as per requirement.

14. **User Story:** Referral email to guest
    a. Points: 3
    b. Status: Completed
    c. Summary: Completed the referral email template, so the guest can receive the referral

15. **User Story:** Update Seating Summary
    a. Points: 3
    b. Status: Completed
    c. Summary: The feature was successfully implemented and the seating summary was updated as per the number of tickets committed by the guests and remaining seats were calculated and displayed.

16. **User Story:** Tracking the corresponding referral information form the referred person
    a. Points: 3
    b. Status: Completed
    c. Summary: No debugging reports so far

17. **User Story:** Reward calculation when we have the corresponding referral tuples
    a. Points: 3
    b. Status: Completed
    c. Summary: See further debugging report if we need to do the debugging

18. **User Story:** Rspec examplification, Cucumber deexamplification on 14
    a. Points: 3
    b. Status: Completed
    c. Summary: No debugging reports so far

19. **User Story:** Rspec examplification, Cucumber deexamplification on tracking the corresponding referral information form the referred person
    a. Points: 3
    b. Status: Completed
    c. Summary: No debugging reports so far

20. **User Story:** Parsing the corresponding information of tickets buyer (quantity, payment) directly from the uploaded .xlsx file into the corresponding referral table
    a. Points: 3
    b. Status: Completed
    c. Summary: No debugging reports so far

21. **User Story:** Referral email to guest's friends
    a. Points: 3
    b. Status: Completed
    c. Summary: Guests' friends can receive the referral email and buy tickets online

22. **User Story:** Fix Heroku deployment
    a. Points: 3
    b. Status: Completed
    c. Summary: Last semesters heroku deployment was taken down and needed to figure out why heroku did not like us uploading EventNXT code

23. User Story: Test for sending email to guests

a. Points: 3
b. Status: Completed
c. Summary: Completed the test for sending email to guests

24. User Story: Test for sending email to guest's friends

a. Points: 3
b. Status: Completed
c. Summary: Completed the test for sending email to guest's friends

Sprint 4:

1. **User Story**: On Click Coloring Effect of Client Event Summary
   a. Points: 3 points
   b. Status: Implemented using switching the color method
   c. Summary: This feature implements certain marking by color mechanism, which will provide certain convenience for the users to distinguish between the different column or row cells, especially when facing a large and long file.

2. User Story: Coloring Effect of Client Event Summary Jest Testing

a. Points: 3
b. Status: Completed
c. Summary: This story implements the JavaScript React Jest testing for the coloring effect of the cells in the client event summary which is by checking various switching testing cases.

3. User Story: Sorting Mechanism of Client Event Summary

a. Points: 3
b. Status: Completed
c. Summary: This story implements the sorting mechanism of the client event summary which allows the user to sort the corresponding columns in the summary table.

4. **User Story**: Sorting Mechanism of Client Event Summary testing
    a. Points: 3
    b. Status: Completed
    c. Summary: This story implements the testing of sorting mechanism of the client event summary which allows the user to sort the corresponding columns in the summary table.

5. **User Story:** Update Email feature
    a. Points: 2
    b. Status: Completed
    c. Summary: The email feature was implemented successfully.

6. **User Story:** Removed 'Guests Committed' from Events page
    a. Points: 3
    b. Status: Completed
    c. Summary: The client wanted to remove the ‘guests committed’ column from the events page and was implemented successfully.

7. **User Story:** Remove 'Guests Committed' from New Guests Page
    a. Points: 3
    b. Status: Completed
    c. Summary: The client wanted to remove the ‘guests committed’ column from the new guests page and was implemented successfully.

8. **User Story:** Add a 'Section' column to Seats
    a. Points: 5
    b. Status: Completed
    c. Summary: The client wanted to add a ‘Sections’ column to the new seats page and was implemented successfully.

9. **User Story:** Add a 'Section' column to Guests
    a. Points: 5
    b. Status: Completed
    c. Summary: The client wanted to add a ‘Sections’ column to the new guests page and was implemented successfully.

10. **User Story:** Debugged App Deployment
    a. Points: 5
    b. Status: Completed
    c. Summary: The app deployment issue in EventNXT was fixed and the app was deployed successfully.

11. **User Story:** React Tests for 'Component/User' - Coverage 75%
    a. Points: 2
    b. Status: Completed
    c. Summary: Finish React tests on ‘Component/User’ that last semester's team did not finish.

12. **User Story:** Write Header React Tests
    a. Points: 2
    b. Status: Completed
    c. Summary: Finish React tests that last semester team did not finish

13. **User Story:** Write location Filter React Tests
    a. Points: 2
    b. Status: Completed
    c. Summary: Finish React tests that last semester team did not finish

14. **User Story:** Write Is paid Tests
    a. Points: 2
    b. Status: Completed
    c. Summary: Finish React tests that last semester team did not finish

15. **User Story:** Write Category Filter React Tests
    a. Points: 2
    b. Status: Completed
    c. Summary: Finish React tests that last semester team did not finish

16. **User Story:** Sign Out Feature for CastNXT
    a. Points: 5
    b. Status: Completed
    c. Summary: Implement logout feature that signs you out of all the apps

17. **User Story:** Write a Cross-Origin Resource Sharing Method
    a. Points: 2
    b. Status: Completed
    c. Summary: For CastNXT, we needed to write a cross-origin resource sharing method to allow CastNXT to call the events360 logout page to sign you out.

18. **User Story:** Update Header Tests After Sign Out Feature
    a. Points: 2
    b. Status: Completed
    c. Summary: After implementing the sign out feature had to go back and adjust the header react tests.

19. **User Story:** Implement Email Notification when Update Decks is Pressed
    a. Points: 4
    b. Status: Completed
    c. Summary: Implemented the code to trigger a method that does the email notification but did not have time to debug it.

20. **User Story:** Add “image” attribute to form
    a. Points: 4
    b. Status: Completed
    c. Summary: Modify the data structure(model) of the form, that is, add "image" attribute to the model

21. **User Story:** Add image upload option
    a. Points: 3
    b. Status: Completed
    c. Summary: Make the "AdminCreateEvent" page's "Form Preview" has the image upload option (pure front-end, because it's just a review)

22. **User Story:** Add image upload support
    a. Points: 4
    b. Status: Completed
    c. Summary: Make the "UserEventRegistration" page support the image upload feature

23. **User Story:** Make images visible
    a. Points: 4
    b. Status: Completed
    c. Summary: Make the image visible for client's selecting page and Admin's view page (3 pages in totally)

24. **User Story**: React testing for the AdminEventSummary.js components paymelink testing
    a. Points: 1
    b. Status: Completed
    c. Summary: The story implements the React testing for the AdminEventSummary.js components paymelink testing.

25. **User Story**: React testing for the AdminUserTable.js components paymelink testing
    a. Points: 1
    b. Status: Completed
    c. Summary: The story implements the React testing for the AdminUserTable.js components paymelink testing.

26. **User Story**: React testing for the AdminEventSummary.js components csv file generating testing
    a. Points: 1
    b. Status: Completed
    c. Summary: The story implements the React testing for the AdminEventSummary.js components csv file generating testing.

27. **User Story**: React testing for the AdminUserTable.js components csv file generating testing
    a. Points: 1
    b. Status: Completed
    c. Summary: The story implements the React testing for the AdminUserTable.js components csv file generating testing.

28. **User Story:** ‘Manage Seating Levels’ was updated to display the Sections Column
    a. Points: 5
    b. Status: Completed
    c. Summary: ‘Manage Seating Levels’ were updated to display the Sections column successfully.

29. **User Story:** Seating Summary was updated to display the Sections Column
    a. Points: 5
    b. Status: Completed
    c. Summary: ‘Seating Summary’ was updated to display the Sections Column successfully.

30. **User Story:** Create a new section in “Add New Seats”
    a. Points: 5
    b. Status: Completed
    c. Summary: The user can now successfully add sections when creating a new seat.

31. **User Story:** Display the new seating summary based on categories and section
    a. Points: 5
    b. Status: Completed
    c. Summary: New seating summary based on categories and section was implemented successfully.

32. **User Story:** react test for components/admin, **including 24, 25, 26, 27**
    a. Points: 8
    b. Status: Completed
    c. Summary: react test for components/admin above 90%

33. User Story: Debugged App Deployment in CastNXT

a. Points: 5
b. Status: Completed
c. Summary: The app deployment issue in CastNXT was fixed and the app was deployed successfully.

Sprint 4 - Backlog:

1. **User Story:** Debug email notification when button update decks is pressed
   a. Points: 4
   b. Status: Incomplete
   c. Summary: Need to debug why the email notification would not send when the button update decks is pressed. Ran out of time.

2. User Story: Write Signout Cucumber tests

a. Points: 2
b. Status: Incomplete
c. Summary: Need to write cucumber tests for the sign out feature.

3. User Story: Write Signout RSPEC tests

a. Points: 2
b. Status: Incomplete
c. Summary: Need to write rspec tests for the sign out feature.

4. User Story: Write Email Notification when button update decks is pressed Cucumber tests

a. Points: 2
b. Status: Incomplete
c. Summary: Need to write cucumber tests for the email notification feature.

5. User Story: Write Email Notification when button update decks is pressed RSPEC tests

a. Points: 2
b. Status: Incomplete
c. Summary: Need to write rspec tests for the email notification feature.

3. Legacy project: understanding the existing code and refactoring

Working on the legacy software development project for the FashioNXT web application presents both challenges and opportunities. This project builds on work from previous semesters, which, while largely unchanged in scope, offers specific areas where enhancements and modifications can be made to the existing framework. This includes updates across three closely related subprojects.

The initial task in tackling this legacy code is not trivial; it requires a deep understanding of what has already been implemented. We need to first comprehend how previous teams guaranteed functionality within the application, focusing on the compatibility of these functions with the core framework. This understanding is critical as it forms the basis of our Sprint 1 Plan, where we set new expectations and identify improvements.

To effectively enhance and possibly introduce new functionalities, it's essential to meticulously document user stories and develop corresponding storyboards and sketches. This ensures that any new developments align seamlessly with the established framework. For example, when considering improvements to the event arrangement feature, we must concentrate on how new methods will integrate with and impact the existing framework.

A specific area for enhancement is the login function, which requires a comprehensive overhaul of the previous implementation. We need to fully understand the existing login mechanisms to effectively redesign or enhance this feature. This involves not just reorganizing the authentication process but ensuring it is integrated and coherent.

Furthermore, engaging with customers and previous developers is crucial. This engagement will enhance our understanding of the existing codebase, including how various methods and classes are interconnected and how previous functionalities were implemented based on user stories. Such insights will pinpoint the best starting points for development and guide our overall strategy in managing legacy code in this complex environment.

By adopting this methodical approach, we can ensure that our enhancements are thoughtful, well-integrated, and contribute positively to the FashioNXT project, despite the complexities of working with legacy systems.

4. Team Roles

Sprint 1:

- Product Manager: Alex Wise
- Scrum Master: Louis Turrubiartes

Sprint 2:

- Product Manager: Louis Turrubiartes
- Scrum Master: Alex Wise

Sprint 3:

- Product Manager: Amalesh Arivanan
- Scrum Master: Tianchen Huang

Sprint 4:

- Product Manager: Tianchen Huang
- Scrum Master: Tong Wu

5. Summary of accomplishments and points completed per sprint

In ***Sprint 1***, the team successfully completed the setup and deployment of three legacy applications—PlaNXT, EventNXT, and CastNXT—both locally and on Heroku. Each task was allocated three points and all were marked as completed. This sprint focused on ensuring that the foundational environments for these applications were properly established on Heroku, facilitating further development and updates in future sprints.

Sprint 2 saw substantial progress with the successful deployment and testing of several key features for the PlaNXT app and its integration with Event360. The sprint completed the deployment of a sign-out feature that logs out users from PlaNXT and other related apps, along with its corresponding tests that were executed successfully. Additionally, a new feature for generating readable, downloadable data was implemented and tested, enhancing user experience with easily accessible data outputs. Similarly, a feature for saving floorplan data was also rolled out and tested, ensuring users can analyze downloaded data efficiently.

However, the sprint also faced some challenges with incomplete tasks. Work on the test case for the upload of CSV floor plans remains unfinished. This test is crucial for ensuring the platform can handle various floor plan formats reliably. Another incomplete task is the development of the functionality for uploading existing floor plans, which is essential for users looking to reuse and modify floor plans for events. These tasks are critical for enhancing the platform's robustness and user-friendliness, indicating areas of focus for the next sprint.

Sprint 3 was highly productive, with the team successfully completing a series of complex tasks aimed at enhancing the user experience and functionality of the EventNXT app and other integrated applications under Event360. Noteworthy accomplishments included deploying a sign-out feature for EventNXT, implementing and refining the Upload Manage Guests feature—along with associated UI improvements—and ensuring robust testing across all new functionalities. Additionally, significant enhancements were made to guest management, including the ability to send RSVPs, update seating commitments, and manage box office sales. Efforts were also made to improve system stability and performance, with updates to the system's code coverage and troubleshooting deployment issues on Heroku. The sprint also focused on refining the referral system, ensuring guests and their friends could receive and respond to referral emails effectively, thereby promoting ticket sales. This comprehensive set of updates and tests underlines the team's commitment to delivering a reliable and user-friendly event management platform.

Sprint 4 was marked by extensive feature implementations and robust testing across several components of the EventNXT and CastNXT platforms. The team successfully integrated a new coloring effect for client event summaries, enhancing user interaction by distinguishing between different rows and columns visually, which was also rigorously tested using Jest. Additionally, a sorting mechanism was added to the client event summaries, allowing users to organize data more efficiently, with corresponding tests confirming its functionality. Significant updates were made to the user interface, including the addition and removal of specific columns on various pages, which streamlined the display and management of guest and event data.

Email functionality was improved, and a sign-out feature that logs users out across all apps was implemented and tested. The team also addressed system deployment issues, ensuring smooth operations. Image handling capabilities saw significant enhancements, with new features supporting image uploads and visibility across different administrative and user interfaces.

React tests were extensively written to cover new and existing functionalities, ensuring a high level of code reliability and performance.

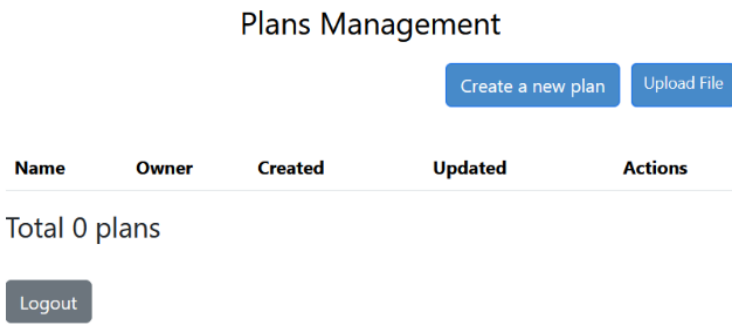
However, the sprint ended with some tasks in the backlog, including debugging and testing the email notification system and the sign-out process, which were not completed due to time constraints. These remaining tasks highlight areas requiring immediate attention in the upcoming sprint to ensure all features operate seamlessly.

6. Design Diagrams - Snapshots

The features were implemented as designed during the Sprint Plan.

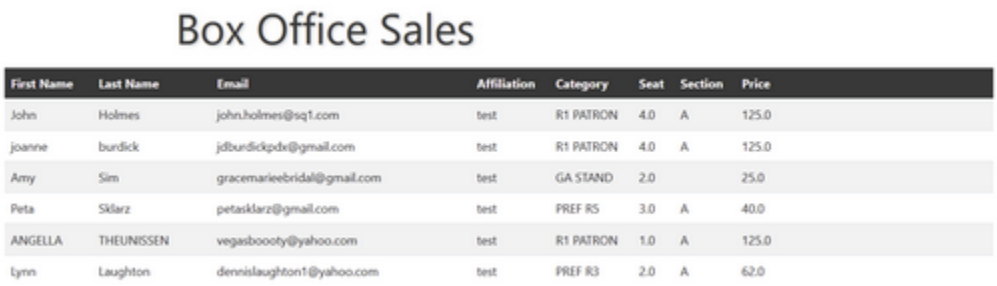
Sprint 2:

- Upload Floor Plan

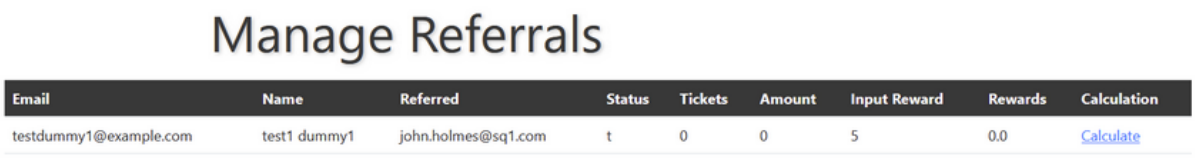


Sprint 3:

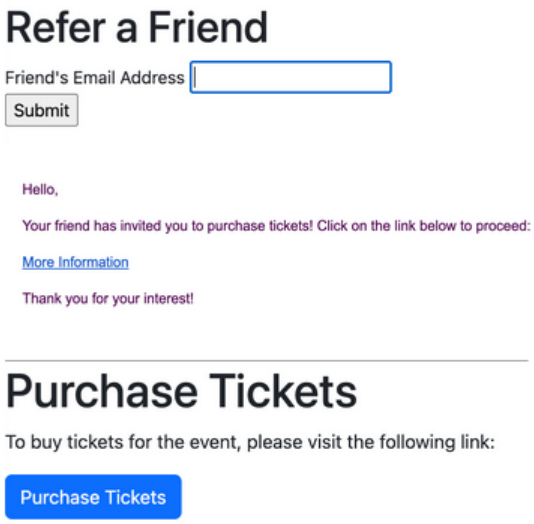
- *Box Office Sales*



- *Manage Referrals*



- *Referral Email*



Sprint 4

- *Column Coloring*

Preference	Name	Gender	Email	DOB
1	ABC	M	abc@example.com	04-05-2024
2	XYZ	M	xyz@example.com	04-05-2024

7. Table of user stories points completed by each member

Team Member	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Total points completed by the member
Alex Wise	9	10	31	33	83
Amalesh Arivanan	9	6	31	45	91
Louis Turrubiartes	9	2	21	23	55
Tianchen Huang	9	11	29	31	80
Tong Wu	9	0	15	16	40
Xin Tong	9	0	15	20	44
Total Points	9	29	142	168	393

8. List of customer meeting dates, and description of each meeting

Client meetings were held every Tuesday at 7:00 PM CST

Sprint 1:

- *Date:* Every Tuesday
- *Time:* 7:00 PM CST
- *Mode:* Recurring Zoom Call
- *Summary:* The feedback the team received after dedicating the entire sprint to project initialization and diving into documentation, the expectation is to transition into implementing both new and existing features seamlessly across the three projects. To handle this, the approach was shifted from operating as three distinct teams working on three different projects simultaneously to focusing on a separate project every sprint as one cohesive unit.

Sprint 2:

- *Date:* Every Tuesday
- *Time:* 7:00 PM CST
- *Mode:* Recurring Zoom Call
- *Summary:* The team was able to achieve the objectives of the sprint as planned. Each member of the team was able to effectively demonstrate their contribution to PlaNXT and the feedback from the client was noted after the weekly demo. Backlogs of the sprint were noted and were planned to execute on a later date.

Sprint 3:

- *Date:* Every Tuesday
- *Time:* 7:00 PM CST
- *Mode:* Recurring Zoom Call
- *Summary:* The team was able to achieve the objectives of the sprint as planned. Each member of the team was able to effectively demonstrate their contribution to EventNXT and the feedback from the client was noted after the weekly demo. The updated features that were requested by the client were noted and planned to be implemented by the next sprint.

Sprint 4:

- *Date:* Every Tuesday
- *Time:* 7:00 PM CST
- *Mode:* Recurring Zoom Call
- *Summary:* The team was able to achieve the objectives of the sprint as planned. Each member of the team was able to effectively demonstrate their contribution to CastNXT and the feedback from the client was noted after the weekly demo. The react tests for the CastNXT app were prioritized by the team as requested by the professor and were successfully implemented.

9. Explain the teams BDD/TDD process and any benefits/problems

TDD is a fundamental practice where we write tests before the actual code. This approach ensures that our codebase only includes necessary features and adheres strictly to the requirements, reducing the likelihood of bugs and regressions. One of the benefits of TDD is that it provides immediate feedback during development, allowing for quick corrections. However, a common challenge is that it can slow down the development process if the test coverage becomes overly detailed or if developers spend too much time maintaining test suites.

BDD takes this a step further by involving non-technical stakeholders in the test creation process. We write specifications in a natural, human-readable language that describes the behavior of the application under various scenarios. This not only ensures that all team members—including product owners and business analysts—are on the same page but also clarifies customer expectations right from the start. While BDD greatly improves clarity and communication, it can introduce overhead in maintaining a large suite of behavior scenarios, especially as project requirements evolve.

Integrating both BDD and TDD helps us build software that closely aligns with both the technical standards and the business needs. It promotes a proactive approach to quality assurance and fosters a culture of continuous feedback and iterative improvement. However, balancing the initial setup time and the ongoing maintenance of both tests and documentation remains a key challenge.

10. Configuration Management approach

Our configuration management approach was carefully designed to accommodate the diverse backgrounds of our team members, considering that half of the team did not have a formal education in computer science or computer engineering. This diversity prompted us to implement several exploratory spikes to ensure that everyone was up to speed and comfortable with our tools and practices. These spikes were particularly useful in areas where more specialized knowledge was necessary, such as advanced version control techniques and automated deployment strategies.

We managed our codebase through six distinct branches, aligning with different features or stages of development. This structure allowed us to isolate work effectively, manage dependencies, and stabilize our code before it reached the main branch. Each change in a branch required a merge request, which was then reviewed by at least one other team member. This peer review process was crucial in maintaining code quality and ensuring that all team members were aligned with the changes being made.

To streamline our workflows and enhance collaboration, we frequently released updates, enabling us to gather user feedback early and often, and to continuously improve our product. This approach not only helped in mitigating risks by catching issues early but also fostered a culture of collective responsibility and learning within the team. Managing the configuration with this level of granularity and oversight proved to be highly effective in maintaining a stable and progressive development environment.

11. Issues encountered in the production release process to Heroku

During our production release process to Heroku, we encountered several issues, particularly around containerization and database management. Last semester had Heroku applications to be containerized for deployment, our team had to learn how to Dockerize our application. This involved creating our own Docker containers, which was a challenge due to varying levels of familiarity with Docker among the team members.

One of the significant hurdles we faced was configuring these containers to work seamlessly with Heroku's environment. The process required us to adjust our Dockerfile settings to meet Heroku's specifications, such as setting the correct port and managing environment variables efficiently.

Another major issue was getting our database to populate correctly on Heroku. Initially, our database migrations were not executing as expected, which led to significant troubleshooting to ensure that our application's data schema was properly set up when deployed. We had to refine our approach to database migrations and seed data, ensuring that scripts were robust and could handle the transition from a local development environment to Heroku's production environment.

12. Issues encountered using Github and other tools

Since our team didn't encounter significant issues with GitHub or other tools during our development process, it allowed us to focus more on development and collaboration rather than troubleshooting tool-related problems. This smooth operation with GitHub, which served as our primary platform for version control, helped in maintaining a consistent workflow, allowing us to efficiently manage our code, track changes, and collaborate without disruptions.

13. Description of other tools/Gems used and their benefits

In our project, we utilized a variety of tools and Gems to enhance our development process, ensure code quality, and facilitate effective collaboration. Here's an overview of each, along with their benefits and any challenges we faced:

1. ***GitHub***: As our central version control system, GitHub was instrumental in managing our codebase and collaborating. It provided features like pull requests, issue tracking, and branch management, which were essential for maintaining code quality and facilitating team collaboration. The platform's robust ecosystem and integrations with other tools streamlined our workflows. However, the challenge was ensuring that all team members adhered to our branching and commit conventions to keep the repository organized.

2. ***CodeClimate***: We used CodeClimate for automated code review and quality analysis. It helped us maintain a high standard of code quality by identifying potential refactorings and flagging complex code before it was merged into the main branch. CodeClimate's detailed reports on code smells and coverage were invaluable for continuous improvement. One downside was that sometimes its analysis could be overly stringent, requiring configuration tweaks to align with our specific project needs without overwhelming the team with notifications.

3. ***SimpleCov***: This Gem was crucial for monitoring test coverage across our Ruby applications. SimpleCov provided easy-to-understand reports that helped us identify untested parts of our codebase, ensuring that critical functionalities were covered by tests. The main benefit was its ability to integrate seamlessly into our existing test suite, providing real-time coverage data. However, at times, achieving the coverage threshold set by our team standards could be challenging, particularly for legacy code or when integrating new features.

Each of these tools played a vital role in our project's success, offering both solutions and learning opportunities. While there were occasional challenges in terms of tool configuration and meeting stringent standards, the overall impact on our development process was profoundly positive, driving us towards best practices in code health and team collaboration.

14. Links to public Github repositories

All features were developed in the repository linked here, this allowed us control over the project and the ability to quickly make changes without needing authentication from the sponsor. At the end of this semester, the final versions of the app were pushed to the repository linked here, this is the FashioNXT Github that is controlled by the sponsor.

15. Description of contents of the repositories for project deployment

Events360:

- Admin username: test@example.com
- Admin password: password

This is required for pairing new apps on Event360 as well as managing the current apps, below is an example of an app we paired to the CRM and is fully integrated. We do not know what the purpose of “API URL” is except that it’s required, so we reused another apps url for this field (it appears to have zero impact).

PlaNXT:

- All information for deploying the application is located in the README.md.
- All current files needed to run the application are in the repository
  - That being said, we did use the public github repository linked here to implement the 3D rendering, you do not need this repository to run the PlaNXT application but I’m including it in case you need further information about it

CastNXT:

- All information is in the readme of the git repository, but the mongodb atlas url needs to be modified later.
- The mail server needs to be configured correctly. Place that key in config/, which contains credentials.yml.enc. After that, the mail server will work.

EventNXT:

- All information for deploying the application is located in the README.md.
- All current files needed to run the application are in the repository.

16. Recommendations to the next group

There's a known bug in the Event360 application where, as a best guess, it does not actually clear your login cookies. For example: if you use the PlaNXT app for the first time and try to login/authenticate using Event360, then you are asked for your username and password like normal. However, if you logout of the app and then attempt to login again, you are then instantly logged in without entering your credentials. The only way to "truly" logout and be forced to re-enter your credentials is to login to Event360 and manually logout of it there and *then* attempt to log in to PlaNXT (or use an incognito browser). Fixing this should not be very difficult, but would benefit all affiliated apps greatly.

PlaNXT:

We believe we left the app in a solid state for future development. One significant feature we didn't have time to implement, which could be a key objective for a future semester, involves enhancing the integration between the 2D and 3D views. One option is to embed the 3D rendering tool within the 2D floor plans page, allowing the 3D view to change in sync with the 2D plan as the time lapse bar is adjusted. Alternatively, adding a time lapse bar to the 3D rendering page would enable it to display a sequence of setups and teardowns, rather than just a single snapshot, from the 2D plan at each step. The latter option appears to be more feasible and practical for future development.

CastNXT:

For the next team tasked with continuing the development of the application, we recommend focusing on several critical enhancements to ensure a smooth transition to production. Firstly, there is a notable issue with the email notifications triggered by the 'Update Decks' button. Despite server logs indicating that the email has been sent, it does not reach its intended recipients. It is crucial for the upcoming team to debug this discrepancy to ensure that this essential functionality is reliable. Secondly, due to previous time constraints, certain parts of the

application were not covered extensively by tests. The new team should prioritize enhancing the test coverage. This initiative is vital for ensuring the stability and robustness of the application, significantly reducing potential risks before moving to full-scale production. Additionally, the application experiences noticeable latency issues that affect the user experience. The next team should focus on optimizing both front-end and back-end processes to improve response times and overall application fluidity. Lastly, while the app is nearly ready for production, addressing these specific areas will refine its operations and ensure it performs reliably in a live environment. These focused improvements will not only polish the application but also enhance its performance and user satisfaction upon launch.

EventNXT:

We believe that the app has been left in a solid state. All of the features requested by the client were implemented without any conflicts. The only suggestions for the next team was to handle multiple events simultaneously, add multiple guest list assistants given permission by the event owner. In the RSVP table in the Added By field a pull down menu shows all Assistants' emails, as well as, work on improved 'Mobile' view. Some more UI/UX features would be needed as per the client's request.

17. Resource Links

Teamwide:

- CSCE 606 Github Landing Page
- FashioNXT Github
- Slack
- Pivotal Tracker
- Events360 Github repository
    - **Note:** No work was done on Events360
- Events360 Heroku App

PlaNXT:

- Heroku Deployment
- Github Repository

CastNXT:

- Heroku Deployment
- Github Repository

EventNXT:

- Heroku Deployment
- Github Repository

18. Link to the Presentation and Demo videos

Youtube Link:

https://youtu.be/GcghmsIx5Kw

Google Drive:

https://drive.google.com/drive/folders/18X58iWTIT5yYw7BzFfc1bHliYhsdvfUd?usp=sharing