FashioNXT: PlanNXT

Final Report - Fall 2024

Table of Contents

1. Summary

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. The primary stakeholders of this project are the customer Tito Chowdhary, the end-users, the development team, the CSCE 606 professor Dr. Ritchey and the TA, Anirith Pampati. The three main requirements of the client were to completely sync the 2D and 3D view, introduce the chronology bar in the 3D view and implement the dependency between items and automatic timeline updates feature. Along with the implementation of these three main features, we also added many minor features and UI changes to improve the user experience.

We have managed to successfully sync both the views where we can seamlessly add, update and delete items both in 2D and 3D and the changes will be reflected in the other view. Many items that could not be added from the 2D view can now be added easily to the plan. The chronology bar in the 3D view can now be moved and the items are automatically added and removed from the plan. We can now add dependencies between items and when we either change the start time or duration to set-up or breakdown an item, the corresponding set-up or breakdown start time of the dependent items will be updated automatically. We also made significant contributions towards making the application user friendly by changing the layouts so that the important buttons are easy to reach and also introduced features such as filtering and sorting the items table and presenting the list of the items count by type that would be useful while placing orders.

2. Description of all user stories

- **Sprint 1**
    - **Feature: Understanding the legacy code**
      I want to explore and understand the legacy codebase for PLANXT so that I can effectively work on enhancing the application.
    - **Feature: Setup of application on local and Heroku**
      I want to set up the PLANXT application with tools and dependencies on my local machine and Heroku so that I can develop, test and deploy changes to the application.
    - **Feature: Add measurements hint to the input**

I want to see a hint next to the input field indicating the measurement unit so that I can accurately enter item dimensions.

- **Feature: Add a Date and Time widget for input**

  I want a date and time widget so that I can populate the fields automatically without needing to manually input this information.

Application snapshots:

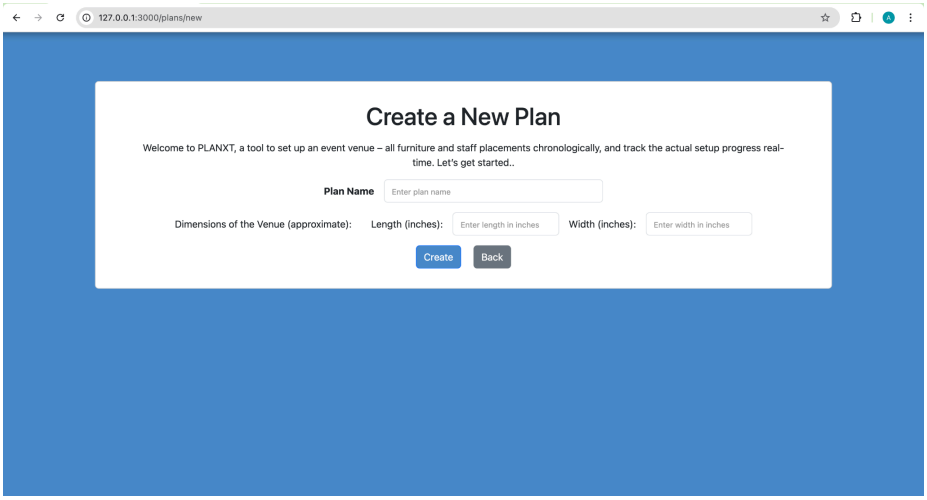


Fig 1: Implementation of measurement hint for form input

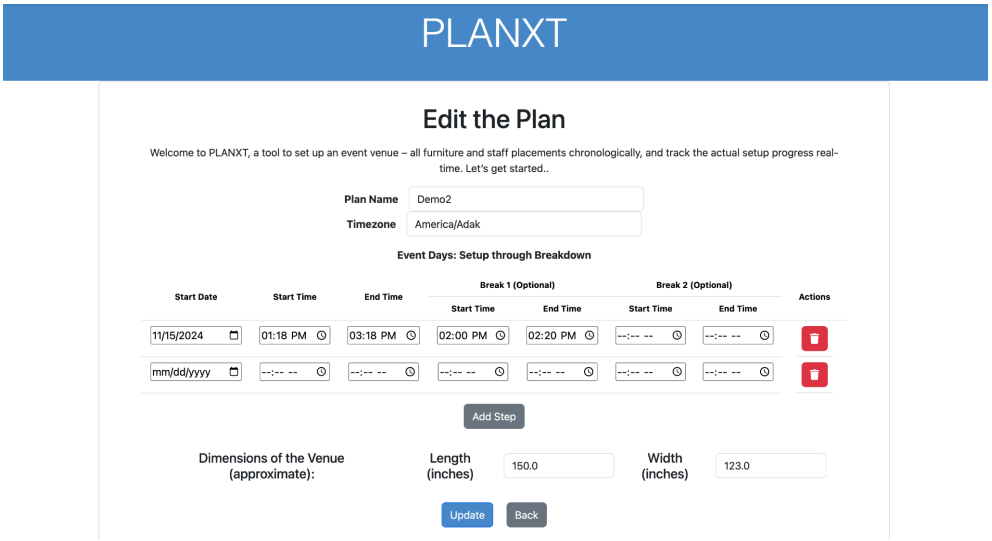


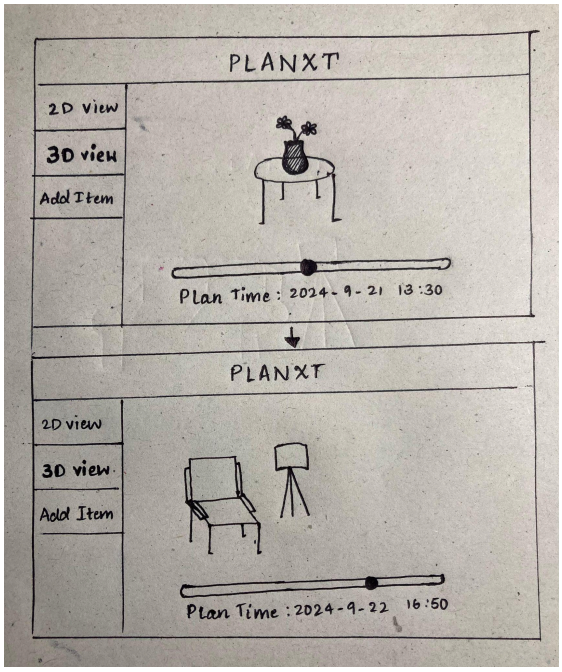
Fig 2: Date and time widget for input

- **Sprint 2**
  - **Feature: Fix chronology bar functionality in the 2D Top View**

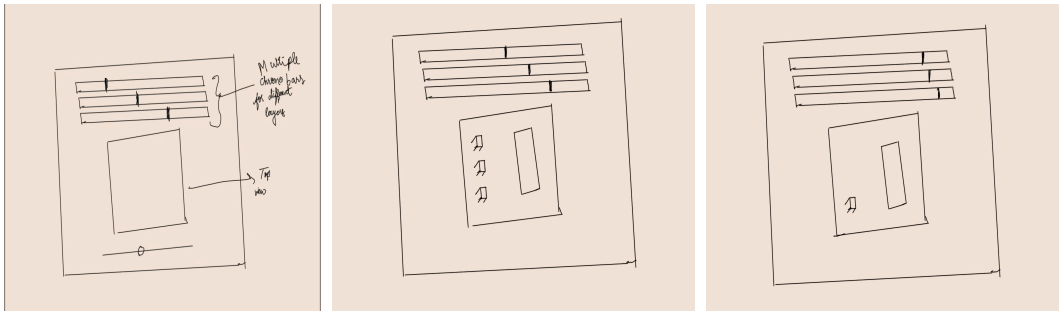
I want items to appear and disappear based on their setup and breakdown times when I scrub the 2D chronology bars so that I can view the top-down status of the venue at any given time.

- **Feature: Explore and compare 3D Interior Design tools and pick a rendering solution**
- **Feature: Reflect the changes of 3D in 2D top view**
  I want to be able to make changes in the 3D view and see those changes reflected on the 2D top view page so that the changes are consistent across all the views.
- **Feature: Chronology Bar in 3D Preview**
  I want to be able to scrub a chronology bar in the 3D preview of the floor plan (similar to the existing chronology bar in the 2D top view) and see the various items added in the 3D view at the appropriate times.

UI mockups:



3D timeline bar



Item status indicator depending on time

Application snapshots:

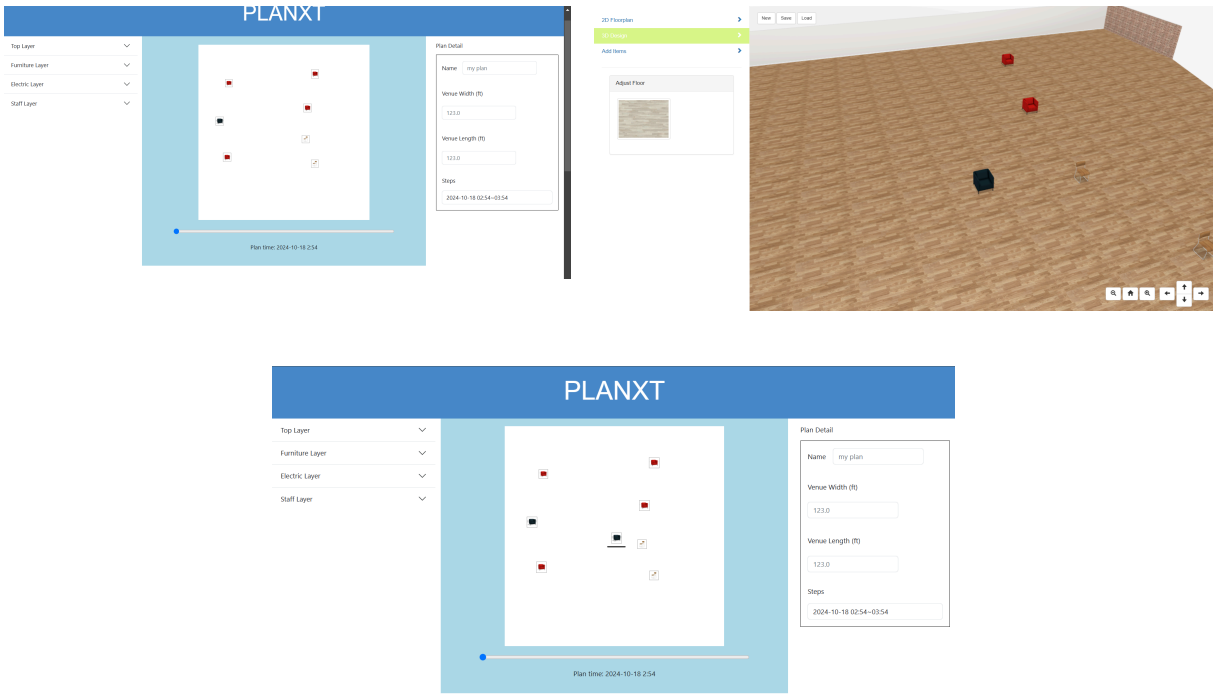


Fig 3: A blue chair added in 3D view, updates the 2D view

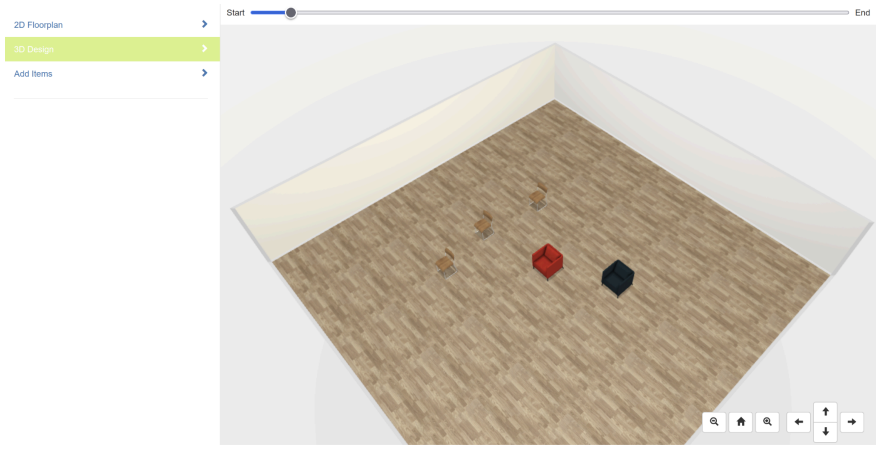


Fig 4: 3D timeline bar

- **Sprint 3**
  - **Feature: Reflect the changes of 3D in 2D top view with setup and breakdown times**

    I want a modal in 3D view to be able to add setup and breakdown in 3D view and see those changes reflected on the 2D top view page so that changes are consistent across all the views.
  - **Feature: Current plan time in 3D**

    Add functionality to update and display plan time as the slider is moved in 3D.
  - **Feature: Live scrolling functionality in the 3D timeline**

    I want a live scrolling functionality in the 3D timeline scrubber so that I can see the real-time updates on the interface.
  - **Feature: Real world scenario testing**

    I want tests to be done using a huge load with around 500 items so that I know that the app can perform well in a real world scenario.

    Application snapshots:

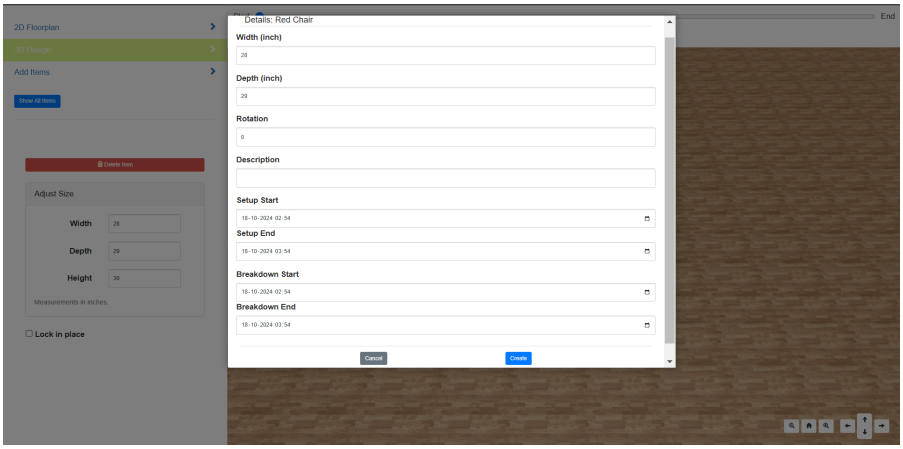


Fig 5: Input Modal in 3D view

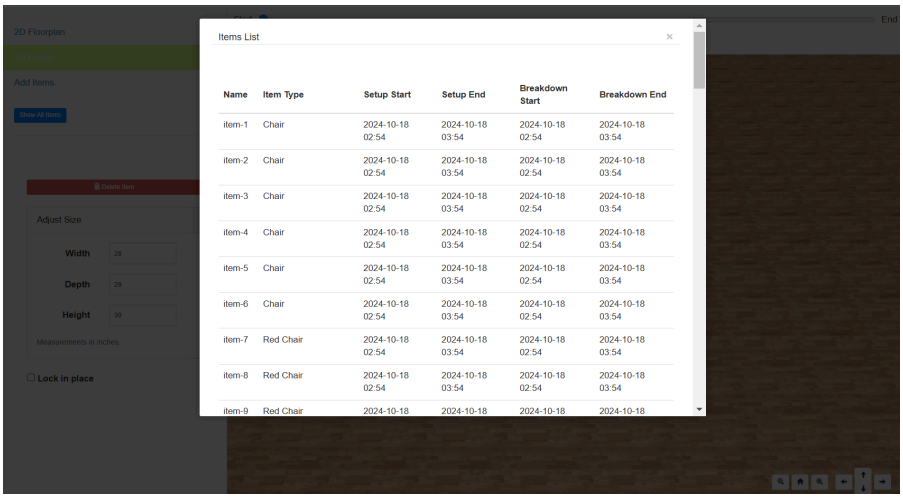


Fig 6: Items Table in 3D view

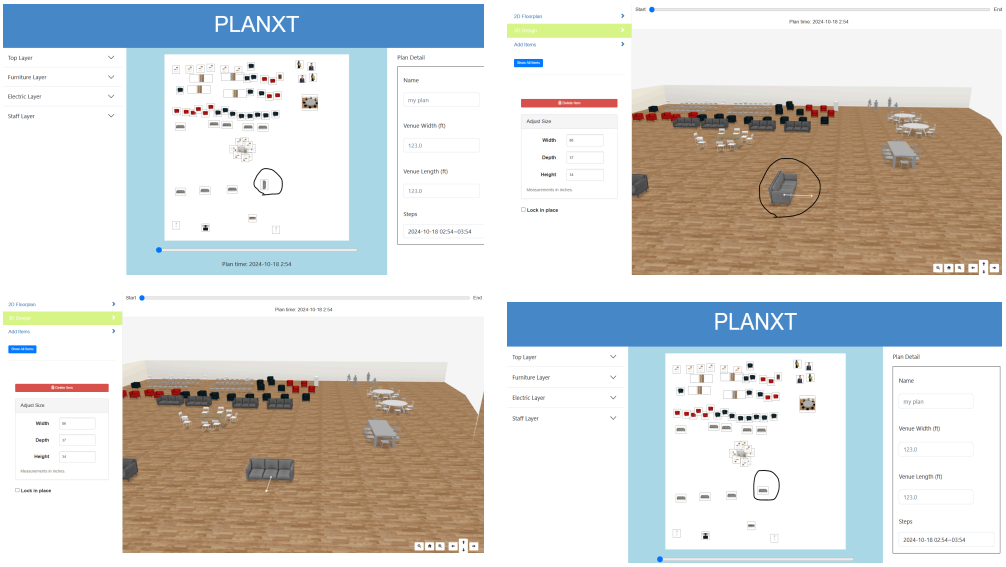


Fig 7: Sync rotation in 3D and 2D view

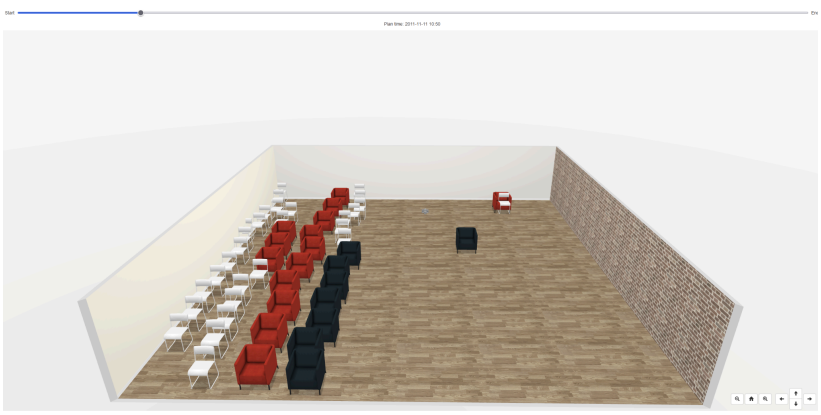


Fig 8: Live timeline bar in 3D

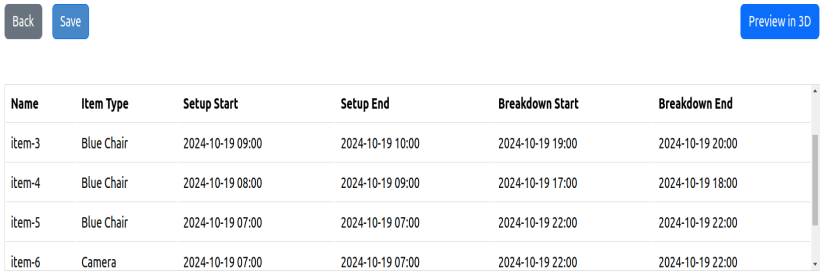


Fig 9: Updated UI in 2D plan page

- **Sprint 4**
  - **Feature: Create Dependency between items**
    I want to be able to create a dependency between two items, such as a table and a chair, where one item's time change will automatically update the other based on a time offset so that I can link related items in my schedule.
  - **Feature: View items on 2-D plan by layers**
    Add a checkbox next to each layer. Display items on the 2-D plan based on the selected checkboxes.
  - **Feature: Automatic Timeline Updates**
    Instead of selecting the end time, we can specify the duration of the setup and breakdown of an item. The end times are automatically updated only based on the selected duration. Disable change of end times. When the start time or duration of any item is modified, the start time of the dependent items is updated automatically.
  - **Feature: UI and backend Changes for Dependencies**
    I want a user-friendly interface to add and update dependencies between items so that I can easily establish connections between related items in my schedule.
  - **Feature: Filtering and search feature for tables**
    I want a table like view with the count, and a search and filter feature in the table so that I can know the count of each item and also search the items.
  - **Feature: Dependency Visualization**
    I want a new button that opens a pop-up displaying the dependencies of selected items so that I can quickly understand the relationships between items.
  - **Feature: Modify the layout of 2-D plan page**

Move the buttons above the items table. Add an internal scrolling function to the items table.

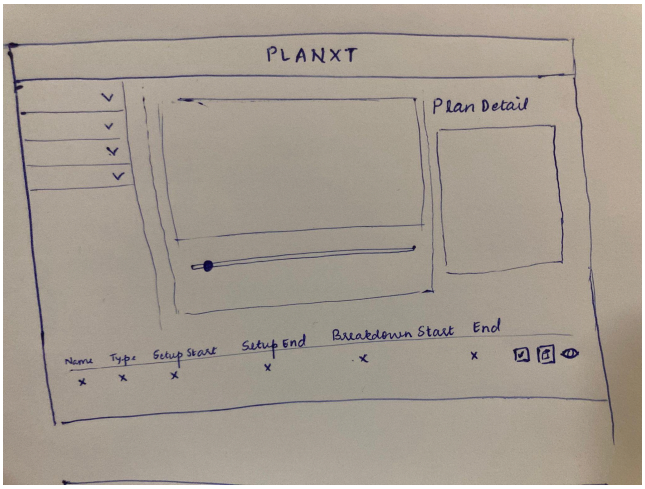
- **Feature: Exploring the possibility of customizable wall shapes and sizes in 2D and 3D**

  I want to be able to assess the possibility of adding an orthographic camera in the 3D view to see a 2D projection of the scene, and sync the existing 'create walls' feature in the blueprints library with this 2D projection so that I can implement the feature of manually changing the venue floor shape.

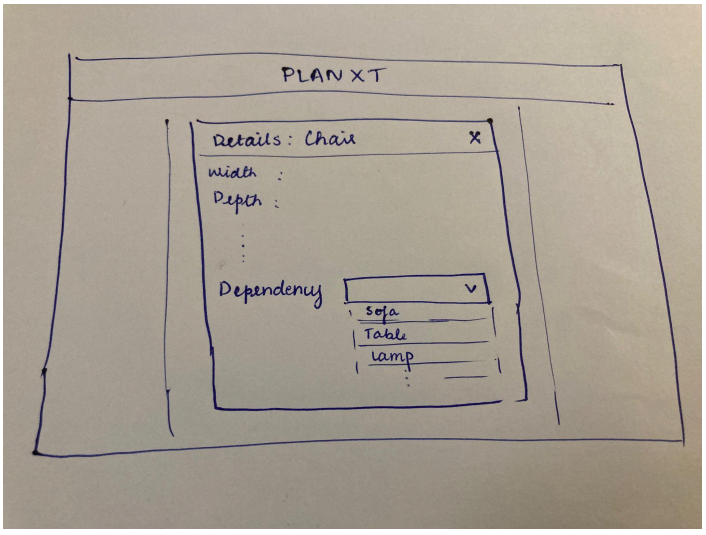
- **Feature: Add an item at the current scroll bar time**

  I want to just drag and drop the item without having to fill the time so that I can add an item at a current time in the scrollbar.

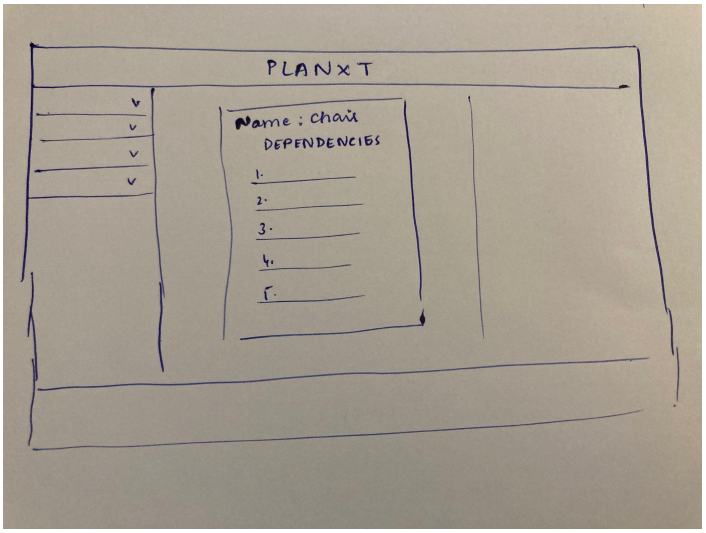
  UI mockups:



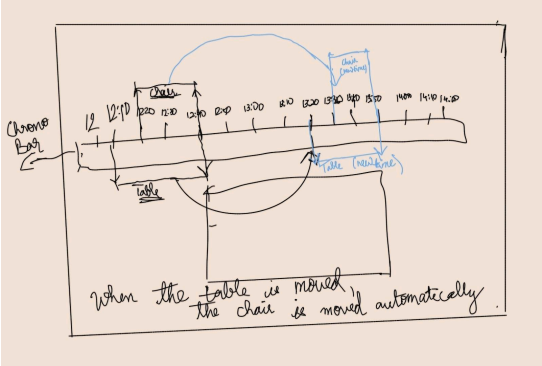
Option to view item dependency



Form giving option to choose the dependency (Add/Update/Delete)



Form displaying all the dependencies for an item



Modify a dependant item automatically

Application snapshots:

Search: Search... Back Save Preview in 3D

Item ID	Item Type	Setup Start	Setup End	Breakdown Start	Breakdown End			
item-1	Man	2024-11-15 13:18	2024-11-15 15:18	2024-11-15 13:18	2024-11-15 15:18	View Dependencies		
item-2	Woman	2024-11-15 13:18	2024-11-15 15:18	2024-11-15 13:18	2024-11-15 15:18	View Dependencies		
item-3	Chair	2024-11-15 13:18	2024-11-15 15:18	2024-11-15 13:18	2024-11-15 15:18	View Dependencies		
item-4	Red Chair	2024-11-15 13:18	2024-11-15 15:18	2024-11-15 13:18	2024-11-15 15:18	View Dependencies		
item-5	Blue Chair	2024-11-15 13:18	2024-11-15 15:18	2024-11-15 13:18	2024-11-15 15:18	View Dependencies		
item-6	Sofa	2024-11-15 13:18	2024-11-15 15:18	2024-11-15 13:18	2024-11-15 15:18	View Dependencies		
item-7	Floor Lamp	2024-11-15 13:18	2024-11-15 15:18	2024-11-15 13:18	2024-11-15 15:18	View Dependencies		
item-8	Bookshelf	2024-11-15 13:18	2024-11-15 15:18	2024-11-15 13:18	2024-11-15 15:18	View Dependencies		

Fig 10: Options to view dependencies, edit and delete item

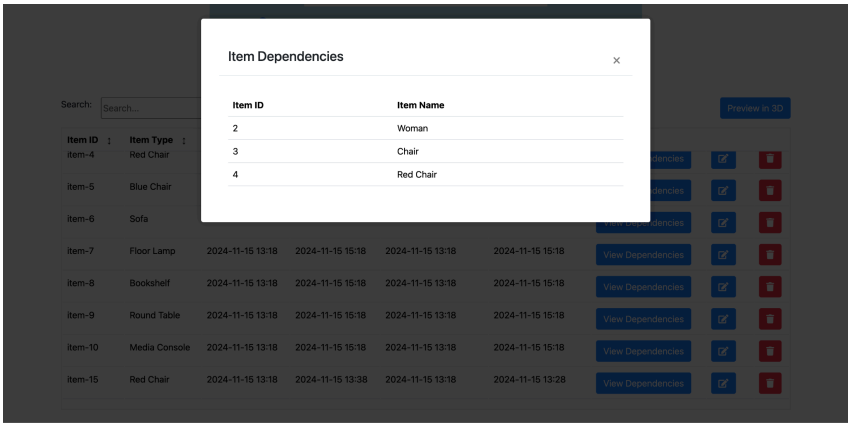


Fig 11: Dependencies of an item

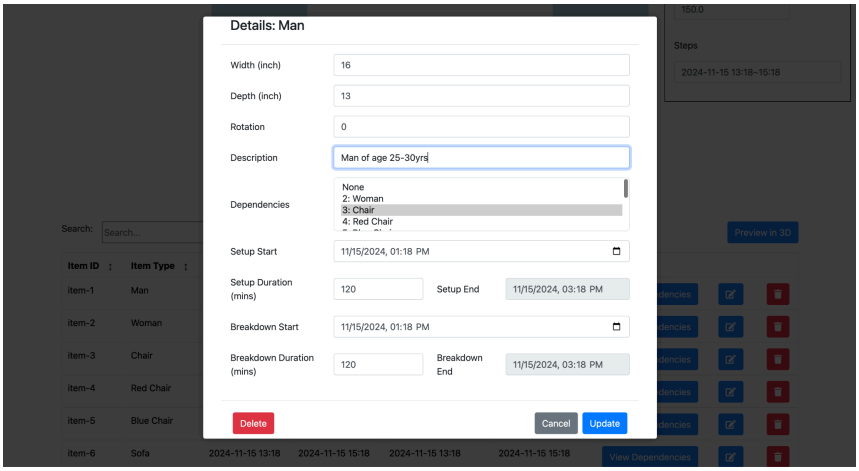


Fig 12: Edit an item

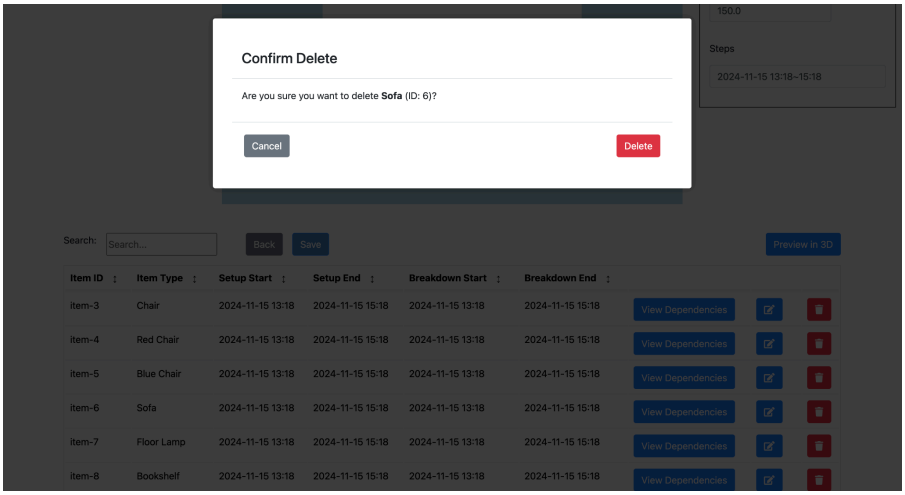


Fig 13: Delete an item

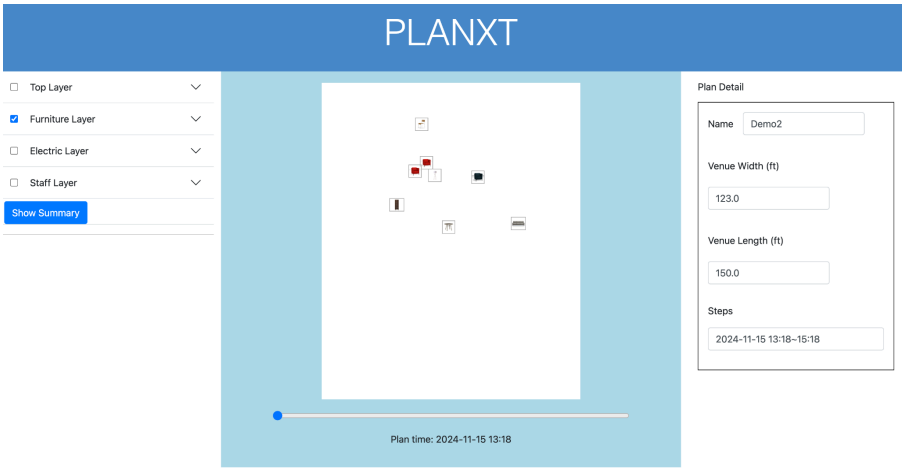


Fig 14: Option to view 2D items by layers

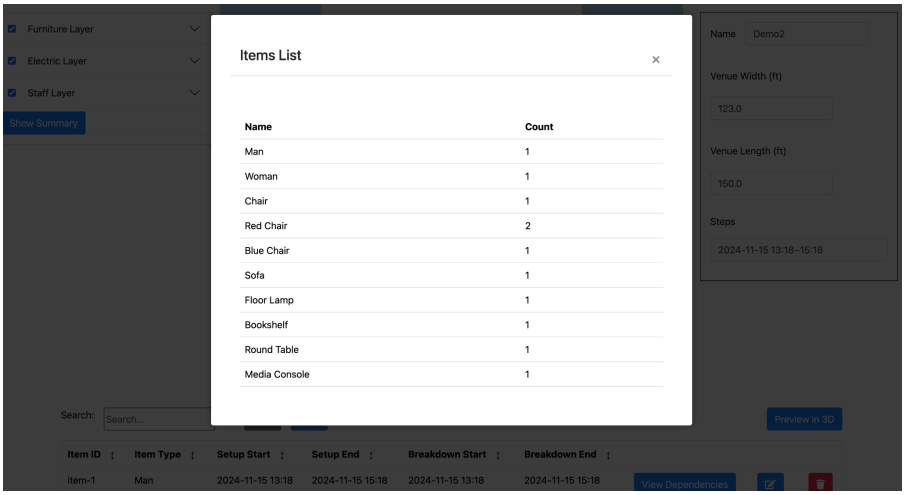


Fig 15: List of items

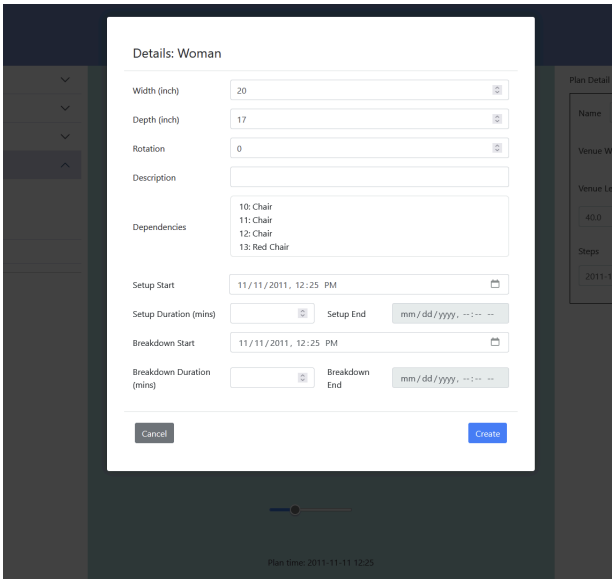


Fig 16: Automatic timeline update

Demonstrating the automatic timeline updates feature

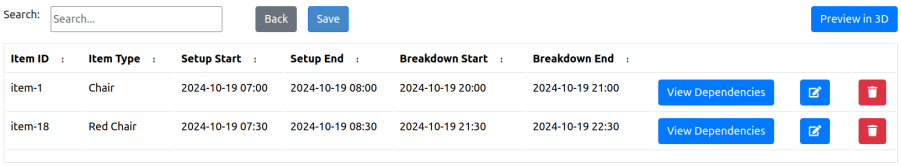


Fig 17: Initial setup and breakdown times of chair and red chair

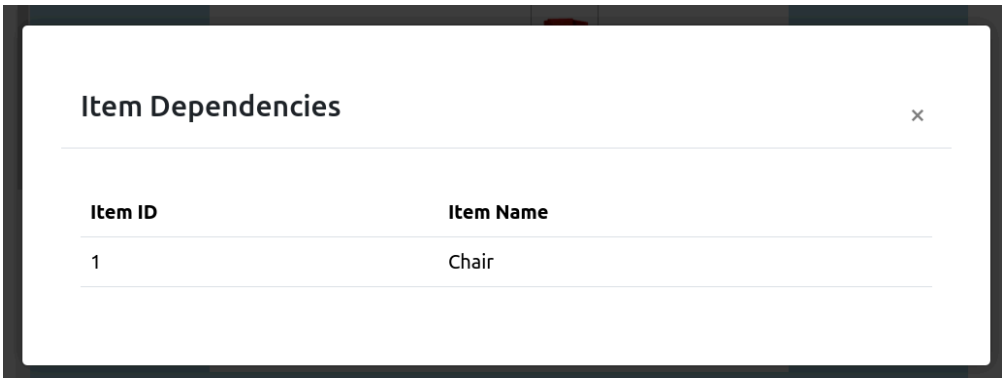


Fig 18: Red chair is dependent upon the chair

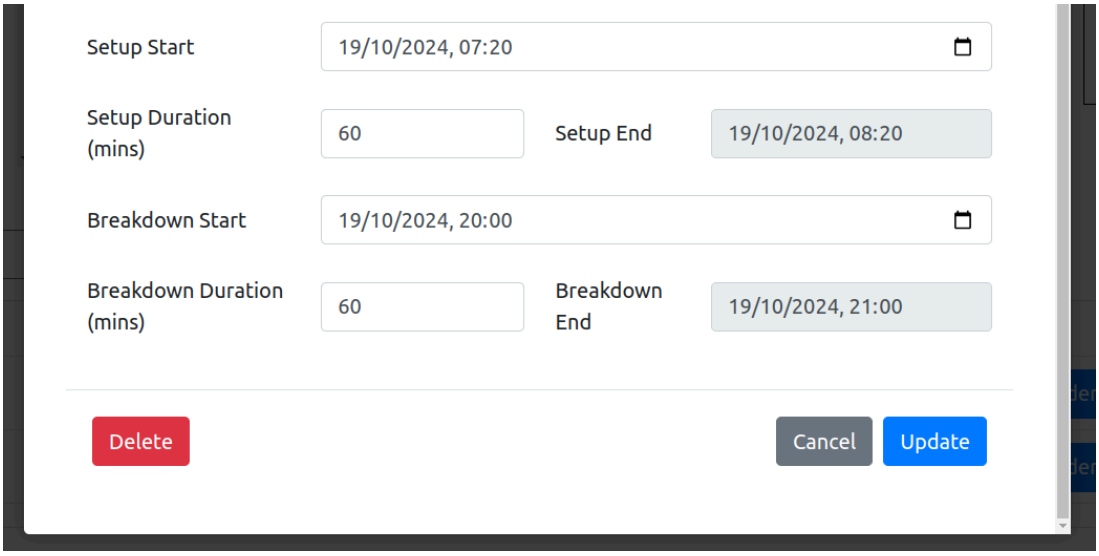


Fig 19: Updated the setup start time of the chair

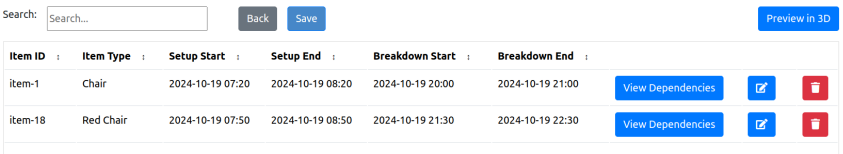


Fig 20: Setup start time of the red chair is updated automatically

3. Legacy project: understanding the existing code and refactoring

This project has been worked on for the past few semesters and we have inherited the code that has been written by several teams across the past few years.

Login

We found that the “forgot password” mechanism is not fully implemented; the link sent to the user’s email address leads to a 404 page. We have added a user story to finish this implementation. It will be necessary for us to understand the login system in order to be able to fix this issue.

2D View

When creating a new plan, adding items, or performing any action that involves inputting a date and time, the input process is very cumbersome. We have added user stories to simplify the item adding process by prefilling the date field with the date of the event. It will also be useful for the team to brainstorm intuitive UI designs for making the user experience as seamless as possible. Additionally, one of the goals of the project is to add a rich icon library to the 2D top view. It will be important for us to understand how the existing icon library and drag-and-drop system works so we can expand it.

3D View

The current system opens the 3D preview in a new tab. There are certain functionalities in the 3D view that don’t correspond to the top view and it is not clear what the connection between the two are. For example, there is the option to add walls and items to the 3D preview. In fact, there is a much larger icon library in the 3D preview than in the 2D Top view. However, making any changes here are not reflected in the 2D top view, and if you reload the page, all your changes disappear. It will be useful for the team to make design decisions about the 3D view and clearly communicate the functionality and limitations of the 3D view. One option suggested by the previous team is to show the 2D and 3D previews side by side, and have the 3D preview update live.

4. Team Roles

- **Sprint 1**

Product Owner: Mahima Ganni

Scrum Master: Govind Joshi

Developers: Ankitha Prasad Sai Siva Rohith Tirumalasetti

- **Sprint 2**

  Product Owner: Sai Siva Rohith Tirumalasetti

  Scrum Master: Ankitha Prasad

  Developers: Govind Joshi Mahima Ganni

- **Sprint 3**

  Product Owner: Govind Joshi

  Scrum Master: Sai Siva Rohith Tirumalasetti

  Developers: Ankitha Prasad Mahima Ganni

- **Sprint 4**

  Product Owner: Ankitha Prasad

  Scrum Master: Mahima Ganni

  Developers: Govind Joshi Sai Siva Rohith Tirumalasetti

5. Summary of accomplishments and points completed per sprint

Sprint / Iteration	Summary	Points
1	Every member in the team tried to understand the legacy code and set-up the project locally and on Heroku. Additionally, measurement units like feet and inches were added to input and date and time were automatically set to the plan's date and start time while creating an item.	**46**
2	The functionality of the chronology bar in 2D plan was implemented. We decided to continue with the existing 3D rendering tool and also added the chronology bar to the 3D view. The syncing (add, update and delete items) of the 2D and 3D view was also successfully completed.	**12**
3	A form to update the items was introduced in the 3D view. The live scrolling functionality to view real-time updates as the scrubber is moved was implemented and the current plan time was also added to the 3D view. We also performed a real-world test to check if	**19**

	the application could handle a large number of items.	
4	The feature that allows items to be dependent on each other so that the start time of the later item can change automatically when either the start time or duration of the former item changes was introduced. Features to filter and search in the items table and view items by layers were introduced in the 2D page. The layout of the 2D page was modified and a feature to add items at the current scrollbar time was added. Additionally, we explored the possibility of changing the shapes and sizes of walls both in 2D and 3D view.	**27**

6. Table of user stories points completed by each member

Name	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Total Story points	Percentage
Govind Joshi	2 stories 10 points	2 stories 7 points	1 story 3 points	2 stories 6 points	7 stories 26 points	25%
Mahima Ganni	2 stories 10 points	1 story 5 points	2 stories 8 points	1 story 3 points	6 stories 26 points	25%
Sai Siva Rohith	3 stories 13 points	0	1 story 3 points	2 stories 10 points	6 stories 26 points	25%
Ankitha Prasad	3 stories 13 points	0	1 story 5 points	2 stories 8 points	6 stories 26 points	25%
Total	10 stories 46 points	3 stories 12 points	5 stories 19 points	7 stories 27 points	25 stories 104 points	100%

7. Customer Meetings: Dates and Summary

Sprint 1 :

Customer Meeting Time: Oct 4, 2024 3:00 PM & Oct 11, 2024 3:00 PM on Zoom

In the meetings, we demoed the project running locally on our machines. We presented our ideas for adding value to the product in the next sprint. Specifically, we mentioned that we would add a visual indicator to the items on the 2D canvas to represent that they were being set up, were already set up, or were being broken down. Additionally, we mentioned that we would work on

adding a dependency feature between items. Tito mentioned that we do not want to chase low hanging fruit, and instead focus on the big picture features. He wants us to prioritize the dependency feature, and in fact he would first like to see the chronology bar in 3D. He is not married to the existing 3D rendering tool, and would like us to explore other options and quickly decide on a new one (or choose the existing solution). We also presented the idea of item groups, and while he liked the idea, he wants to see the product completed with the main features first.

Sprint 2:

Customer Meeting Time: Oct 18, 2024 3:00 PM & Oct 25, 2024 3:00 PM on Zoom

In the meetings, we showcased the progress made during the current sprint by demonstrating the recent changes we implemented. The client provided valuable feedback, highlighting areas for improvement and additional features needed. The client expressed a desire for live scrolling functionality in the 3D timeline scrubber developed during this sprint, which would enable real-time updates in the interface. The client also emphasized the importance of prioritizing a dependency feature between the objects. We informed the client about the development changes made for 2D-3D synchronization, acknowledging that further work is needed in this area for the next sprint. Additionally, the client also encouraged us to consider the intensity involved when upgrading the current legacy code or libraries. Lastly, he requested real-world performance testing to assess the system's ability to handle a large load (more than 500 items) and to confirm that effective rendering is maintained under such conditions.

Sprint 3:

Customer Meeting Time: Nov 1, 2024 3:00 PM & Nov 8, 2024 3:00 PM on Zoom

During our meetings, we demoed the syncing 3D to 2D feature and the live 3D scrolling feature and had several discussions about the next direction in which to take this application. The client requested several new features and we documented all of the additional functionality in which the client was interested. There were a whole host of features including an online collaboration and access control functionality with the ability to share plans with other users, leave comments on different aspects of the plan, and so on. We also presented a survey of existing floor planning tools at the request of the client, and the client picked his favorite tool and the specific features

that he would like to see from these existing commercial applications in our application. However, we set up realistic expectations with the client and picked a few primary user stories to focus on for the next two weeks. Specifically, we want to add all the functionality unique to Tito's vision. The live timeline bar feature is complete, and therefore the next priority is the dependency feature. Minor UI changes have been added as pending user stories. All other changes have been documented, but will not be focused on, at least for the upcoming sprint.

Sprint 4:

Customer Meeting Time: Nov 15, 2024 3:00 PM & Nov 22, 2024 3:00 PM on Zoom

During our meetings, we demoed the dependency feature and the modifications done to the items table and had several discussions about the next direction in which to take this application. The client was happy with our progress and all the changes done till now as we had completed the two major features expected for this semester plan - the synchronization of the 2D and 3D view and the dependency feature. He has a few features in mind that the future team could implement and we documented all of the additional functionality in which the client was interested. There were a whole host of features including an online collaboration and access control functionality with the ability to share plans with other users, leave comments on different aspects of the plan, and so on. He also wanted an additional chat-like feature where he could send the orientation document to the team. All these changes were documented and will be shared with the future team.

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

We integrated both Behavior-Driven Development (BDD) and Test-Driven Development (TDD) in our development to create robust, high-quality software that aligns closely with clients needs and technical standards.

BDD Process - We begin with BDD to establish a shared understanding of the desired application behavior between the client and us. We had meetings with the client every week to discuss requirements and define behaviors. Then the requirements are broken down into user stories, focusing on specific functionalities from the user's perspective. We used the *As a - So that - I want* format to write clear, concise scenarios that describe the context, action, and expected outcome of each behavior.

TDD Process - Once we had a clear set of behaviors, we moved into the TDD cycle. We first create a test that defines the desired functionality. We then implemented the code to make the test pass. And then we improved the code structure while ensuring all tests still passed.

Benefits:

1. BDD allowed for better understanding between the client, product owner and the developers by using a human readable language that described the behaviour of the features..

2. TDD led to cleaner, more reliable code with reduced redundancy. This allowed us to only include only the necessary features and reduce the likelihood of the bugs.

3. Both BDD and TDD ensured that development is driven by actual requirements and user expectations.

4. By using TDD we were able to identify any bugs and correct any issues at the beginning itself.

Problems:

1. Writing tests before code and creating detailed behavior scenarios slowed down the development initially.
2. As the project evolved, maintaining a large suite of tests and behavior scenarios was a bit time-consuming.

9. Configuration Management approach

Our team implemented a robust configuration management strategy using Git for version control. This approach allowed us to effectively manage our codebase, track changes, and collaborate efficiently throughout the development process.

We utilized Git as our version control system, which provided us with the following benefits:

1. Distributed development, allowing team members to work independently

2. Easy branching and merging capabilities

3. Efficient collaboration through pull requests and code reviews

Branching Strategy - We adopted a feature branching workflow, which aligned well with our agile development process:

1. Most of the user stories and features were developed in its own branch (we had around 20 branches)

2. The main branch was kept stable and always contained the latest production-ready code

3. Pull requests were used to merge feature branches back into the main branch, ensuring code quality through peer reviews

To streamline our workflows and enhance collaboration, we frequently released updates and each release was thoroughly tested and validated before being merged into the main branch. This enabled us to gather user feedback early and often, and be able to continuously improve the application.

During our project, we did not need to implement any formal spikes. Our team was able to address uncertainties and technical challenges within the scope of our regular user stories and sprint planning.

10. Issues encountered in the production release process to Heroku

Our team's deployment to Heroku was largely smooth, with only one notable issue related to database changes. After making changes to our database schema locally, we noticed that these changes were not reflected in the Heroku production environment. We then had to manually run database migrations on Heroku after deploying code changes that included database schema updates.

11. 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** served as our central version control system, playing a crucial role in managing our codebase and enabling team collaboration. Its features, like pull requests, issue tracking, and branch management, were essential for maintaining code quality, tracking bugs and improvements, and coordinating work efficiently. The platform's integration with other tools streamlined our workflow, allowing us to collaborate seamlessly. However,

challenges such as merge conflicts, especially during large sprints, and ensuring all team members followed branching and commit conventions, occasionally slowed progress.

2. **CodeClimate** was used for automated code review and quality analysis, which helped us maintain high standards by identifying code complexity, duplication, and potential security issues. Its integration with SimpleCov provided real-time test coverage reports, helping us ensure thorough testing. The detailed feedback from CodeClimate allowed us to continuously improve our code. However, it sometimes flagged non-critical issues, leading to unnecessary changes, and required some customization to better fit our specific project needs.

3. **SimpleCov** was essential for monitoring test coverage in our Ruby applications, helping us ensure we met our 90%+ test coverage goal. It generated clear HTML reports that made it easy to identify untested code, allowing us to focus on improving test coverage where needed. Its seamless integration with our test suite provided 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.

4. **Heroku** made deployment of our application incredibly easy by allowing us to push code live without worrying about server setup or configuration, thanks to its seamless Git integration. It also provided sufficient scalability for our small project, with the option to scale resources as needed. However, the free tier's limited resources sometimes led to performance issues, especially when traffic increased. Additionally, certain advanced features like databases and caching required paid add-ons, which could raise costs if the project expands further.

5. For the PlanNXT project, we used **RSpec** for unit testing and **Cucumber** for Behavior-Driven Development (BDD). RSpec's flexible syntax made it easy to write clear tests for models, controllers, and views, while Cucumber allowed us to write tests in plain English, making the application's behavior more understandable for non-developers. However, maintaining these tests as the codebase evolved was time-consuming, especially during refactoring.

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.

12. Code is pushed to GitHub

All code has been pushed to the GitHub repository.

13. Repository contents and deployment process

All the information needed to deploy the application is detailed in the README.md under the section 'To deploy in production environment (Heroku)' and all necessary files to run the application and test it are available in the GitHub Repository.

14. Important Resource Links

Application Page: https://planxt.herokuapp.com/

Github: https://github.com/CSCE-606-Event360/Fall2024-PlaNXT

Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2721605

15. Link to the Presentation and Demo videos

Presentation: Presentation Video.mp4
Demo: Demo.mp4