NYCCal!

Final Documentation

CSCI 499

December 20, 2023

Brajan Halili and Nick Melkadze

Table of Contents

1. Domain Analysis

Event finder applications have a large collection of events that are overwhelming to users. Although having a lot of choices to choose from might look appealing at first for users, it could also become a very demotivating factor. A study detailed that a large amount of options can lead to dissatisfaction and become counterproductive. Specifically there was a study conducted on students willingness to take on extra credit assignments based on the amount of options given. The results showed that students were more willing to take on extra credit when given less options and their quality of work for extra credit improved (Iyengar & Lepper, 2000). In addition, even when tourists have knowledge of the destination, providing many choices will still have a large impact on consumers who can get overwhelmed. This makes filtering of content shown important (Park & Jang, 2013).

Ticketmaster is the main ticket event sales company that people use. It is designed for people to find tickets for their favorite artists. It has regional and personal preferences for its users. However, its hundreds of events among dozens of categories could be overwhelming for the user. It is also designed for customers to purchase high priced tickets because of their high demand.

This decision fatigue plays a role in decision making when it comes to choosing events to attend. Users can get overwhelmed seeing a lot of choices to attend and become unmotivated. NYCCal plans to lower the choice overload for NYC residents when it comes to finding nice events they are interested in. By focusing on user personalization, the app will reduce options and provide a few choices for the user to consider. If the user is not satisfied, they will be able to click a button to show the rest of the events for them to consider. Our app is also focused on NYC and increasing event attendance. Thus, the focus will be on increasing event attendance in local, lesser known events as well as provide a quick way for the users to find events they like.

2. User Requirements

1. The user shall be able to log in their account and access their profile settings. This requirement will exist because the app's focus is on personalization. Since many users place importance in their event preferences, they would not like to input their preferences every time their registration in the site has expired.
2. The user shall be able to select their preferences when they create the account. Many users would like to attend events based on their preferences so getting their preferences right after they log in for the first time is useful.
3. The user shall be able to change their preferences in their profile settings. User's preferences do not change that much often to create an entire button on the menu. Instead they can be in the profile settings to simplify the main navigation menu.
4. The user shall click on the "Events" button and see a few events after clicking on a day in a calendar setting. Calendar setting provides a clean UI look, lowering the time requirement for users to get accustomed to the website. The number of events shown immediately will be 1-3 per day in order to lower choice overload.
5. The user shall be able to see main details about the events after they click a day in the calendar. They will be able to see a very broad overview of the event that will be enough for the user to consider as an option.
6. The user shall be able to click on the event card to learn more about it in order to provide more information for the user to make a decision.
7. The user shall be able to click on the "See more events" button after clicking in a day if they want more choices. This works to help the user choose events if they are not satisfied with the recommendations. Putting an extra button as a barrier works as a balance between decision fatigue and users who can be unhappy with the events chosen.

8. The user shall be able to mark the events that they will attend. This will create a record of the events they are looking forward to being present at.
9. The user shall be able to share event details by clicking the “Share” button and pasting the information where they want.
10. The user shall be able to view future events they are planning to attend by going to “Attendance” and seeing the event listings there.
11. The user shall be able to view events they have attended in the past by going to “Attendance” and clicking to see previous events.

3. System Requirements

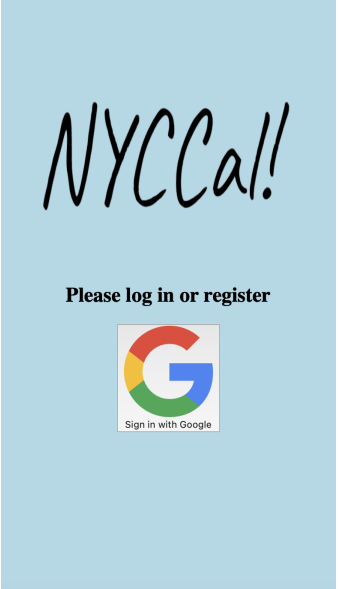
1. The system will be able to create an account based on the gmail provided by the user. This account can be accessed every time afterwards if the user has access to the gmail.
2. The system will prompt the user with a list of preferences right after the creation of the account.
3. The system will have an option to change preferences at any time in the user's settings.
4. The system shall generate 1-3 events in a day to be easily accessed in the calendar interface.
5. The system shall redirect to the event day when a calendar date is clicked.
6. The system must take less than 3 seconds to load up a page.
7. The system must take the user's preferences into account when displaying events in the calendar.
8. The system shall be able to display the event cards after a specific day is clicked in the Calendar. Each card will include: Title, date and time, summary, users signed up and price as an event overview.
9. The system shall provide full description, accessibility options and contact information in addition to overview details when the user clicks to see more details.
10. The system will provide more events (if such events exist) when the user clicks "See more events". The display will just be an overview of each event.
11. The system shall log the attendance of the user when the user clicks on the "Sign up" button.
12. The system shall take less than 2 seconds to validate the attendance of an event.
13. The system shall copy the event overview information and a link to the specific event.

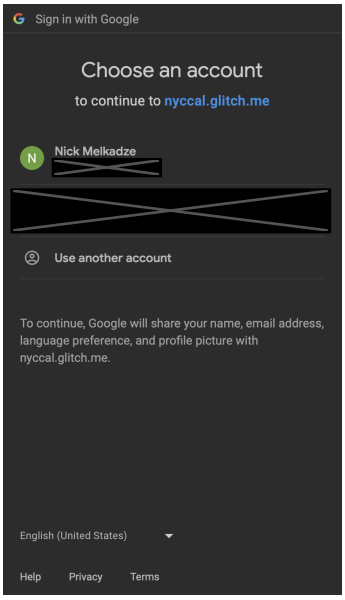
14. The system shall take less than half an hour for users to be accustomed to. It will be very simple to navigate.
15. The system will display any events the user has planned to attend or already attended in a list in the “Attendance” tab.
16. The system shall protect the users data by not displaying any database information in public files.
17. The system shall get OS data for debugging purposes.
18. The system shall be available in every device with an internet connection, although its UI is designed for mobile use.
19. The system shall have a color contrast ratio of at least over 3.0 .
20. The system shall have a rate of failure of less than 3 occurrences per day.
21. The system shall have less than 5% of events causing failure.

4. User Interface Design

NYCCal! is broken up into five primary views, two internal views, and an external Google view. The focus of our project is on mobile usage, in order to best cater to the vast audience that is smartphone users and to encourage consistent engagement with our application wherever the user is. Thus, we will be showing mobile screenshots and usage instructions.

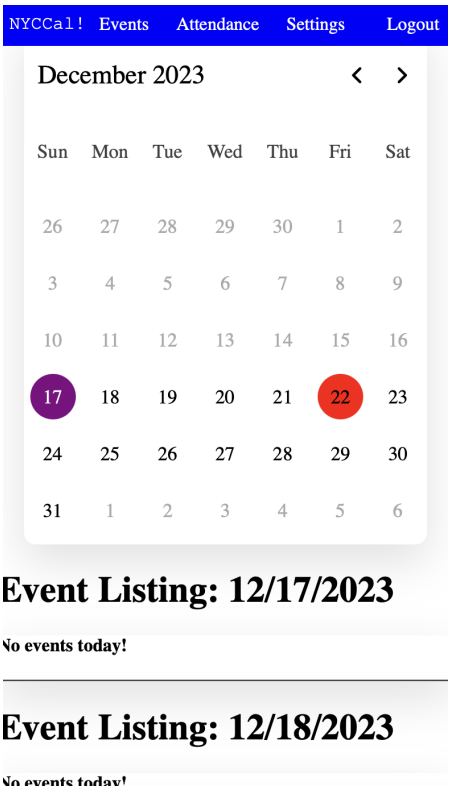
The first page a user will see is the Welcome page, accessible at the root URL "/", but truly located at "/auth/login". It displays the application logo and prompts the user to click the Google sign in button to log in or register an account (the system will log them if their account was found, or dynamically create a new account if one was not).

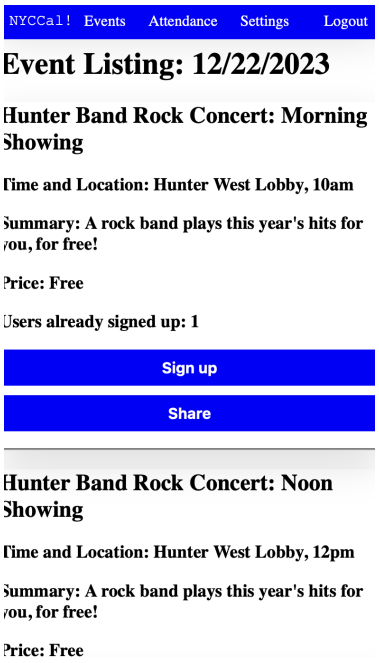




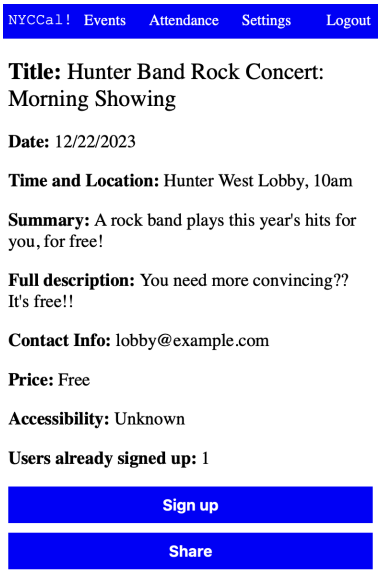
The next page is the external Google oAuth page. This page is located at "/auth/google". Here, a user will select the account they will log in to our application with (or sign into it if it is not already on the list), and be immediately redirected to the Events page on an existing user's login, or sent to the Settings page to select their preferences if they are a new user.

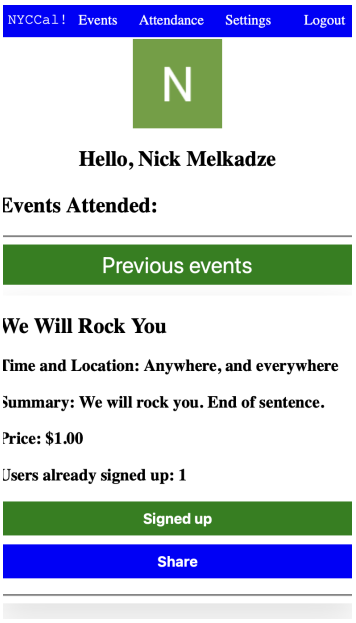
The main page of the application is the Events page. It is available at "/events", and will be where most users find themselves after signing in. It (along with the Attendance and Single Event pages) includes a header that links to each of these pages, along with the buttons to alter settings and log out of the application, and the title of NYCCal!. Below the header, there is a calendar widget which marks the current day in purple and includes a heatmap element with the number of events on other days. Past days are grayed out, and as are previous months if they are navigated to via the widget. When scrolling down, event listing for the current day and all further days in the selected month are shown. Events include their titles, times/locations, summaries, prices, and the number of users already attending the event. There are buttons to Sign Up (which will track the event in the Attendance page) and Share (which will copy a link to the relevant Single Event page along with a description of the event to the clipboard). The selected month can be changed by using the left and right arrows on the calendar. Clicking on any event will navigate to their Single Event page.



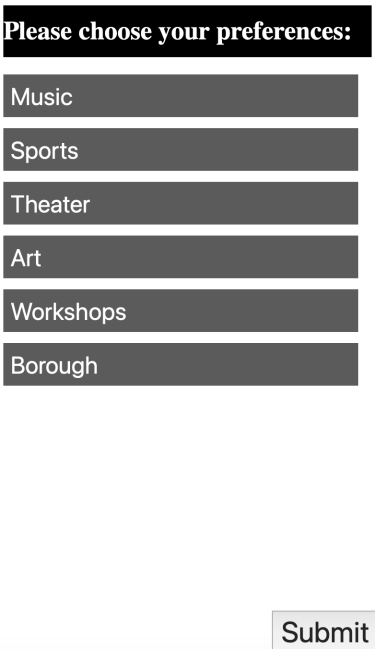


The Single Event page is accessible by clicking on any event, or by sharing a link via the Share button. Such links are in the format of "/events/single/<ID of the event>". It includes all of the public information for an event, including its title, date, time/location, summary, description, contact information, price, accessibility information, and the number of users who signed up for it. Like the Events page, it contains buttons to Sign Up for the event, or Share it to others.

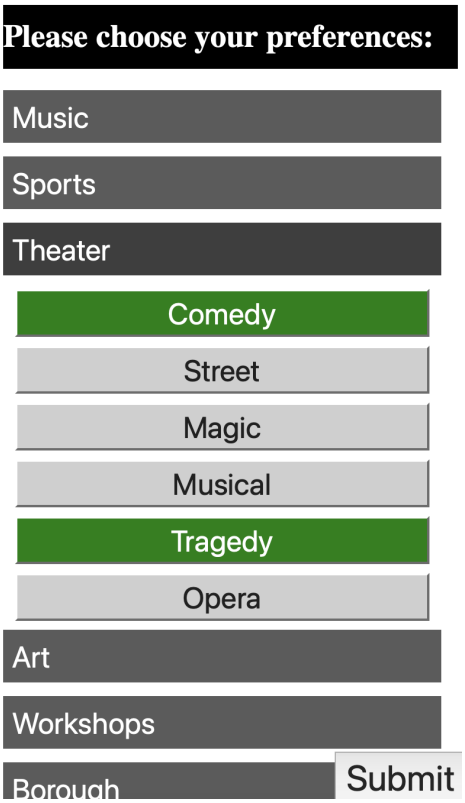


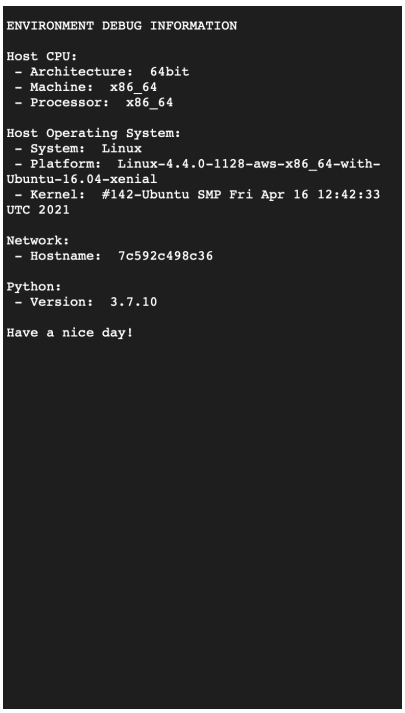


The Attendance page (located at "/attendance") shows only the events that the current user is currently registered for, or was registered for and already attended. Besides this difference, the events themselves contain the exact same information and button functionality as on the Events page. At the top of this page, there is a "Previous events" button which, when pressed, will expose all of the events that the user signed up for but have already occurred. This allows our users to keep their focus on the upcoming events, while still retaining the ability to see what they experienced as a result of finding events on NYCCal!



The final primary page of the application, the Settings page, is located at "/settings". It is shown when the user clicks the Settings link in the header, or when the user registers an account with the application for the first time. It provides a way for a user to select what types of events they want to see, by only showing them events on the Events page if they match at least one of the preferences selected here. Each of these six categories can be expanded by clicking on them to reveal the preferences themselves, and their selection can also be toggled by clicking them. Users will press the Submit button once they have decided on their preferences, and will be navigated to the Events page afterwards, with the events already filtered according to their settings.





The first of two internal (not intended to be accessed by users pages is the debug page (at /debug), which features a printout of environment information such as the host's processor and OS, the network, and the version of Python that is executing the debug code.

The second internal page is the Events Creation page at ("/events/add"), which allows NYCCal! administrators to easily add events.

5. Architectural Design

NYCCal! uses a traditional client-server model with the implementation of a Node server backend that serves web pages capable of executing their own client JavaScript code.

The Node backend is responsible for handling requests sent to any of its routes, and is designed to both respond with webpages (for routes specific to our client's implementation) and traditional CRUD responses (for routes that handle general database read/write actions).

The web client uses the JavaScript Fetch API to communicate with the Node backend, and gets its data through using the routes that any frontend application can utilize. This client is limited primarily to the server's exposed /public folder, which includes subfolders for images (/assets), CSS code (/styles), and JavaScript code (/scripts). The HTML files themselves are dynamically generated by the server.

In addition, while all information used by the web client is exposed via the server's routes (given proper authentication), there are dynamically rendered elements where their inclusion would result in a more simple and efficient design. However, since we wanted to utilize a client-server model in order to harness the benefits of a backend which can be used in any number of further releases (such as native mobile versions), our focus was on creating robust traditional routes for all of our database functionality.

6. Database Design

NYCCal! includes four database models: Users, Preferences, Events, and Attendances. Preferences and Attendances are owned by a User, and the latter refers to an Event by ID. All models record their timestamps of creation time and update time, and have their own database identifier.

Users represent any person who is meant to utilize the system to attend events. The system populates all fields of a user based on Google's response after they sign in with oAuth. These fields include their full name, their unique Google identifier (which is used to authenticate users), their email address (currently unused in the system), and their thumbnail's URL (we did not want the overhead of storing images on our server, and can have thumbnails auto-update if users change them by relying on Google's URLs). Users can own Preferences and Attendances.

Events are the centerpoint of our application, and represent anything our users would want to sign up for and attend. They include fields for name (the event's title), date (kept in the JavaScript Date format, but interpreted by the system as a human-readable date), appointment (at what time and where the event takes place), summary (a short description), description (a long description), contact (a way of reaching the organizers; usually an email address or website), price (kept as a string to allow for descriptions, such as scaled pricing dependant on age), accessibility, and tags (a space delimited string of attributes which are compared with the User's preferences). Of these fields, only tags is hidden from the user.

Preferences are the tags which a User wants to narrow their events down by. Events will only be shown to users if one of their tags matches a tag the user has a preference for. Preferences themselves are relatively simple, with fields for the tag name, true/false status, and the user who owns it. These Preferences are set on sign-up, or by clicking Settings when logged in.

The final model is Attendances, which represent if a User sets that they will attend an Event. Users do this by clicking the Sign Up button under an event, and can reverse their decision by clicking the button again. The fields for Attendances are a true/false status, and references to both of the specified Event's and User's internal IDs.

7. Class Diagram

As we did not use an object-oriented framework, we cannot provide a class diagram.

8. Testing

The testing was done manually after the addition of new features to ensure that the application did not stop working properly after each feature merge. This had a magnifying result as the testing period became larger and larger. We focused on following user stories that were created from the user requirements as a guide even though we also used some more specific cases to test the product. The complete list of the cases was:

- Login page
    - Click the login button to see if there is a prompt to select gmail account
    - If it is the first time logging in, check if the settings page shows up.
- Menu bar
    - Can click the menu options and go to the right tabs.
    - Stays on top of the page.
    - Logout works as intended.
- Settings page
    - Submit button adds preferences.
    - Select preference and check if the events shown match.
    - Change preferences and see events shown change.
- Events page
    - Current month calendar starts with the current day event listing, previous months show nothing, future months start the listing from the first day.
    - Add events to see if the event shows up in the calendar.
    - Add 1,2,3 and then 3+ events on different dates to see if the date color changes in the calendar.
    - Click the calendar date to see if it leads you to the date.

- Event card
    - Has title, time and location, price, summary and number of users attending.
    - Click “Sign Up” and check if the event shows up in Attendance.
    - Click Share and see if the event info is pasted correctly.
    - Click on an event card to see if the single event page opens.
- Attendance page
    - Shows the users name and their profile picture at the top.
    - Displays attendance history if it exists, button is clickable otherwise.
    - Displays events attendance after the attendance history.
    - Unclick the “Sign up” button to see if the event disappears from the page.
    - Click on the card to see if it leads to the single event page.
- Single event page
    - Check to see if the information given includes the full description, accessibility information, contact info as well as the information in the event card overview.
    - Check to see if the “Sign up” and “Share” buttons work correctly.

Testing helped find bugs that might have happened when code cleanup occurred or new features were added. It also helped us with the testing when we faced challenges. For example, we had issues passing along the event ID info for the “Sign up” button, making the hitbox for calendar clicking as well as filtering the events and sorting by date. These test cases helped with catching issues and debugging. We followed those steps before finishing the weekly reports, making sure our progress did not hurt any part of the system by successfully passing all the test stories.

9. Ethical Considerations

Our product collects important data from the users. We record their preferences, their attendance history as well as future event attending plans. Thus, it is our obligation to guarantee the safe collection of data as well as the safe handling of data by not selling it to third party entities. This is because it is not ethical to sell such information that can cause real harm to our users. Knowing event attendance or preferences of people can lead to great security risks for our users such as: harassment, stalking and/or bodily harm.

When it comes to the event creation side, the addition of unethical events in our platform can also lead to problems. For example, we would not allow the addition of a coding workshop that teaches attendees to exploit system vulnerabilities of a certain service. This would lead to harm being done to the users, employees and the company that is going to suffer from the exploit. Thus, it would not be ethical to host such events on our platforms.

10. Project Plan

NYCCal! was developed by Nick Melkadze and Brajan Halili. Nick handled the general server logic and setup, backend routes, database models, connection with MongoDB and Google oAuth, Python integration, and events fetching on frontend pages. Brajan implemented the Login page frontend, Events page (with calendar clicking, event cards, and switching months), Attendance page (with attendance history and attendance list), Settings page (with buttons to open the category panes and category buttons), and the header bar.

Below is a week-by-week breakdown of the application's development:

WEEK	DATES	Nick (Backend)	Brajan (Frontend)
1	**9/25 to 10/1**	User interviews and requirements document	
2	**10/2 to 10/8**	App template and MongoDB setup	Calendar framework
3	**10/9 to 10/15**	Google oAuth setup	Finished calendar and header
4	**10/16 to 10/22**	Full end-to-end login support	Events framework and event listings by day
5	**10/23 to 10/29**	Pages linked to EJS; Preferences functionality	Completed Events page and buttons; started Preferences
6	**10/30 to 11/12**	Prompt users to set preferences if new; started Sign Up button functionality	Sign Up button can be clicked, Share button copies info, and More Events button works
7	**11/13 to 11/19**	Sign Up was finished and implemented, and Events route was updated	Frontend for User Attending page; calendar clicking and heatmap on Events page
8	**11/20 to 11/26**	Functionality and user personalization on Attending page, and Single Event page functionality	Created single event page and updated header
9	**11/27 to 12/3**	Debug route and page using Python code was implemented	Linked Events and Attendance pages with Single Event page, and fixed its buttons
10	**12/4 to 12/11**	Show the number of users attending events; major cleanup/refactor	Attendance history added, and logos were added. Bugs on Events page addressed
11+	**12/12 to 12/20**	Application polish, final presentation, and final documentation	

11. Future Revisions

For NYCCal! 2.0, we would like to implement a few additional features.

The first of these features would be email support. We already collect the email addresses of our users when they sign in to the application, and we of course know when our events will occur. This gives us the potential to send reminder emails and signup confirmation emails to users, making sure they keep an eye on the events they are registered for. Additionally, we would be able to send a digest newsletter of any interesting events coming up which are personalized to each user's preferences.

Another feature we would like to add is to enable paying for events directly on NYCCal!. This would allow us the chance to further simplify registration on both ends (the attendee and the organizer) by handling payment and signup at the same time, and it could even provide a path to profitability for the application by us taking a small portion of each transaction.

A final feature we would like to add is a dedicated desktop version. Our target with NYCCal! 1.0 was to create a mobile site, so while it works fine on desktop, it could use further optimizations. Some of these optimizations could include a new horizontal interface on the Events page (with a large calendar on the left side and a scrolling list of events on the right side) and a column view on the Attending page which shows both your previous and future events at the same time.

12. Citations

Iyengar, Sheena S. & Lepper, M. R. (2000). When Choice is Demotivating: Can One Desire Too Much of a Good Thing? *Journal of Personality and Social Psychology*, *79*(6), 995–1006. https://doi.org/10.1037/0022-3514.79.6.995

Park J., & Jang, S. (Shawn). (2013). Confused by too many choices? Choice overload in tourism. *Tourism Management (1982)*, *35*, 1–12. https://doi.org/10.1016/j.tourman.2012.05.004