Bordr Documentation - Group 8

Introduction

People often have many different interests, but no consolidated method of sharing these interests or connecting with people who share similar interests who are in the same location. In order to achieve a way to connect people through common interests, a tool is needed to streamline this process to connect people who want to spend their free time together engaging in activities that they would both enjoy.

Five Rensselaer Polytechnic Institute students took on the challenge of creating a tool to provide students, employees, families, people with internet access an instantaneous and streamlined way to bring people with common interest together to organize a fun activities between like-minded individuals that otherwise may have had no way to efficiently organize themselves.

Project Team and Assignments

A team of five students from Rensselaer Polytechnic Institute - Daniel Brenner, Albert Chang, Adam Kruchkow, Chandler Maskal, and Conrad Mossl - utilized a diverse set of skills and experiences to develop a sustainable Information Technology solution to meet the needs of our stakeholders:

- Chandler Maskal, a Junior Information Technology and Web Science (ITWS) major with a concentration in Cognitive Science, contributed to the front-end and back-end solutions to the application. She has experience working with high level programming languages, database management, and collaborating with a team to ensure cost-effective and reliable solutions for clients.

- Adam Kruchkow, a GSAS major taking an interest in ITWS and web design. He is familiar with game code and user interface, and applies this knowledge to front-end development. He also has some artistic experience.

- Daniel Brenner, a CS major who is interested in ITWS. He has a strong understanding of backend coding and is familiar with many front-end coding techniques. His main strength is implementing and debugging code.

- Conrad Mossl, a junior ITWS major with a concentration in Management Information Systems. With interests spread from music to student government, he is constantly on the lookout to improve current processes, both technical and standard, in order to make the world around him a more efficient and productive

place. While he cannot admit to being an expert in any one technical field, he takes great pride in being able to solve most problems before him with a combination of ingrained lessons, concentration, and old-fashioned hard work.

- Albert Chang, a sophomore ITWS major concentrating in Management Information Systems, collaborated with Adam in designing and implementing the front end. Being the youngest and least experienced member of the team, he quickly learned the ropes of working with senior team members.

Problem Statement

Everyone is familiar with the feeling that their precious weekend is slipping away while they sit around trying to figure out what to do. Bordr is our attempt to make that feeling a thing of the past. By connecting users with similar interests, they can make the most out of their spare time, a precious commodity to most. When people of any kind have spare time they want to fill with an activity, they endure the monotonous process of Facebook messaging, text messaging, physically finding, or making a status to inquire whether other people are currently free, whether other people would like to participate in an activity, and when and where the activity will occur. A large portion of the free time at hand is dedicated to this tedious process of connecting people which can result in cancellation of the event and/or additional stress for the planner.

IT Solution

To address the described problem, a solution was developed to streamline the process of displaying common interests between friends and strangers and creating events to allow people to engage in activities involving these similar interests. Each user will select a set of activities that they enjoy when they create their profile. Then, whenever the user has time to spare, they will log into Bordr and set their profile status to “bored” which indicates that they are online and looking for activities to participate in. After logging in, the user will have the option to view their profile, create an event, view events that they have created that have not expired, and search for existing events. The profile page gives you the option to edit your profile login information, and add interests to your profile which will allow you to more easily connect with events that include your interests and other people who share your interests. The create event page allows you to create events when you’re bored. The assumption behind the creation of these events is that they are instantaneously active with the intention of happening as soon as possible. These events will automatically end after 24 hours from the creation time. The tags that you input with the event are all connected to a database and allow them to be searchable in the event listing page. Each event also has a privacy option for users who prefer to

only engage in activities with other users they've added to their friends list. Once an event is created, you can view or edit this event in the my events page. The my events page also includes chat capabilities that allow you to communicate with event attendees to further plan the event. The landing page also includes a logout function for logging out. Unless the user logs out of the application, their information will be saved for a period of time eliminating the hassle of logging in and out when checking on events. An encrypted cookie is issued to any user who logs in. When the user attempts to access the server, the cookie is sent along with the request. The server decrypts the sent cookie using a "secret" which is held on the server side. This secret is updated automatically every midnight for security. If the server is offline at midnight, it will instead update the next time the server is started. If the cookie is decrypted successfully, then the cookie is valid, and the user is granted permissions as if he/she has already logged in. If not, the user is prompted to log in before they may continue. The cookies are set to expire every 30 minutes without action, but taking an action that re-sends the cookie (i.e. requesting a resource) refreshes the cookie's timer up to 5 minutes.

IT Technologies

A web application solution was implemented to allow for mobility of the application and access across multiple platforms. Bordr is responsive on many devices and single page making it lightweight. Bordr utilizes the AMP stack for both development and deployment. Because of its accessible syntax, legacy/cross browser support, and because it is becoming a general standard, we plan to script our page in HTML5. Our front end is designed using our own CSS combined with Bootstrap library to create an distinct and visually appealing experience for our users. This is especially important when trying to use a similar style and theme on both the desktop and mobile phone interfaces. JavaScript was chose due to its popularity and industry standard. JavaScript allowed for us to have consistency across the entire application with ease of use and high functionality.

In terms of functionality, the bulk of our front end is implemented using Javascript/Node.js. For our back-end, user, event, and interest data is stored in a Mongodb database within three separate collections that cross-reference the data which is accessed through get and post requests. Based on a team member's recommendation, we planned GoToMeetings (an online meeting service) in conjunction with Google Docs for project management and coordination. Our application was intended to be hosted on a Digital Ocean server which was chosen due to the extensive documentation that was found which made the process of setting up the server easier than other options. Finally, we used a GitHub repository to handle source control due to its wide usage and each member's past experience using this method of source control.

IT Requirements

Functional

- Back-end database of user profiles, interests, and activities.
- Secure log-in
  - Encryption of cookies to hold log-in information
- Autofill capabilities for adding interests to alleviate cluttered repeat listings
- UI for querying the internal database
  - Users must be able to find friends to add them to their friends list.
  - Users must be able to find activities they are interested in and add them to their profile.
- Mobile-friendly interface
  - Design will be mobile-first

Non-Functional

- At least ¾ of our test users must agree that the layout and design of the website must meet the following qualities
  - Attractive
  - Simplistic
  - Functional.
- Terms of service for using our web application, that include an agreement to only use our tool to organize legal activities.

Project Plan

It was necessary for the student team to be able to refer back to and modify the original site architecture and scope outline throughout the development phase. In order to do this, the team chose to implement an agile development methodology because it best fit the team's development needs and members were most familiar with this methodology.

The first phase of our project began with the brainstorming phase where the team all worked on identifying an issue that that could be solved by the creation of a web application. Once ideas were presented, the team weighed the pros and cons of each idea and chose an idea. Out of the dozens of ideas discussed, the team eventually agreed on this idea because it struck a chord with every member. Each group member was familiar with the feeling of boredom and the associated frustration when trying to organize something to do. Because of this ubiquity, we felt both a strong motivation to

create the application and a sense of confidence that many people would be interested in using our application. In addition, this project has a clear and central goal to work towards while also allowing for several avenues of expansion. This stage lasted approximately eight days and was followed by the planning stage. During the planning stage the team gathered functional and nonfunctional requirements for the project and decided on which technologies to implement. During this stage the team also assessed every member's strengths and weaknesses in order to effectively and evenly divide the work between the five team members. The team then began the official project proposal and wireframe drawings. After submission of the project proposal, the wireframes were modified as more requirements were added and removed from the original plan. The planning stage lasted approximately 18 days. Once the wireframes were approved and understood by every member, the team began the development stage. The agile methodology was implemented after this stage began as the team moved back and forth between developing the various front and back end components and planning how to approach future development and how to tie every component together, modifying the requirements, and developing further. The development phase lasted approximately 49 days including the agile cycles. Once functionality was present, the team began testing the various components of the site and cycling back to development as bugs were found, or better, cleaner solutions were proposed. The testing phase lasted approximately 21 days with development continuing during this phase. The final phase included hosting the site on the Digital Ocean server, writing the final presentation, and writing the documentation for the project.

Overall, the project plan represented the equal division of time and work between members on the team to ensure goals were met within the given timeframe. The project plan was created and shared with the student team in an effort to maintain visibility and communicate the project scope and deadlines. Similarly, by scheduling weekly meetings and striving to maintain an open line of communication, the team remained well aware of in-class assignments as well as client deliverables to result in the successful completion of the project.

IT Design

The front and back ends of the project were primarily developed in parallel. Our initial design was mobile-first with an emphasis on ease of use. However, as development began, our design slowly changed. Figure 1 shows the original site architecture.

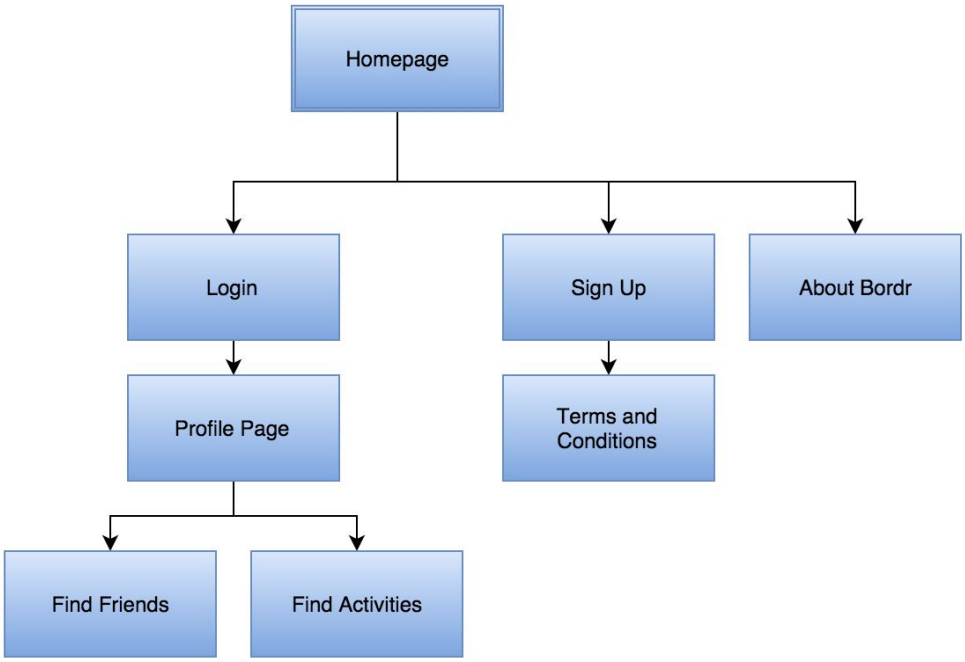

Figure 1

As development began, our site architecture was simplified with the goal of reaching any part of the site within two clicks for ease of use. Figure 2 shows the updated wireframe for Bordr's site architecture.

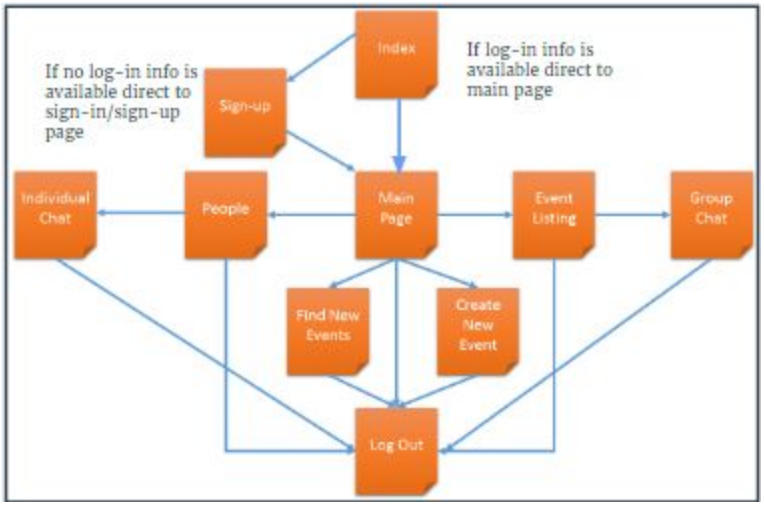

Figure 2

Development began with a disorganized and cluttered page organization. As development continued, time was put into re-organizing all of the html and javascript files for each page into their separate files in order to more easily merge changes on source control. This helped speed up development, clarified task assignments, and streamlined the merging process in source control while keeping the single page structure that we chose. Originally the team intended to make use of the entire MEAN stack due to how well these elements worked together. However, as we all began developing different pages simultaneously, the use of Angular.js proved to be more of a

hinderance than a help. JavaScript was adopted in place of Angluar.js due to the simplicity and consistency of using only JavaScript and very little jQuery. Transitions between each page are applied in transitions.js and the HTML for each page has a corresponding JavaScript file that uses post and get requests to communicate with the database through the server file, requests.js.

Future development of the application would include improvements to the front-end ascetic, addition of notifications to users when events in their area are created, and notifications to users when users in their friends list create new, public or private events. We also discussed the creation of a terms of service agreement to have users agree to only use our tool to organize legal activities.

Conclusions

The results of this project were twofold in that a complete and functional web application was created that reached our goal of effectively organizing events between like-minded people with shared interests and the team effectively learned many lessons regarding project management and working in a team.

Each member of the team was assigned to a role in the project development that best reflected their strengths, however this did not prevent the team members from learning new skills in order to complete the project. Each individual implemented the technologies we were exposed to during class and effectively applied them to a the development of a web application. Members learned how powerful libraries like Bootstrap can be and how using some technologies such as Angular.js, in our case, were more tedious to implement than helpful. Members also learned how to set-up and integrate a chat server into an interface within a web application and learned how to setup a hosting server.

Each member had previous experience in Information Technology and creating solutions using Information Technology, but not every member had experience with implementing a RAD methodology, or closely following a project plan. Throughout the development cycle for this project, the team learned valuable lessons concerning the practice of utilizing an agile development methodology. The team found that implementing this development strategy was ideal for the way the work was split between the teammates. However, when splitting the work between each member, the specific task requirements of each team member were not stated explicitly enough for the project to be developed as effectively as possible. This resulted in members working on the same files and being unaware of what still needed to be worked on and the most effective order in which functionality should be added. The team learned that explicitly stating the requirements for each team member early on and effectively communicating the status of the project throughout the development cycle is necessary for successful development.

Early on, the team encountered an issue with pulling too many commits to GitHub due to installing all the necessary npm packages and pushing all of those to the original

repository, Bordr. This caused GitHub to crash and other team members were not able to sync their local repositories. To fix this issue, the team created a new repository, Bordr_v2, and pushed the code without the packages to this new repository.

Towards the end of the development cycle, after significant functionality was implemented but nowhere near finished, the organization of the code was cluttered which made merging and adding functionality tedious and more time consuming. It was proposed that the code needed to be re-factored and reorganized in order to make everything clearer and make branching easier. The code was then refactored, but to a point where much of the code that was functional was lost and most if not all of the original functionality had to be re-done. Because the code had been reorganized into separate files and also new functions, the original code could not simply be copied over. All styling aspects were also lost during this refactoring. Moving forward with the project in this state was a major set-back. However, the team moved forward despite this setback and added as much functionality as possible within the time constraint. However, the old site before the code refactoring is still running on the server and the link can be found below.

Time management was one of the biggest issues the team encountered and the team did their best to mitigate this risk as it arose. Our original project plan did not accurately represent the amount of time it would take to complete each feature which set the team back during the development cycle. The team mitigated this by attempting to re-factor the task assignments more effectively and opening communication between members more. The team also encountered the problem of scope creep as we moved into the development cycle. The scope of our project was larger than we realized when we outlined the scope and continued to grow uncontrollably as we developed the application. This also set the team's development back and was harmful to the completion of the project. The learning curves associated with GitHub and BugZilla proved to be another time commitment that the team did not originally plan. Lack of expertise when branching, merging, and syncing member's work led to loss of work, inaccurate merging, and delay of development. The team intended to implement the cross-referencing of users attending events as a property of each event and a page where this would be displayed for the creator of the event to see. The team also intended to code a cleaner, more sophisticated front-end interface, allow users to edit their events, and successfully host the application on the Digital Ocean server.

Links:
https://docs.google.com/presentation/d/1gv3qh3kE3LcyDiKdq0wEpvZZSVOKqzaz1T7meaxxr6A/edit?usp=sharing
https://drive.google.com/open?id=0B98AxxpJQb98Ym5OLUNkT212SFU
http://conradmossl.me/Bordr/testing/index.html
https://github.com/GreenSpiny/Bordr
https://github.com/GreenSpiny/Bordr_v2