BSF Portfolio

Alex Krings, Andrew Anselmo, Michael Buffington, and Nathan Parnell
May 5, 2023
ESOF 423

Section One: Program

The source code of the program is located on our github repo: G3-BSF.

Section Two: Teamwork

This project was completed by a team of four developers. Within our group dynamic there ended up being two primary engineers and two engineers. For the most part, team members one and two developed/contributed the most on the project. In order to help share the workload, team members 3 and 4 created the technical documents for the Users and any future developers.

Team Member One:

This team member was responsible for creating and figuring out how to run and use Firebase. This was a massive portion of the project as Firebase included hosting, database selection, and API design. Team member one was also instrumental in the implementation of CRUD operations for the various app features. As well as they implemented the page routing plan and created the initial database schema and original UI design. Overall, they contributed around 27% of the total project.

Team Member Two:

Team member two was another important collaborator who implemented key features of the project. The first being the waiver functionality which was a core requirement for the project. They also created a login page in order to protect the website. And the last big

feature that was implemented was the email API, which sends two emails to users who sign up.

For developers, team member two was also responsible in setting up unit tests for the project to satisfy continuous integration requirements. Team member three contributed around 27% towards the project.

Team Member Three:

Team member three created various screens that used database API calls to display the proper information for that screen. A key feature that member three took on was styling and some UI changes. For the grand scheme of the project, the styling wasn't the most important feature but was huge to show the client that we cared about how the final product would look and feel.

Member three helped with the creation of user documentation and helped out the other members with any tasks they may have had. Team member three contributed close to 22% of the total project.

Team Member Four:

Like member three, team member four created various screens to display information, but their biggest contribution was a screen that allowed for the user to import the bridger codes and then export them through the database. Team member one helped out with the export function.

Team member four also created both the user and the developer technical documents. Another technical document that they created was the bug tracker. While this was a requirement for the project-- it was helpful for any team member to check and fix bugs if they had the time. Lastly this team member contributed around 24%.

Section Three: Design Pattern

A design pattern that was used for this project was the Model-View-Controller (MVC) pattern. What this pattern does is specify that the web app consists of a data model (M),

presentation of the information/data (V), and controlling the information/data (C). The pattern then requires that each of these three things be separated into different objects and mostly relates to the UI/interaction portion of the application.

You can see this in our project in a couple of different areas. Our data model is a firebase NoSql database where we house all of the information. The presentation of the data is all of our various screens located in /react_project/src/screens/. And lastly our controlling of the data is located under /functions/api/. This separation that the MVC naturally creates was great in increasing presentation as we knew where everything was located. For example, we didn't have to search through each .js file looking for our database calls. We knew exactly where they were located.

Section Four: Technical Writing

There are two pieces of technical writing that accompany this portfolio. One document is a user help guide and the other is a programming document that outlines the project and what tools we used or had to install to accomplish it.

User Document:

The user document is located under the help tab on our BSF website as well as you can view the document through the project in /react_project/src/screens/help.js.

Help Page

Important Links:

GitHub Repo

Bug Tracker

Races

The race page is where we create new races as well as get to the volunteer sign in page.

- First: select the race(s) the volunteer would like to sign up for.
- Second: select a date from the top and then press 'LAUNCH CHECKIN'

The launch check in button will bring you to a sign-up page.

- Page One: This page is the contact information page. Here a volunteer needs to fill out all the fields with their correct information. Once filled in, the volunteer will click 'Next' at the bottom right of the page which will bring them to the next step in the sign in process.
- Page Two: The second page is where the volunteer selects the position that they will like to volunteer for. Once selected, go to the next page.
- Page Three: This page contains a link to the waiver they need to fill out in order to become a volunteer. Have the volunteer click the link and then have them fill in the information on this page. Once they submit the waiver, help them navigate back to the BSF website and go to the next page.
- Page Four: The final page contains a submission button. If the volunteer has filled out all the correct information they can click the submit button and they are all signed up!
- There is a back button on the page that can be used to go back in case the user typed something wrong or would like to change anything.

Last thing here is a button to send the voucher codes to the volunteers. Press that button when all volunteers are finished signing up and you're ready to send them.

Create Admin

This page allows for the creation of new accounts. In order to create an account, just fill out the email and password fields and then click create account. You can tell if it works, by signing in! Make sure the password contains one capital letter, one special symbol, and one number.

Import Codes

This page is where you can import the Bridger Ski voucher codes.

- First: Select the import button. Navigate to the file you want to upload (.csv or .xslx) and select the upload button once the right file is selected. If you selected the wrong file, just click the button again and find the correct file.

- Second: Once you check the file, press the Upload file to send it to the database.

Log Out

When you are going to be away from the volunteer page or you are done for the day, press the 'Log Out' button to log out!

Bugs!

If you notice any bugs, reach out to the Brider Ski technical team. Make sure to include a screenshot if you are able, as well as a description of what occurred.

Developer Document:

The developer document is located in the README.md on our GitHub repository.

Bridger Ski Foundation Sign-Up App

This app is designed to help and increase the workflow for registering and signing up volunteers for bridger ski activities and events! Due to this web-app, there is no installation needed for the users. All they have to do is go to the [BSF Signup] website for the latest build.

The goal of our project was to create an app that was useful and easy to navigate.
This is Sprint number five and is our final build for the project. Over the course of the project, we were able to develop all the main features required.

The developers for this project were Michael Buffington, Andrew Anselmo, Nathan Parnell, and Alex Krings.

Bugs

If there are any bugs that you notice during development, please take note of it and add it to the bug backlog. If a user reports a bug, please take note of any relevant information and again add the bug to the backlog.

If you would like to report a bug that you have found, please notify the Bridger Ski technical team and send any and all relevant information. Please attach a screenshot if possible and take note of the time

Getting Started

React, Node.js, and Firebase were the technologies used to create our web app. In order to start developing, you need to first make sure that you have npm installed on your machine (as well as other helpers explained below).

You can run the following command to make sure that you have the most recent version of npm installed on your machine:

npm install -g npm

If you have any issues with installation or npm-- we recommend checking out npm documentation: npm docs.

The last step is to set up the repo on your own machine.

Firebase

This web-app is being hosted on Firebase, and is also what we are using for our database. For our team-- every sprint we would create a new branch from main and develop there. At the end of the sprint we would merge back to main and then deploy to production.

Layout of repo

The two main folders for development are functions/ and react_project/.

Functions

Functions is where we create our firebase APIs for the CRUD operations as well as for login functionality.

Within functions/ there are two additional folders functions/api/ and functions/util/.

- api/ contains all of the firebase calls pertaining to CRUD operations. Most importantly this deals with the email functionality as well as the BSF voucher functionality.
- util/ is also important and contains the firebase functionality for admin login as well as authentication.

React Project

Now here is where the real magic happens.

Before we go too far, you will need to make sure you npm install the following:

- npm install @mui/material @emotion/react @emotion/styled @mui/x-data-grid-pro
- npm install firebase
- npm install axios
- npm install papaparse

firebase and axios are used for various firebase calls, but also used to push to production.

@mui is a React styling API that we used throughout the project. Such as for the stepper when creating a new volunteer or for the table that displays the vouchers.

papaparse is only used to import the bridger ski codes.

The firebase and axios are used for getting the data from firebase, and @mui is used for various styling used in our project.

So within react_project/--we have three folders: /build, /public, and /src.

We don't mess with the build. Firebase uses this folder to build and compile our project when we are finished making changes for the sprint.

Now /src, /src is a fun time.

src

This holds our app.js file as well as (you guessed it) contains more folders.react_project/src/components and react_project/src/screens

- /components -- is where we put our helper components and functions for the various screens.
- /screens -- this holds all of our individual pages (or screens if you would) for the website. This folder also contains our test files. These files run when ever a push is made to the repo to do a check on the build.

Pushing to live

In order to update the live site you need to do the following.

- First: Merge the branch with main and resolve all conflicts.
- Second: Open up a terminal and go to where you have the project.
- Third: Go to the /react_project and run the following: npm run build.
- Fourth: Go to the root of the project and run the following: firebase deploy.

Local development.

First things first. In order to start developing locally-- you'll have to change a line of code located in package.json.

Change "start: xxxx" from "start: react-scripts start" to "start: 3000

Up next, open up a terminal and navigate to /react_project and run the following command.

npm start

This runs the app in the development mode.
Open http://localhost:3000 to view it in your browser.

//Note: if you change the port to something other than 3000, then change the localhost url to the matching port.

This page will now show whatever current progress you have made locally and will also be able to show any bugs that there may be.

npm test

Launches the test runner in the interactive watch mode.
See the section about running tests for more information.

npm run build

Builds the app for production to the `build` folder.
It correctly bundles React in production mode and optimizes the build for the best performance.

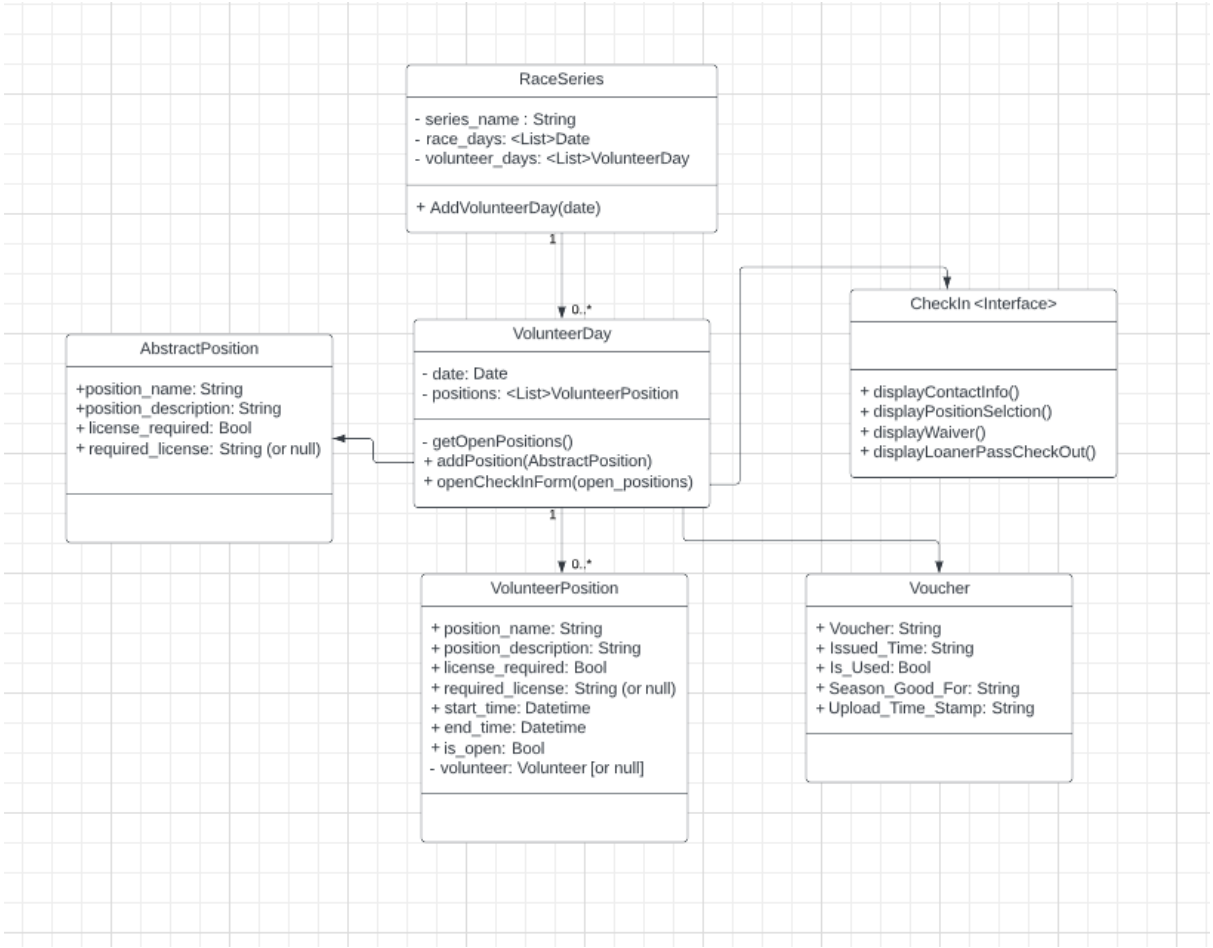
The build is minified and the filenames include the hashes.
Your app is ready to be deployed!

See the section about deployment for more information.

The End

And that's it! If you can think of anything else to add feel free to add to the readme.

Section Five: UML



Section Six: Design Trade Offs

There were many small design trade offs made during this project in order to make sure functionality was both complete and robust. One design trade off that was made during this project (and the most notable one) had to do with the importing and exporting of the voucher codes. The team member responsible for implementing this was having trouble figuring out how to create one atomic transaction for the whole file. Due to time constraints, there are multiple transactions that have to occur which increases the overall

time complexity for this function. Thankfully, the file size will not be big enough to make any notable difference.

Section Seven: Life Cycle Model

The life cycle that was utilized during this project was the Agile Software Development Life Cycle. This life cycle was one of the biggest reasons why this project was a success for a couple of different reasons: two week sprints and standup meetings.

Sprints are one of the key things that makes Agile methodologies successful and it helped our team out tremendously. For starters, it forced us to start working on the project early and not put it off closer to the deadline. It also helped show progress not only to ourselves but to our client, as every two weeks there was some improvement made to the website whether it was big or small.

Standup meetings are another big Agile component that helped our team not only make progress but better communicate. These were short five to ten minute meetings where the team was able to chat and talk about how things were going, whether that be about the current sprint or other life circumstances that could be going on.This allowed us to better get to know one another and increase productivity.

Overall, the Agile life cycle model that was used was the best possible for this project as it made us work on it consistently throughout the semester and complete the project in a timely manner.