Texas A&M Disc Golf Club Website

Team Royal Flush

Andrew Cramer
Alok Chandrawal
Chandler Ochs
Prathiksha Prasad
Snehasish Sen
Vignesh Babu Manjunath Gandudi

Summary

The TAMU Disc Golf website delivers each of the customer's needs and more. Our customer's main need was to be able post a series of information for the members of the club to consume. This information needed to be delineated across several easily accessible pages and modified dynamically in an intuitive way. The website accomplishes all of this.

Some other requests our customer had was for there to be a *Lost and Found* section that could be dynamically updated by members, a login and authentication system, and an image gallery on the landing page. We were able to complete both the dynamic *Lost and Found* section and the authentication system in full, however we were unable to complete the dynamic gallery because we were unable to store photos yet. The primary stakeholder for this project are the club officers, who will be using this website to inform club members.

User Stories Implemented

1. Testing
    a. Front-end testing (3 pts): For this we had to setup our testing practice and learn to use Mocha and Zombie. We were unfamiliar with Mocha and Zombie at the time so we gave it 3 points. It took a very long time.
    b. Continuous Integration/Continuous Development Github Pipeline (1 pts): For this we had to figure out how to test and deploy using the Github Actions service. Because we knew it would be more reading that coding we gave it 1 point.
    c. CICD pipeline with postgres server creation and seeding (1 pt): We were unsure at first what the problem was with our CICD testing section of our pipeline but once we figured it out it was very straight forward. Because we knew that it would only require a small amount of research we gave it 1 point.

2. Express Routing
    a. Basic Express Routing (3 pts): For this we had to set up express routing. Because we were very unfamiliar with express at the point we gave it 3 points.
    b. Advanced Express Routing (3 pts): For this we wanted to programmatically setup express routing using the express router. Again, because we were still very unfamiliar with express we gave it 3 points.
3. Nav Bar and Site footer
    a. Navigation Bar (2 pts): We created a nav bar that would be shown on all documents. We did this before we started using handlebars at the time so we estimated it would take a lot of (time and it did) which is why we gave it 2 points.
    b. Add specified NavBar Buttons (2 pts): this required us changing a lot of tests that we were unfamiliar with at the sime so we gave it 2 points despite it being very simple to do now.
    c. Site Footer always below or on the bottom of the page (2 pts): We had a great deal of trouble with the footer before creating the user story. Our problem was that the footer of the website would be displayed in the middle of the screen. We didn't know what to do so we gave it 2 points. It took a lot longer than that.
4. Login/Authentication
    a. Login Button (0 pts): For this story we had to create a button that would later be used for logging into the site. Because the button didn't have to do anything at this point we gave it 0 points.
    b. Registration Page and form (3pts): This story required us to access the database and update it depending on several conditions. We also implemented the redirect logic and acceptance pathing here. It was a lot of code at a point where we weren't completely comfortable accessing the database yet so we gave it 3 points.
    c. Log In Permissions (3 pts): Similar to the last user story with the added complication of verifying passwords without storing the password in plain text, and keeping the user logged in with passport. It was a similar amount of work so we gave it the same number of points. Because it was going to use a lot of the same db accessing methods we gave it 3 points as well.
    d. Registration, Login and Edit Profile success/error flashing (1 pts): Because the registration, login and edit profile backends were already filled with the needed logic, all this required was flashing a descriptive string and displaying the flashed message on the frontend.

    e. Email form validation (1 pts): This involved writing more tests using regex in the login and registration forms.
    f. Dynamic Nav Bar (2 pts): This involved displaying different compound elements in the nav bar depending on whether or not someone is logged and the role of the user who is logged in. We gave it 2 points because we were unsure of how to approach it, however again it took a lot longer than we expected.
    g. Profile and Edit Profile Page (2 pts): This involved creating forms and corresponding post links (protected by authentication) for the elements of the user profile. We estimated 2 points because it was a lot of the same backend work as registration and login.

5. DB Migration
    a. DB Migration Setup (3 pts): In this story, we had to setup database migrations with sequelize ORM so that the local database is setup for development and any DB level changes are seamlessly applied. This was particularly challenging as we had to figure out how to automate the entire process and deploy it on Heroku. Therefore, we gave it 3 points.
6. Upcoming Events Module
    a. Upcoming Events (3 pts): This story required us to add an upcoming events tab on the landing page. We estimated 3 points as it was the first attempt at adding this section and involved setting up different components.
    b. Upcoming Events dynamic (3 pts): This story involved making the upcoming events section dynamic by adding database connections. We gave this 3 pts as it involved backend coding efforts.
7. Lost and Found Module
    a. Add Lost & Found Item (3 pts): This story involved adding the lost and found section to be able to add lost and found items. We estimated 3 points as it's a completely new module with significant frontend and backend changes for creating a form and saving the contents in the DB.
    b. Refactor lost & found code (2 pts): The task was to refactor lost and found module to use Sequelize ORM for database interactions. We gave it 2 points as it only required a few modifications to use sequelize and the est of the functionality remained the same.
8. Competitive Teams Module
    a. Competitive teams section (2 pts): This story involved adding information about competitive teams in the club. We estimated 2 points as it did not involve much effort on the backend.

   b. Make competitive teams section dynamic (1 pt): The task was to make the module dynamic by adding DB interactions. We assigned 1 point as it was similar to other modules and did not involve significant efforts.
9. Officers Module
   a. Officers section (2 pts): The story involved adding the officers section describing the leads of the club. We estimated 2 points as the changes were straightforward with some frontend changes.
   b. Make officers section dynamic (1 pt): This story required us to make the module dynamic with DB connections. We assigned 1 point as it was similar to other modules and did not involve significant efforts.
10. Local Course Module
    a. ‘Local Courses’ Section (3 pt): The story involved adding the local course module for users to identify nearby disc golf courses. Courses data is fetched and stored in the DB. Details are shown on a map. We assigned 3 points to this story as it involved several challenges in extracting the required data and using the right APIs.
11. Admin Page
    a. Add Admin Page to manage all website data (3 pts): This story required adding an admin page for authorized admins to perform CRUD operations on the data shown on the website. We estimated this 3 points as it involved understanding sequelize, it’s query operations, HTML structure, and replicating similar operations for all the modules.
12. About Us Module
    a. About Us (1 pt): The task was to include details about the club. We assigned 1 point since it content was mostly static
13. Social Media
    a. Social Media Links (1 pt): The story required us to add links to Facebook, twitter, and instagram accounts of the club. We have added placeholders currently as the pages don’t exist yet. We assigned 1 point as we just added placeholders.
14. Gallery
    a. Image Scroller Gallery (1 pt): This story involved adding an image carousel for gallery on the landing page. We assigned 1 point as it was a straightforward task of making some HTML changes.
15. Project Setup
    a. Tech Stack (3 pts): This story involved researching and finalizing the tech stack for the product. We explored several available options and zeroed in on Javascript based frontend and backend. We assigned 3 points to this story as it was research intensive.

b. Nodejs project initialized (3 pts): For this story we had to create a NodeJS project as a starter for the code. We assigned 3 points since we had to set things up from scratch and it involved significant efforts.
c. Create backend server (3 pts): The task here was to setup express server for backend operations of the project. We estimated it with 3 points since we had to explore and setup the server in a modular way so that all the modules can efficiently use the services.
d. Heroku hosting (3 pts): For this story, we had to figure out how to automatically host the application on Heroku. We setup Github in such a way that any new change in master automatically deploys the master branch on heroku. Therefore, we gave it 3 points.

16. News and Recent Events
    a. News/Events Section (3 pts): The main task hers it to build the news and recent events module with a calendar to display events. We estimated 3 points since it involved more work on the frontend to render a calendar and populate the right data on click of the day on the calendar..

Team Roles

Scrum Master: Prathiksha Prasad
Product Owner: Chandler Ochs

Iteration Summary

In all the iterations, the minimum story point was 1 and maximum was 3.

1. Iteration 0. In this iteration, we primarily focused on talking to the customer and getting to know about the project. We built lo-fi UI mockups and a few initial user stories.
2. Iteration 1. This was the first development iteration. We mostly worked on setting up our project and building static webpages as placeholders for different modules. We setup the navbar, testing framework, and CICD pipeline. We achieved 26 points in total.
3. Iteration 2. In this iteration, we did the groundwork of setting up the backend server to make all the modules dynamic. In addition, we enabled a few more modules in the site. We achieved 13 points in total.
4. Iteration 3. In this iteration, we made all the modules implemented in the previous iterations dynamic. We setup database connections and added queries to fetch data dynamically. Additionally, we started working on DB migrations to handle database changes seamlessly and setup local development database. We also started working on login and permissions. We achieved 20 points in total.
5. Iteration 4. In this iteration, we completed the elements that were in progress in Iteration 3. DB migrations, login, and registration modules were completed. Local

course and admin pages were added. Lost & Found and other modules were refactored to handle migrations. Minor cosmetic issues were also fixed. We achieved 30 points in total.

Customer Meetings

All the meetings with the customer were held on Zoom one day before the iteration report due. All the features implemented in that iteration were shown to the customer and feedback was collected and incorporated in the next iteration.

TDD/BDD Process

For our BDD/TDD process we used Mocha.js, Zombie.js and Github actions. Mocha and Zombie work very well together to create the full range of test types in a simple readable way. We used Github actions for the CI/CD implementation in our project. The testing section of our CI/CD pipeline was very useful to use as we never broke an existing feature in a release to Heroku because all tests were run every commit and displayed clearly on pull requests. The only problem we encountered was running these tests when our application became dynamic, but we were able to fix this quickly.

Configuration Management

We used Git extensively to manage our codebase and version control our commits. We maintained the “master” branch as the production branch. For every feature, the feature owner forked a new branch from master and raised a Pull Request (PR) for other team members to review. We have only merged PRs after a comprehensive review by a team member. We have treated every iteration as a release and tagged the appropriate commits accordingly on Git.

We needed one spike to handle the Local Courses Page. As it was a big feature we broke it down to smaller subtasks and one spike. In one of the iterations, we did all the background work for the feature and came up with a scalable architecture to implement the feature. This was a first hand experience in managing and breaking big tasks into smaller user stories.

Issues with Production Release to Heroku

As part of CI/CD we decided to deploy our project using Github Actions every pull request of a feature branch to the master branch. This became an issue when our application made the switch to being dynamic. Many of our big changes to the website involved sequelize database migrations. This was only a problem because sequelize uses a config.json file to access the local database which has different configurations to ones we use to develop. We were able to get around this by creating a python script to populate this config.json file and running this python script before restarting the node server on heroku every deployment. This was done by modifying the procfile.

Issues with Other Tools

Another issue we had was also related to CI/CD and making our application dynamic. The problem was that our application pulls information from a locally hosted postgres server, which was not being hosted in the testing environment used by Github Actions. This was solved by rewriting the action to start a postgres server with the expected access configuration, and adding sequelize migrations and seeding to the testing script.

Tools Used

1. Zombie.js - Zombie is a headless browser that works well with Mocha.js. We found it to be simple yet powerful, making it ideal for our acceptance tests. Browser actions can be completed in very few steps, and both element and raw HTML existence search is supported. We found it to be a very important part of our testing setup.
2. Mocha.js - Mocha is a test framework for javascript. We found it to be simple and intuitive to use while maintaining decent readability. It was also particularly useful for backend development, as it automatically alerts you to slow tests.
3. Bcrypt - Bcrypt is a library we used to encrypt passwords, as well as compare submitted passwords to encrypted passwords. This allowed us to never store raw password strings on any level of our application.
4. Node.js and npm - Node.js is an open-source, cross-platform, back-end JS runtime environment that executes JS code outside web browsers. Node.js has allowed us to run JS code for server-side scripting to produce dynamic web page content. Consequently, node.js validated and verified our code so that we could send it for deployment on heroku platform. Npm is package manager for the Node JavaScript platform. Npm manages our project's module dependency by putting all modules in one place so that Node can find them easily.
5. Express.js - Express.js, or simply Express, is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.
6. Handlebars.js - Handlebars is a templating engine which adds programming constructs to a HTML page. It is minimal, powerful and blends well with express.
7. Postgres - Postgres in an open-source RDBMS. It manages concurrency through MultiVersion Concurrency Control(MVCC) that ensures the database maintains ACID principles. It is our back-end database which runs as a service in

our heroku web application deployment. It has database tables that store users, officers, events and teams information that populates on the dynamic web pages on the website.

8. Sequelize - Sequelize is a promise-based Node.js ORM for Postgres, MySQL, MariaDB, SQLite and Microsoft SQL Server. It features solid transaction support, relations, eager and lazy loading, read replication and more.

Github Repo

https://github.com/acramer/tamu-disc-golf

Pivotal Tracker

https://www.pivotaltracker.com/n/projects/2467338

Heroku Deployment

http://tamudiscgolf.herokuapp.com

Poster and Demo Video

https://vimeo.com/485783151