Project Plan

Zoomologists
Alan Xiao, Daniel Gude, Tyler Kiesman

Overview

The Seneca Park Zoo Dashboard collects business metrics for various categories such as donations, memberships, and merchandise. The data is spread out across platforms such as Blackbaud's Altru, Financial Edge, Facebook analytics, Google Analytics, and Microsoft Excel, which can be challenging to access all at once. The dashboard can provide visualizations for metrics in categories like volunteer programs, special events, and merchandise over a period of time. The leadership team can set up alarm triggers based on trends in the metrics. These alarm triggers can have notifications appear in the dashboard and send emails to the appropriate staff when they are triggered.

The Seneca Park Zoo Society is the main customer for the project. They manage functions at the Monroe County's Seneca Park Zoo such as the front gate, ZooShop, conservation initiatives, marketing, etc. It can be difficult to manage these various business functions and track the assortment of data associated with their operations.

For our project we are going to follow a Scrum approach with our own modifications to better fit our project. The beginning of the project will be spent on getting acquainted with the sponsors, creating initial documents, and understanding the project. Once that is done, biweekly sprints will deliver functionality to the sponsors, culminating in a sprint demo. In each sprint, an additional data platform will be integrated into the dashboard. Early in the summer semester, features will stop being added, and resources will be dedicated to fixing bugs and polishing the platform, in addition to documenting and deploying the project on their work environment. Later in the summer semester, development will halt, and the rest of the effort will be spent on documentation and deployment. Throughout the summer semester, a project poster will be created to be displayed at the end of the semester.

Alan Xiao will be the technical lead for the front end, Daniel Gude will be the technical lead for configuration management, and Tyler Kiesman will be the technical lead for the back end and testing.

Goals and Scope

The main goal is to create a dashboard that displays information on various business metrics. One major feature that is out of scope is for this application to enact business decisions. This application is mostly presentational. Almost all functions will involve grouping, arranging, and comparing available data. Business metrics outlined in the project description will be calculated using the given data. Values used to calculate business metrics will be able to be modified. This includes things such as the yearly budget. The specific data and metrics available will be determined and prioritized incrementally throughout the duration of the project. For presented data, a feature in scope is to show a graphical representation of the information available over a period of time in monthly increments. Users would have permission to only view certain sets of visualized data. Alarm triggers can be created and triggered based on metrics as well as trends in the metrics. The first alarm triggers that will be developed will be via email. They can also be customized to send emails to specific users as well as show a notification on the dashboard when they are triggered.

Deliverables

1. Mockups of UI screens and interactions
    a. design mockups of the main screens
    b. text descriptions for what actions can be made on each screen
2. Basic front end UI with no data visualization
    a. underlying website structure and navigation
    b. placeholders for data representation aspects
3. Back end for authentication
4. Front end with authentication
5. Alarm trigger functionality with emails and UI notifications
6. Visualization for a data category and alarm triggers for the data category (multiple deliverables of this type)
7. Deployable version of dashboard installed, tested, and hosted at the zoo

Risk Management

This project involves interactions with a multitude of external data sources such as Blackbaud's Altru and Google Analytics. One major technical risk is that one or more of these sources could change the format of the reported data or the specification on how to consume it. A change like this could be detrimental to the development and usability of the application since the data provided by those platforms are the integral to the usefulness of the dashboard. Possible mitigation strategies are to separate functionalities of the presentation and the collation of the data. This separation isolates the amount of work which would need to be updated to the new provider's specifications. Another mitigation would be to give special focus towards these interactions and do research into the providers for any design guidelines they advise.

A potentially devastating technical risk is that the application is not secure and it is open for attackers to exploit the site. There could be holes in the code or deployment that allow for malicious attackers to violate the integrity, confidentiality, or availability of the dashboard. While the data that is being worked with is not sensitive, it is important to make sure hackers conno taccess any of it. To prevent any sensitive information from being leaked, we should try to limit the application's access to this data. When developing and deploying, we should be cautious of attack surfaces that hackers could exploit. The team will think of abuse and misuse cases that could arise and make sure that we're cleansing all sorts of inputs. If a hacker does get into the site, the maintainer of the site can hit a kill switch to turn off the application. This allows for developers to look into the issue and patch the exploit before any more damage can be done.

Developing in the time of a pandemic means that one's health and ability to work can change quickly and dramatically. A risk involved with that is for any stakeholder may take time away from the project without notice. A mitigation for this is incorporated into our methodology. Using short sprint timespans, only the currently occurring sprint's plan would be affected, and the following sprint's plan can be modified accordingly.

Deploying to the zoo may pose some risk with website deployment being finicky. ReactJS builds websites by transpiling React code into plain Javascript and then bundling it into a folder, which is done automatically though a library called Webpack. If there are issues deploying the built version of the dashboard to the zoo, a lot of time could be spent correcting it. This can be mitigated by deploying the built version of the dashboard to the zoo's work environment during the sprint demo, allowing deployment issues to be caught quickly and for the offending code or behavior to be fixed.

Scheduling and estimates

The first 4 weeks will be used to decide on a methodology, get introduced to our sponsor's storage platforms, set up our project website, and decide on the scope of the project. This set up period makes up Sprint 0. The next few months will be spent on development and documentation. All development following sprint 0 will be in 2 week long sprints with a customer deployment at the end of each. In the sprints where RIT requires a deliverable there may be less features implemented compared to other sprints. For sprint 1, the objective will be to spend time on making the UI elements of the dashboard and finding appropriate tools for data visualization. On sprint 2, the team will work on developing authentication. For sprint 3 and onwards, the team will try implementing visualization and alert triggers for each data category at a rate of 1 category every sprint. In week 7 the team will spend additional time working on the peer evaluation in addition to working on the project. In week 11-13, in addition to working on the dashboard, the team will be working on the interim presentation and then present it on week 14. The team will end the spring semester by submitting a summary of the interim reflection meeting and delivering an evolving product.

In week 5 of the summer semester, the team will have a feature freeze. After the feature freeze, the team will only work on bug fixes and polishing. The team will then have a code freeze in the 8th week of the summer semester. Once the code freeze is in effect, the team will spend the rest of the time on finalizing developer and user documentation as well as deployment. During this time, the team will work on the posters and presentations for the rest of the semester. The team will end the project by filling out the team reflection and delivering a deployed and documented product to the sponsor.

For estimation, our main strategy will be to use the T-shirt sizing method. Each user story will be assigned a size of extra small, small, medium, or large by the team. Since we will be working in small time increments, using relative sizes will be easy and efficient since our sprint backlog will only contain a low number of user stories. We will use a scrum board to track our user stories, and this would allow us to choose our schedule every sprint. If we need to make changes to the sprint, we will contact the sponsors to ask if they are willing to accept the change in the stories for the sprint and if any new stories should be added in exchange, and if they are okay with the changes, move any removed stories to the product backlog and add the newly added stories to the sprint backlog.

Measurements & Metrics

One of our most important metrics will be our burndown chart. This will be used to determine our progress within a sprint and whether we are on track to finish the sprint scope or not. We will also keep track of sprint velocity as well as it helps us estimate future sprints. Our user story measure will be done through T-shirt sizes which was discussed earlier in our scheduling and estimating section. In order to track how much team members have been working, we will all keep separate logs of what we've done and our total time doing that work. This will be used to ensure that team members are spending adequate time working on the project. We will also measure the actual time spent on tasks, and use this to calculate how accurate our original size estimations were for refining future estimates. In order to measure the usability of the UI, the team will conduct usability tests to collect quantitative metrics for usability. In the tests, Zoo employees will complete tasks and the time it takes to complete the task and their feedback and feedback will be recorded. This information will serve to tell how usable the UI is.

Technical Process

For our project we are going to follow a Scrum approach with our own modifications to better fit our project. Our reason for using Scrum is that we will have an active product sponsor who we will be meeting with every week for feedback and show deliverables to. We do have a broad overview of features, but we will need to be constantly meeting to get into more detail. Because of this, we will want to take a more iterative approach to this project. We will have a backlog of tasks which will have points assigned to them with the level of effort. This level of effort will be determined using the T-shirt sizing method. Sprints will be two weeks long and at the end of each sprint there will be a demo shown to the project sponsor and the project will be deployed to the sponsor. We will also have a burndown chart as a metric of project tracking and progress. Our initial sprints will heavily focus on exploring the data and what different options we can take. After that we will focus more on the implementation of features.

Our project will be stored in a private GitHub repository. This needs to be private because it is not an open source project and is meant to only be viewed by employees of the zoo. The team will utilize feature branches when implementing new features. A pull request reviewed by team members is required when merging new features into the main branch.

Testing will be done in two ways, at the code level and requirements level. Our plan is to develop the code first and then write unit tests after that to make sure the code is sound. Test-driven development does not work in our case because we won't truly know the details of the features we will be developing ahead of time. Once we have written and tested our features, we will test our features' design with the project sponsor. This will involve usability tests and meeting with the sponsor to check that the feature meets their expectations.