**Table of Contents**

[1. INTRODUCTION 2](#_Toc387410368)

[1.1 Purpose 3](#_Toc387410369)

[1.2 Scope 3](#_Toc387410370)

[1.3 Definitions, Acronyms and Abbreviations 4](#_Toc387410371)

[1.4 References 8](#_Toc387410372)

[1.5 Overview of Contents of Document 9](#_Toc387410373)

[2. GENERAL DESCRIPTION 10](#_Toc387410374)

[2.1 Product Perspective 10](#_Toc387410375)

[2.2 Use Case Models of the System’s Features 10](#_Toc387410376)

[2.3 Interaction Design Specifications for Each Feature 18](#_Toc387410377)

[2.4 User Characteristics 21](#_Toc387410378)

[2.5 General Constraints 22](#_Toc387410379)

[2.6 Assumptions and Dependencies 23](#_Toc387410380)

[3. SPECIFIC REQUIREMENTS 24](#_Toc387410381)

[3.1 Use Case Specifications 24](#_Toc387410382)

[3.2 INFORMATION MODEL 78](#_Toc387410383)

[3.3 Performance Requirements 80](#_Toc387410384)

[3.4 Design Constraints 80](#_Toc387410385)

[3.5 Quality Attributes 81](#_Toc387410386)

[4. APPROVALS **Error! Bookmark not defined.**](#_Toc387410387)

[APPENDIX A - Data Dictionary 84](#_Toc387410388)

[Elements 84](#_Toc387410389)

[Tables 96](#_Toc387410390)

1. INTRODUCTION

This is the Software Requirements Specification (SRS) Document for the GotTrackDays.com project sponsored by the proprietor of Got Track Days Inc., Juan Calderon.

This project is being undertaken by the jBehaving development team. The team is comprised of undergraduate students majoring in Computer Science at California State University, Sacramento. The team members are enrolled in a two-semester senior project course required of all undergraduate majors. Successful delivery of the desired software product will fulfill the senior project requirement for the student team members.

**Project Sponsor:**

Juan Calderon

Proprietor of Got Track Days, Inc.

Email: juan@GotTrackDays.com

Phone Number: (916) 501-7147

**JBEHAVING DEVELOPMENT TEAM:**

|  |  |  |
| --- | --- | --- |
| **NAME** | **EMAIL** | **PHONE NUMBER** |
| Ashley Finger | phyllangela@gmail.com | (408) 623-0062 |
| Bai Xiong | bai.xiong91@gmail.com | (916) 583-4058 |
| Cody Lanier | cody.lanier9@gmail.com | (916) 842-7869 |
| Cody Prior | hiimcodas@gmail.com | (916) 740-0594 |
| Daniel Gallegos | daniel.r.gallegos@gmail.com | (707) 803-1519 |
| Michel Watson | michelwwatson@gmail.com | (916) 596-8253 |

**Table 1 – jBehaving Contact Information**

1.1. Purpose

The purpose of the Software Requirements Specification (SRS) Document is to establish and define the requirements of the GotTrackDays.com website. This includes both the functional requirements (Website functionalities) and non-functional requirements. The document also serves as a guideline for designing and developing the website in a future stage.

This document describes how customers and employees interact with the website, how both the site and the database are designed to accommodate these interactions, and detailed requirements for each of the main functions.

1.2. Scope

The scope of this SRS includes describing the requirements of the GotTrackDays.com software project, presenting the artifacts created during the software requirements engineering phase, and should serve as a contract for agreement to the software requirements described herein.

1.3. Definitions, Acronyms and Abbreviations

The following are lists of definitions, acronyms and abbreviations that are used throughout this document.

1.3.1. Definitions

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| .csv File | A file that stores numbers and text type data. |
| Anonymize | The process of removing personal information related to an account from the database. |
| Deactivate | Flagging a member account in the database so that the member may not log in. |
| Employee | Employees include more than just the Standard Employees. It represents all members of the Got Track Days, Inc. organization, except for those classified as Owners. |
| Event | A track day. |
| Facebook | A popular social networking site. |
| Forum Log | Internal log to be updated by employees when they promote events on a specific car forum website. This will help to make sure that events are being much more evenly promoted throughout the many different forums. Forum Log is currently a working title. |
| Gigabyte | A unit of measurement for quantifying the size of digital data. Commonly abbreviated as ‘GB’ or ‘gb’. |
| Google+ | Pronounced: Google Plus; it is a popular social networking site. [5] |
| Google Drive | Private, public, and/or shared web-based, cloud-storage location used to create and edit documents, spreadsheets, PowerPoint presentations, and more. A Google account is required. [6] |
| Google Wallet | An online payment system utilizing debit cards, credit cards, loyalty cards and gift cards. [8] |
| GotTrackDays.com | The website that Got Track Days Inc. will use to allow customers to register for track time. [2] |
| Guest | A system user who either does not have a registered account or is not currently logged in to the system. |
| Home Page | The initial webpage displayed to user upon arriving at the website. |
| Internal | This classifies the user being talked about as being employed by Got Track Days Inc. |
| Manage | This term is generally used throughout the Use Case section of this document to reference the acts of creating, viewing, updating, and deleting or the various combinations and variations of those actions. |
| Manifest | A list of persons who have registered and are planning on attending a specific event. |
| Member | A user of the website who is currently logged into the system with a previously registered standard account. |
| Microsoft Excel | Spreadsheet software. |
| Need | A condition that must be met in order for the software to be considered complete. |
| Participate | This term is used in the Use Case section of this document to not only reference a physical participation, but also a digital interaction with events. It includes sharing events via social media, searching and browsing events, requesting a refund for events, and also registering and paying for events. |
| PayPal | An online payment website that sends and receives money for personal or business purposes via debit cards, credit cards, and also bank accounts. |
| Postcondition | A condition that must be fulfilled after the Use Case being described happens. |
| Precondition | A condition that must be fulfilled before the Use Case being described can happen. |
| Priority | Need – Required for the software to be considered complete.  Want – May or may not be completed depending on time constraints. |
| Qualifying Account | When signing on the website to do a specific task the user must be signed in on an account that has the correct permission levels for the task the user wishes to perform. |
| Related Use Case | All included and excluded use cases that relate to the one being discussed. |
| Share Events | Promoting an event via one or many social networks. These social networks include Facebook, Twitter, and Google+. [5] |
| Stakeholder | An entity that can be affected by the results of the business Got Track Days Inc. |
| Terabyte | A unit of measurement for quantifying the size of digital data. Commonly abbreviated as ‘TB’ or ‘tb’. |
| Track Day | A day at the track. This is the term used for the all day events that GotTrackDays.com will be selling tickets for. |
| Track Day Calculator | A tool used on the website, GotTrackDays.com, where the user can compare the cost of an event hosted by Got Track Days Inc. vs. a similar competitor’s event. |
| Virtual Garage | A personal virtual database for information that pertains to vehicles that the user owns and are available to sign up for events with. Currently, Virtual Garage is a working title. |
| Want | A requirement that is desired, but may or may not be completed depending on time constraints of the project and class. |

**Table 1.3.1 - Definitions**

1.3.2. Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| CSV | Comma Separated Values or Character Separated Values |
| SRS | Software Requirements Specification |
| UC | Use Case |
| USD | United States Dollar |

**Table 1.3.1 - Acronyms**

1.3.3. Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Definition** |
| Etc. | Etcetera |
| Inc. | Incorporated |

**Table 1.3.1 - Abbreviations**

1.4 References

The following documents and books were used in the preparation of this document.

1. Juan Calderon. "www.GotTrackDays.com." *GotTrackDays.* n.p., 2/26/2014. Web. 2/26/2014. <http://www.GotTrackDays.com/>.
2. Juan Calderon. "GotTrackDays, Inc – www.GotTrackDays.com." *Business Plan*. (2014): n.pag. Document.
3. R. Buckley. "Guide to Preparing the PROJECT CHARTER DOCUMENT." (1.30.2014): n.pag. Document.
4. "GitHub." GitHub Inc., n.d. Web. 2/28/2014. <https://github.com/>.
5. "Google+." Google, n.d. Web. 2/28/2014. <www.google.com/+/>.
6. "Google Drive." Google, n.d. Web. 2/28/2014. <http://www.google.com/drive>.
7. "Google Hangouts." Google, n.d. Web. 2/28/2014. <www.google.com/hangouts/>.
8. "Google Wallet." Google, n.d. Web. 2/28/2014. <www.google.com/wallet/>.

1.5 Overview of Contents of Document

The following describes in general detail the contents of the remaining sections of this document.

**2. General Description**

This section contains information that facilitates the understanding of specific requirements which are to be found in Section Three. The information provided describes the users and the features that the product is to provide, any constraints and all assumptions or dependencies that might affect the requirements specified in this document.

3. Specific Requirements

This section will lay out in detail the requirements of the software. It contains all the information necessary for the design of the system.

**4. Approvals**

All project stakeholders will sign this section. Signatures will serve as proof of having read and agreed to the contents of this document as being accurate and complete to the best of the signer’s knowledge.

**Appendix A - Data Dictionary**

The Data Dictionary describes the structure and contents of the data tables to be used by the system.

2. GENERAL DESCRIPTION

This section contains information that facilitates the understanding of specific requirements. The information provided describes the users and the features that the product is to provide. In addition, this section will include any constraints that will limit the team’s options in designing the software and all assumptions or dependencies that might affect the requirements as specified in this document

2.1 Product Perspective

This software is independent and totally self-contained. It will contain a database that the website can access to retrieve and store records in, but will not link to a database that the Got Track Days Inc. currently has in place. The sponsor, Juan Calderon, was previously using a host server on his own. However, that will be changed to a different server for the purposes of this project.

2.2 Use Case Models of the System’s Features

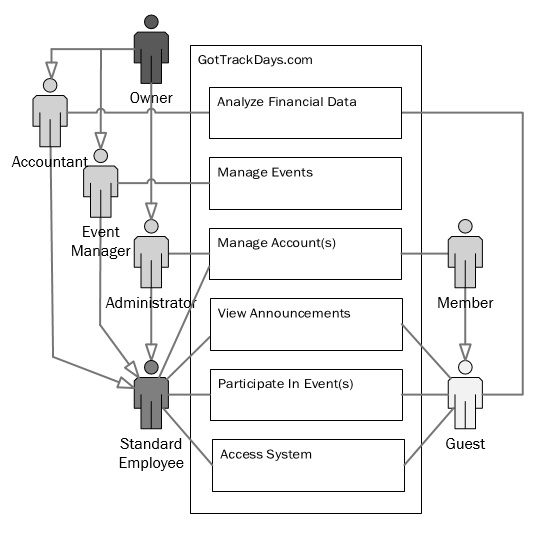
This subsection describes detailed explanations of different actions that can be performed within the system and the different actors that perform those actions. These actions are called use cases and are broken down into a hierarchical structure.

Actors:

* Guest - this is a user that does not have a registered account with the system, but with access to announcements and searching available events.
* Member - this user has a registered account with the system and can register for events.
* Standard Employee - this user has an internal account with the system with access to manage the website announcements.
* Event Manager - this user has an internal account with the system with access to manage events.
* Administrator - this user has an internal account with the system with access to manage accounts and other users.
* Accountant - this user has an internal account with the system with access to managing finances such as refunds.
* Owner - this user has an internal account with the system that allows the user to have access to the entire system.

**2.2.1 GotTrackDays.com**

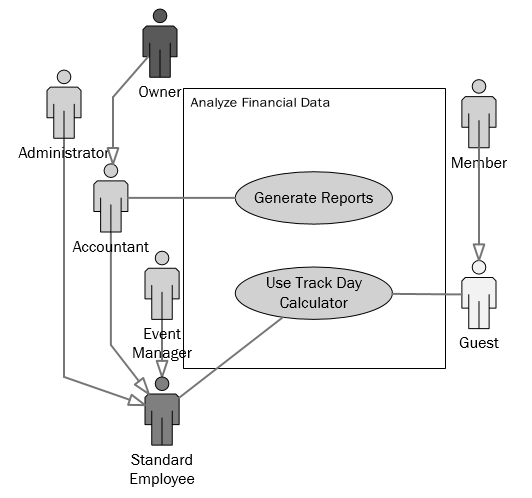
This is a high level view of the overall system and how the actors generally interact with that system.



**Figure 2.2.1 - GotTrackDays.com**

**2.2.2 Analyze Financial Data**

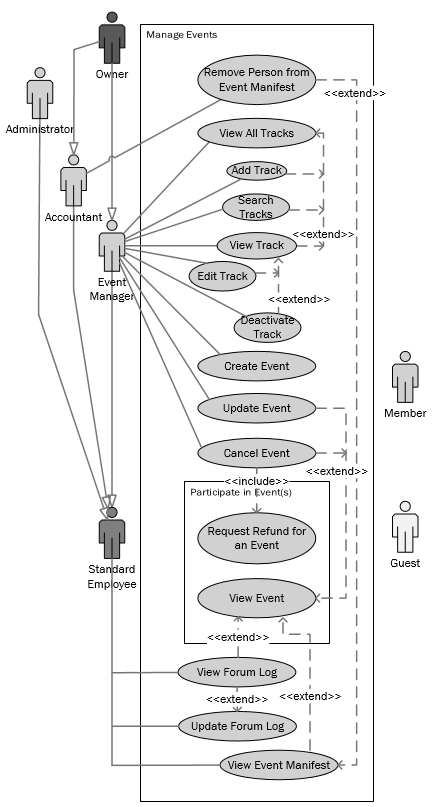
Anything system related that has to do with finances of some sort is visualized here. Every user can view the Track Day Calculator, but only Accountants and the Owner may generate event financial reports.



**Figure 2.2.2 - Analyze Financial Data**

**2.2.3 Manage Events**

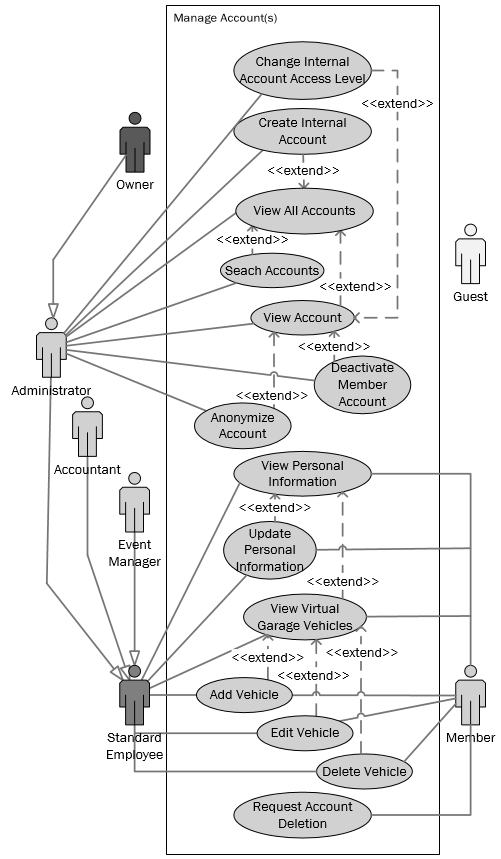
This describes internal functions that have to do with managing the various track day events. The Event Manager role is most prominent in these use cases.



**Figure 2.2.3 - Manage Events**

**2.2.4 Manage Accounts**

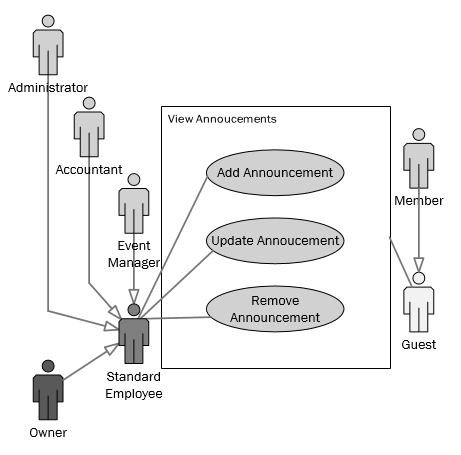
This describes functions that deal specifically with managing and maintaining accounts. Most internal management actions are performed by Administrators, while all Non-Guest users are able to edit their own personal information and Virtual Garage vehicles.



**Figure 2.2.4 - Manage Accounts**

**2.2.5 View Announcements**

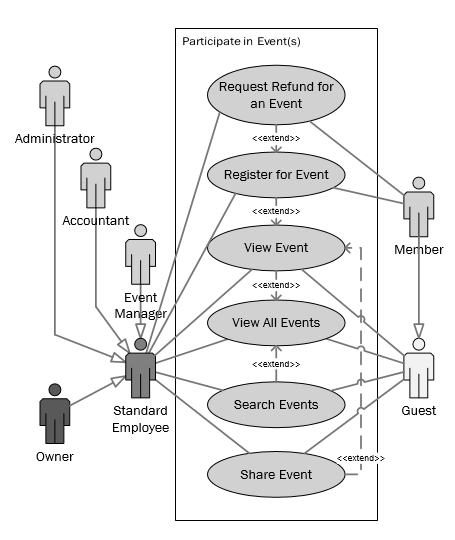
Every user can view the home page announcement section and any internal user can modify the section, when necessary.



**Figure 2.2.5 - View Announcements**

**2.2.6 Participate in Event(s)**

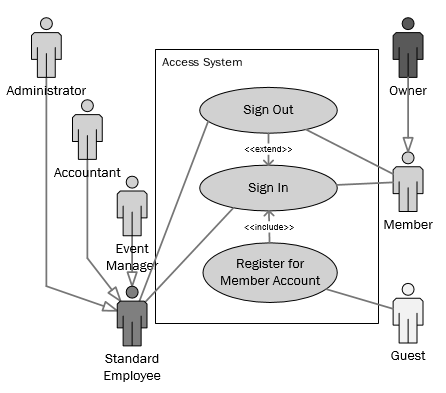
Both members and employees can use the website for what it is intended for, which is registering for track day events. Guests are also allowed to digitally interact with the events, but not register to attend.



**Figure 2.2.6 - Participate in Event(s)**

**2.2.7 Access System**

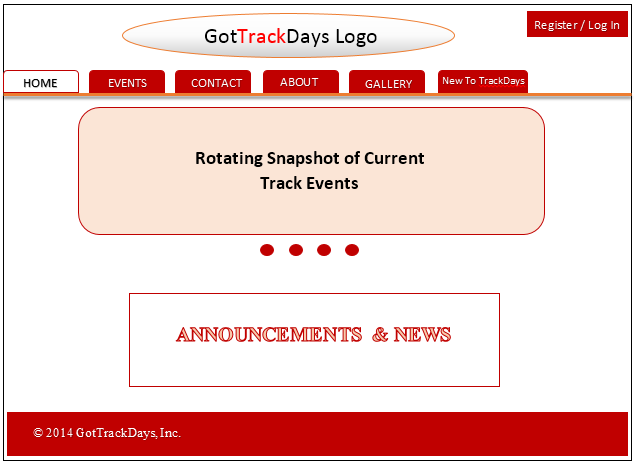
These use cases show functions related to signing in and signing out of the system.



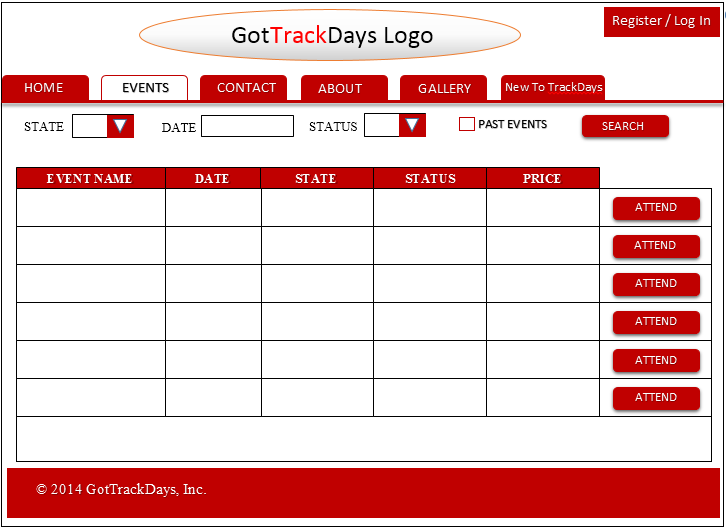
**Figure 2.2.7 - Access System**

2.3 Interaction Design Specifications for Each Feature

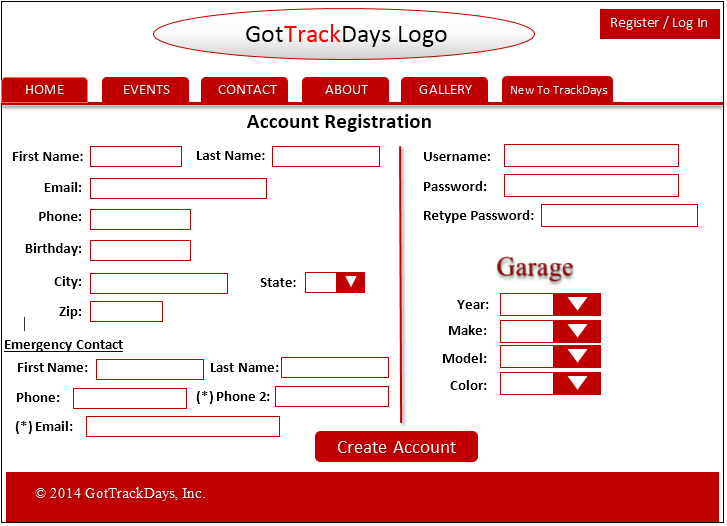
This subsection includes the set of wireframe blueprints for some of the main features of the GotTrackDays.com website. These blueprints represent the logical representation of the uses of each feature by the actor/user. The wireframes provides a visual guide that will show what the website will look like when the user is performing that specific use case scenario.



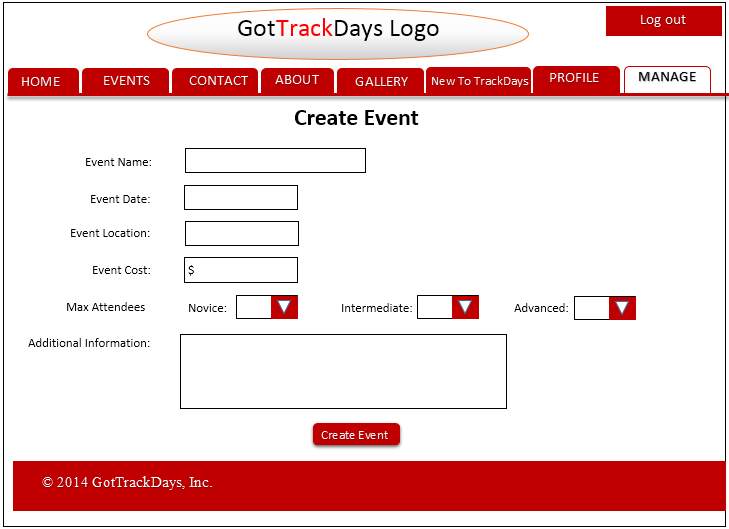
**Wireframe 2.3.1 - Home page**



**Wireframe 2.3.2 - Search Events Page**



**Wireframe 2.3.3 - Member Account Registration Page**



**Wireframe 2.3.4 - Create Event Page**

2.4 User Characteristics

This subsection contains a description of the work that users will expect to be able to do in using the software depending on their user role. The user role and the available features are defined below:

Guest

* Register for user account
* Search, view, and share events
* Use the Track Day Calculator

Member

* Sign in and sign out
* Search, view, and share events
* Use Track Day Calculator
* Register and pay for events
* Request an event registration cancellation and refund
* View and update personal information
* Add, update, view, delete personal vehicles in the Virtual Garage
* Request account deletion

Standard Employee

* Can perform all Member functions
* View and update the Forum Log
* View and print event manifests

Event Manager

* Can perform all Standard Employee functions
* Create, update, and remove events and tracks
* View and remove person(s) from an event manifest

Administrator

* Can perform All Standard Employee functions
* Create new internal accounts
* Anonymize and delete accounts
* Search and view accounts
* Update account permissions and roles

Accountant

* Can perform all Standard Employee functions
* Remove person(s) from an event manifest
* Generate financial reports

Owner

* Can perform all Event Manager, Administrator, Accountant, and Standard Employee functions

2.5 General Constraints

This subsection contains any constraints that may limit jBehaving’s options in designing and implementing the software. Such constraints could include:

* Interfacing with other applications
  + Social Media
    - Facebook
    - Twitter
    - Google+ [5]
* Payment services
  + PayPal
  + Google Wallet [8]
* GUI requirements/Graphic Designers
  + The team may be forced to wait until external contractor’s complete work for the project sponsor before moving forward with specific parts of the project.
* Time and scope of the project
  + The scope may be too large when compared to jBehaving’s overall available time.
* Security
  + Password encryption
  + Each user has a role attribute defining what features they have access to.

2.6 Assumptions and Dependencies

This subsection identifies all assumptions and dependencies that the development team expects to affect the requirements as stated in this document.

1. Team jBehaving is a group of students and the project is related to a course requirement at CSUS. As such, the timeframe of the project is limited to the spring 2014 and fall 2014 academic semesters.
2. As this project is part of coursework for a two semester course, there are expectations on the amount of weekly hours provided by a single student, and these are significantly less than would be expected from a full-time employee.
3. The sponsor understands that the scope of any deliverables will be affected primarily by points 1 and 2.
4. The sponsor’s input is necessary to the development process. The sponsor understands that the sponsor is expected to provide time for meetings, technical reviews, testing, and approvals as necessary.
5. Team jBehaving understands that the sponsor is not able to devote themselves full time to the development process and must take this into consideration when soliciting the sponsor’s time.
6. Any software produced by team jBehaving should conform to any applicable standards as best as possible.
7. The software will be web based and able to run on all major browsers.
8. The software produced by team jBehaving may need to interface with existing systems for social media (Facebook, Twitter) and online commerce (PayPal) as requested by the sponsor.
9. The sponsor bears no financial responsibility for the development process.
10. To reduce development costs and maintenance costs for the sponsor, we will use open source for any software libraries or needed infrastructure when possible.
11. The sponsor may need to provide team jBehaving access to any infrastructure, i.e. web hosting services, needed for the final delivery of the software
12. The sponsor will maintain full access to any infrastructure or hosting services needed for final delivery.

3. SPECIFIC REQUIREMENTS

This section contains all of the technical information and data needed to design the software. The information will be used as a complete set of baseline requirements for the project. The requirements listed are complete, consistent, precise, modifiable, measurable, verifiable, and traceable.

3.1 Use Case Specifications

This subsection documents each of the scenarios identified in the system Use Case Model of section 2. Each Use Case will contain the description, actors, priority, pre and post conditions, relative Use Case(s), and the normal and alternative course(s). Use Cases with priority ‘Need’ are required to deliver a complete system. Use Cases with priority ‘Want’ may be completed depending on time constraints.

The Use Cases for the GotTrackDays.com website are as follows:

1. Access System
   1. Register for Member Account
   2. Sign In
   3. Sign Out
2. Analyze Financial Data
   1. Generate Reports
   2. Use Track Day Calculator
3. Manage Events
   1. Create Event
   2. Update Event
   3. Cancel Event
   4. View Event Manifest
   5. Remove Person from Event

Manifest

* 1. View Forum Log
  2. Add Forum Log Entry
  3. View All Tracks
  4. Search Tracks
  5. Add Track
  6. View Track
  7. Edit Track
  8. Deactivate Track

1. Manage Accounts
   1. Change Internal Account Access Level
   2. Create Internal Account
   3. View All Accounts
   4. Search Accounts
   5. View Account
   6. Anonymize Account
   7. Deactivate Member Account
   8. View Personal Information
   9. Update Personal Information
   10. Request Account Deletion
   11. View Virtual Garage Vehicles
   12. Add Vehicle
   13. Edit Vehicle
   14. Delete Vehicle
2. View Announcements
   1. Add Announcement
   2. Edit Announcement
   3. Remove Announcement
3. Participate in Events
   1. View All Events
   2. Search Events
   3. View Event
   4. Share Event
   5. Register for Event
   6. Request Refund for an Event

**3.1.1 Access System**

|  |  |
| --- | --- |
| UC 1.1 | Register for Member Account |
| Description | Register for an account with credentials to access the member features of the website. |
| Actors | * Guest |
| Priority | Need |
| Precondition(s) | * The guest user does not have a member account registered with their email address. |
| Postcondition(s) | * The guest user will have a member account created. * The member account will be added to the database. * The guest will be signed in to the system with that member account. |
| Related Use Case(s) | UC 1.2 - Sign In |
| Normal Course | 1. User clicks on ‘Sign In or Register’. 2. User clicks ‘Don’t have an account? Click here to register!’. 3. User inputs required personal information for member profile and any car information for the user’s Virtual Garage. 4. User clicks a ‘Submit’ button. 5. System verifies no missing or incorrect data has been entered and that the user does not already have an account. 6. A new member account is created in the database. 7. User is redirected back to the homepage of the website as a logged in member. |
| Alternative Course(s) | Facebook Registration   1. User clicks on ‘Sign In or Register’. 2. User clicks ‘Don’t have an account? Click here to register!’. 3. User clicks ‘Register with Facebook’. 4. If not already logged in to Facebook, user will be required to login using Facebook credentials at this time. 5. User clicks ‘Allow’ to allow the system access to the user’s Facebook profile and data. 6. User inputs any required personal information that was not given by Facebook along with any car information for the user’s Virtual Garage. 7. User clicks a ‘Submit’ button. 8. System verifies no missing or incorrect data has been entered and that the user does not already have an account. 9. A new member account is created in the database. 10. User is redirected back to the homepage of the website as a logged in member.   Google+ Registration   1. User clicks on ‘Sign In or Register’. 2. User clicks ‘Don’t have an account? Click here to register!’. 3. User clicks ‘Register with Google+’. 4. If not already logged in to Google+, user must log in at this time. 5. User clicks ‘Allow’ to allow the system access to the user’s Google+ profile and data. 6. User inputs any required personal information that was not given by Google+ along with any car information for the user’s Virtual Garage. 7. User clicks a ‘Submit’ button. 8. System verifies no missing or incorrect data has been entered and that the user does not already have an account. 9. A new member account is created in the database. 10. User is redirected back to the homepage of the website as a logged in member.   Incorrect or missing information   1. User clicks on ‘Sign In or Register’. 2. User clicks ‘Don’t have an account? Click here to register!’. 3. User inputs required personal information for profile and any car information for the user’s Virtual Garage. 4. User clicks a ‘Submit’ button. 5. Incorrect or missing information found by system. 6. System reload current page with a note at the top in red saying that there are invalid characters and the fields with incorrect information highlighted with red writing. 7. User corrects the missing or incorrect data and clicks the ‘Submit’ button. 8. A new member account is created in the database. 9. User is redirected back to the homepage of the website as a logged in member.   User already has an account   1. User clicks on ‘Sign In or Register’. 2. User clicks ‘Don’t have an account? Click here to register!’. 3. User inputs required personal information for profile and any car information for the user’s Virtual Garage. 4. User clicks a ‘Submit’ button. 5. A duplicate account is found in the database using the provided email address. 6. Display a message that tells the user that the account cannot be created due to an account already existing that uses the provided email address. Display ‘Sign In’ and ‘Forgot your password’ links under the message. |

|  |  |
| --- | --- |
| UC 1.2 | Sign In |
| Description | Signing in to the system |
| Actors | * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * User has a registered internal or member account. |
| Postcondition(s) | * If successful, the user will have access to the system appropriate with their clearance level and will be returned to the last webpage that was viewed or trying to be viewed that required registered user access. * If not successful, the user will be prompted to try again or create a new member account. |
| Related Use Case(s) | UC 1.1 - Register for Member Account  UC 4.2 - Create Internal Account |
| Normal Course | 1. User clicks on ‘Sign In or Register’. 2. User inputs username and password. 3. User clicks a ‘Submit’ button. |
| Alternative Course(s) | Facebook Sign In   1. User clicks on ‘Sign In or Register’. 2. User clicks ‘Sign in with Facebook’. 3. User clicks a ‘Submit’ button. 4. Google+ Sign In 5. User clicks on ‘Sign In or Register’. 6. User clicks ‘Sign in with Google+’. 7. User clicks a ‘Submit’ button.   Forgotten account username or password   1. User clicks on ‘Sign In or Register’. 2. User clicks ‘Forgot my username and/or password’. 3. User inputs the email address associated with account. 4. User answers personal account security question successfully. 5. Username is displayed on screen and the user inputs a new password twice. 6. System verifies passwords match. 7. New password is saved for that member account for future access. |

|  |  |
| --- | --- |
| UC 1.3 | Sign Out |
| Description | Signing out of the system |
| Actors | * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * The user must be currently logged into the system. |
| Postcondition(s) | * The user will no longer be logged into the system and will be recognized as a guest. |
| Related Use Case(s) | UC 1.2 - Sign In |
| Normal Course | 1. User clicks on ‘Sign Out’. 2. User is then redirected back to the homepage of the website as a Guest. |
| Alternative Course(s) | None |

**3.1.2 Analyze Financial Data**

|  |  |
| --- | --- |
| UC 2.1 | Generate Financial Report |
| Description | The process by which a user automatically generates a financial report based on event information in the database. |
| Actors | * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. |
| Postcondition(s) | * A .csv file will be generated for the chosen financial report. |
| Related Use Case(s) | UC 1.2 - Sign in  UC 6.3 - View Event |
| Normal Course | 1. User navigates to the specific event page for which he/she wishes to create the financial report for. 2. User clicks the ‘Generate Report’ button. 3. System automatically starts to download the .csv file which contains the financial report. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 2.2 | Use Track Day Calculator |
| Description | The user uses the track day calculator to compare the cost of an event from the GotTrackDays.com website versus a leading competitor. |
| Actors | * Guest * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * Event Manager or Accountant has already filled out general competitor pricing information for the specific event. * User is on specific event page. |
| Postcondition(s) | * User is shown comparison of Got Track Days Inc. event costs compared to leading competitor. |
| Related Use Case(s) | UC 6.3 - View Event |
| Normal Course | 1. User clicks on ‘Track Day Calculator’ button at the bottom of the event page. 2. User is then redirected to a page with the Track Day Calculator on it. 3. Information for the general competitor will show up on the right column pre populated from information from the database. 4. Information for Got Track Days Inc. event will be filled in the left column. 5. Customer will then have the option to fill out other travel expenses including, but not limited to, hotel cost, gas to and from event, and food cost. 6. Customer will then press the ‘Recalculate’ Button. 7. System verifies correct information filled out in each field. 8. New total price will be updated on the Got Track Days Inc.’s left side column. |
| Alternative Course(s) | Fields Not Filled Out Correctly   1. User clicks on ‘Track Day Calculator’ button. 2. User is then redirected to a page with the Track Day Calculator on it. 3. Information for the general competitor will show up on the right column pre populated from information from the database. 4. Information for Got Track Days Inc. event will be filled in the left column. 5. Customer will then have the option to fill out other travel expenses including, but not limited to, hotel cost, gas to and from event, and food cost. 6. Customer will then press the ‘Recalculate’ button. 7. System verifies correct information filled out in each field. 8. Field filled out incorrectly, such as incorrect amount of decimal places or invalid characters. 9. System reload current page with a note at the top in red saying that there are invalid characters and the fields with incorrect information highlighted with red writing. 10. User will then re-fill out the incorrect fields and press ‘Recalculate’ button. 11. System verifies correct information filled out in each field. 12. New total price will be updated on the Got Track Days Inc.’s left side column. |

**3.1.3 Manage Events**

|  |  |
| --- | --- |
| UC 3.1 | Create Event |
| Description | The process by which a user creates a new track day event. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in to a qualifying account. |
| Postcondition(s) | * A new track day event is created. |
| Related Use Case(s) | UC 1.2 - Sign In  UC 6.1 - View All Events |
| Normal Course | 1. User clicks on the ‘Events’ tab. 2. User clicks on the ‘New Event’ button. 3. User is presented with a form for entering event details. 4. User must enter all required information such as date, time, track, and price. 5. Once the information has been entered, the user clicks the ‘Submit’ button. 6. System verifies no missing or incorrect information. 7. User is redirected to the newly created event page. |
| Alternative Course(s) | Missing or Incorrect Information   1. User clicks on the ‘Events’ tab. 2. User clicks on the ‘New Event’ button. 3. User is presented with a form for entering event details. 4. User must enter all required information such as date, time, track, and price. 5. Once the information has been entered, the user clicks the ‘Submit’ button. 6. System verifies no missing or incorrect information. 7. System reload current page with a note at the top in red saying that there are invalid characters and the fields with incorrect information highlighted with red writing. 8. User corrects the information and clicks the ‘Submit’ button. 9. System verifies no missing or incorrect information. 10. User is redirected to the newly created event page. |

|  |  |
| --- | --- |
| UC 3.2 | Update Event |
| Description | The process by which a user updates the information of an existing event. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. * The event exists in the system. * The user is on the specific event page. |
| Postcondition(s) | * The event’s fields are changed in the event table |
| Related Use Case(s) | UC 1.2 - Sign In  UC 6.3 - View Event |
| Normal Course | 1. User clicks on the ‘Events’ tab. 2. User clicks on the event for which he/she wishes to edit information for. 3. User is redirected to the specific event page. 4. User clicks the ‘Edit’ button. 5. User is presented with a form for editing event details which is prepopulated with the current information. 6. User edits any information and then clicks the ‘Submit’ button. 7. System verifies no missing or incorrect information. 8. User is redirected to the newly created event page. |
| Alternative Course(s) | Missing or Incorrect Information   1. User clicks on the ‘Events’ tab. 2. User clicks on the event for which he/she wishes to edit information for. 3. User is redirected to the specific event page. 4. User clicks the ‘Edit’ button. 5. User is presented with a form for editing event details which is prepopulated with the current information. 6. User edits any information and then clicks the ‘Submit’ button. 7. System verifies no missing or incorrect information. 8. System reloads the form highlighting which information has been filled out incorrectly or is missing. 9. User fixes missing or incorrect information. 10. User clicks ‘Submit’ button. 11. System verifies no missing or incorrect information. 12. User is redirected to the newly created event page. |

|  |  |
| --- | --- |
| UC 3.3 | Cancel Event |
| Description | The process of deleting an event. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. * The event exists in the system. |
| Postcondition(s) | * The event is removed from the database and website. |
| Related Use Case(s) | UC 1.2 - Sign In  UC 6.3 - View Event  UC 6.7 - Request Refund for Event |
| Normal Course | 1. User clicks on the ‘Events’ tab. 2. User clicks on the event for which he/she wishes to delete. 3. User is redirected to the specific event page. 4. User clicks the ‘Delete’ button. 5. User is prompted to verify deletion of the event. 6. User clicks ‘Ok’. 7. Event search page is reloaded with the deleted event no longer present. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 3.4 | View Event Manifest |
| Description | Event manifest is shown on screen with a list of who is registered for the event and the car they will be driving along with any other pertinent information for that specific track day event. |
| Actors | * Standard Employee * Event Managers * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. * User is viewing a specific event page. |
| Postcondition(s) | * The manifest is displayed on the webpage. |
| Related Use Case(s) | UC 1.2 - Sign In  UC 6.3 - View Event  UC 3.5 - Remove Person from Event Manifest |
| Normal Course | 1. User clicks the ‘View Manifest’ button. 2. User is presented with the manifest for the event. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 3.5 | Remove Person from Event Manifest |
| Description | Removing a user registered for an event from the manifest after a refund has been processed. |
| Actors | * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * A refund has been processed for a user attending an event * Employee is viewing the event manifest for the event to remove the user from. |
| Postcondition(s) | * Removed users no longer displayed on manifest * Auto-generated confirmation email sent to removed users |
| Related Use Case(s) | UC 3.4 - View Event Manifest  UC 6.7 - Request Refund for Event |
| Normal Course | 1. User clicks ‘Remove from Manifest’ button on manifest entry for user. 2. User presented with prompt to confirm removal 3. User enters refund confirmation number 4. User enters optional comment 5. User clicks ‘Remove’ to confirm removal 6. User returned to event manifest screen 7. Manifest entry for user is removed. |
| Alternative Course(s) | User does not confirm removal   1. User clicks ‘Remove from Manifest’ button on manifest entry for user. 2. User presented with prompt to confirm removal. 3. User clicks “Cancel” button. 4. User is returned to Event Manifest Screen. |

|  |  |
| --- | --- |
| UC 3.6 | View Forum Log |
| Description | Internal user views the forum log for an existing track day event. |
| Actors | * Standard Employee * Accountant * Administrator * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. * The track day event exists in the system |
| Postcondition(s) | None |
| Related Case(s) | UC 1.2 - Sign In  UC 6.3 - View Event |
| Normal Course | 1. User navigates to specific event page. 2. User then scrolls to the part of the page that shows the table with all current forum postings for the event that they are on the page of. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 3.7 | Add Forum Log Entry |
| Description | Internal user creates a new log entry into the log. |
| Actors | * Standard Employee * Accountant * Administrator * Event Manager * Owner |
| Priority | Need |
| Preconditions(s) | * Event exists in the system. * User has signed in with qualifying account. |
| Postcondition(s) | * New forum log entry is displayed in forum log. |
| Related Use Case(s) | UC 1.2 Sign In  UC 3.6 View Forum Log |
| Normal Course | 1. User navigates to specific event page. 2. User scrolls to the part of the page that shows the forum log. 3. User clicks on the ‘Add’ button to create a new entry. 4. A window pops up with a form. 5. User types information into fields to create a new forum log entry. 6. User clicks the ‘Create’ button to add the new entry into the log. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 3.8 | View All Tracks |
| Description | Viewing all the available tracks currently in the database. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * User has signed in with a qualifying account. |
| Postcondition(s) | * User is viewing a list of all tracks in the database. |
| Related Use Case(s) | UC 1.2 - Sign in  UC 3.9 - Search Tracks  UC 3.10 - Add Track  UC 3.11 - View Track |
| Normal Course | 1. User clicks Manage Track link. 2. User is taken to the Manage Track screen. 3. User is presented with a list of all tracks. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 3.9 | Search Tracks |
| Description | Searching for a track from the Manage Track screen. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * User has signed in with a qualifying account. * User is on the Manage Tracks screen. |
| Postcondition(s) | * Manage Tracks screen displaying list based on search criteria. |
| Related Use Case(s) | UC 1.2 - Sign in  UC 3.8 - View All Tracks  UC 3.10 - Add Track  UC 3.11 - View Track |
| Normal Course | 1. User enters search criteria on Manage Tracks screen such as track name and address. 2. User clicks ‘Search’ button. 3. User is presented with that meet criteria. |
| Alternative Course(s) | User wants to sort the list of tracks in descending order   1. User clicks on column heading of list of tracks. 2. List of tracks is sorted on that column based on descending alphanumeric order. 3. User wants to sort the list of tracks in ascending order. 4. User clicks on a column heading twice. 5. List of tracks is sorted on that column based on ascending alphanumeric order. |

|  |  |
| --- | --- |
| UC 3.10 | Add Track |
| Description | Adding a track to the available tracks for new events. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * User has signed in with a qualifying account. * User is on the Manage Tracks screen. |
| Postcondition(s) | * New track added to the list of tracks. |
| Related Use Case(s) | UC 1.2 - Sign in  UC 3.8 - View All Tracks  UC 3.9 - Search Tracks |
| Normal Course | 1. Click on ‘New Track’ button. 2. User is presented with the New Track screen. 3. User fills out fields for track information such as track name, address and a contact person’s name and phone number. 4. User clicks the ‘Submit’ button. 5. User is returned to the Manage Tracks screen. |
| Alternative Course(s) | User decides to cancel New Track creation   1. Click on “New Track” button. 2. User is presented with the New Track screen. 3. User clicks “Cancel” button. 4. User is returned to Manage Tracks Screen.   Invalid or Missing Information Entered   1. Click on ‘New Track’ button. 2. User is presented with the New Track screen. 3. User fills out fields for track information such as track name, address and a contact person’s name and phone number. 4. One or more fields filled out incorrectly 5. User clicks the ‘Submit’ button. 6. System checks fields 7. One or more fields filled out incorrectly 8. System reloads the form highlighting which information has been filled out incorrectly or is missing. 9. User corrects information 10. User clicks the ‘Submit’ Button 11. User is returned to the Manage Tracks Screen |

|  |  |
| --- | --- |
| UC 3.11 | View Track |
| Description | Details, such as track name, address, contact info related to a specific track is displayed to viewers. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Preconditions(s) | * A track exists in the database. * User is viewing the Manage Tracks screen. |
| Postcondition(s) | * Detailed information of a specific track is displayed. |
| Related Use Case(s) | UC 3.8 - View All Tracks  UC 3.9 - Search Tracks  UC 3.10 - Update Track  UC 3.12 - Deactivate Track |
| Normal Course | 1. User clicks on the name of a track in the list of tracks. 2. The track information is displayed on the screen. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 3.12 | Edit Track |
| Description | The information for an existing track needs to be changed. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with a qualifying account. * User is on the Manage Tracks screen. * The track exists in the system. * User has found a track to change via search or viewing all tracks. |
| Postcondition(s) | * Any edited tracks display updated information in the Manage Tracks screen. |
| Related Use Case(s) | UC 1.2 - Sign in  UC 3.11 - View Track |
| Normal Course | 1. User clicks “Edit Track” button. 2. The user is presented with the Edit Track screen. 3. The user edits fields to be changed. 4. The user clicks “Submit” button. 5. The user is returned to the Track Management screen. |
| Alternative Course(s) | The user doesn’t want to update the selected track   1. User clicks “Edit Track” button. 2. The user is presented with the Edit Track screen. 3. The user clicks the “Cancel” button. 4. The user is returned to the Track Management screen. 5. The information for the selected track has not been changed. |

|  |  |
| --- | --- |
| UC 3.13 | Deactivate Track |
| Description | The company is no longer doing businesses with a track and wants to remove it from the list of tracks available for events. |
| Actors | * Event Manager * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with a qualifying account. * User is on the Manage Tracks screen. * A track has been added to the system. * User has found a track to deactivate via search or viewing all tracks. |
| Postcondition(s) | * The track is listed as deactivated. * The track will no longer be available when creating or updating events. * The track will remain in the database to maintain referential integrity. |
| Related Use Case(s) | UC 1.2 - Sign in  UC 3.11 - View Track |
| Normal Course | 1. The user clicks ‘Deactivate’ button for the desired track. 2. The user is prompted to confirm the deactivation. 3. The user clicks the ‘Confirm’ button. 4. The user is returned to the Manage Event screen. 5. The desired track will be listed as ‘Deactivated’. |
| Alternative Course(s) | The user decides not to deactivate the selected track   1. The user clicks the ‘Deactivate’ button for the desired track. 2. The user is prompted to confirm the deactivation. 3. The user clicks the ‘Cancel’ button. 4. The user is returned to the Manage Event screen. 5. No changes have been made to the desired track. |

**3.1.4 Manage Accounts**

|  |  |
| --- | --- |
| UC 4.1 | Change Internal Account Access Level |
| Description | As an employee is promoted or demoted within the company Administrators will be able to change the employee’s permission level to go along with their new role. |
| Actors | * Administrator * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with qualifying account. * User is viewing the specific account for which the changes are to be made for. |
| Postcondition(s) | * The account being changed has had their access level changed in the database. |
| Related Use Case(s) | UC 1.2 - Sign in  UC 4.5 - View Account |
| Normal Course | 1. User clicks on the ‘Users’ tab. 2. User searches for the specific user account and clicks on the checkbox to select the user account. 3. User clicks on the ‘Edit Access Level’ button. 4. User is then presented with a dropdown box of all possible access levels (user roles). 5. User clicks ‘Submit’ button. 6. Webpage redirects back to user account page to show that the changes were updated for the account and in the database. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 4.2 | Create Internal Account |
| Description | When a new employee is hired by the company, a new account must be created with the appropriate permissions level. |
| Actors | * Administrator * Owner |
| Priority | Need |
| Precondition(s) | * The user is registered and has signed into a qualifying account. |
| Postcondition(s) | * Internal user account has been created. |
| Related Case(s) | UC 1.2 - Sign In |
| Normal Course | 1. User clicks on the ‘Accounts’ tab. 2. User clicks on the ‘Create new account’ button. 3. User enters new internal user information. 4. User clicks the ‘Create’ button. 5. System verifies no missing or incorrect information. 6. Webpage reloads to list of current accounts with newly created account on the list. |
| Alternative Course(s) | Missing or Incorrect information entered   1. User clicks on the ‘Accounts’ tab. 2. User clicks on the ‘Create new account’ button. 3. User enters new internal user information. 4. User clicks the ‘Create’ button. 5. System verifies information and finds incorrect or invalid information 6. System reloads the form highlighting which information has been filled out incorrectly or is missing. 7. User corrects form information 8. User clicks the ‘Create’ button 9. Webpage reloads to list of current accounts with newly created account on the list. |

|  |  |
| --- | --- |
| UC 4.3 | View All Accounts |
| Description | User is viewing all accounts that are in the database. |
| Actors | * Administrator * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with a qualifying account. |
| Postcondition(s) | * User is viewing a list of all accounts in the database. |
| Related Use Case(s) | * UC 2.1 - Sign In * UC 4.2 - Create Internal Account * UC 4.4 - Search Account * UC 4.5 - View Account |
| Normal Course | 1. User clicks on the ‘Accounts’ tab. 2. User is then presented with a list of all accounts in the database. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 4.4 | Search Accounts |
| Description | User is able to search through the list of users by keyword. |
| Actors | * Administrator * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with a qualifying account. * User is on ‘Accounts’ tab. |
| Postcondition(s) | * User is presented with a filtered list of users in the database. |
| Related Use Case(s) | UC 2.1 - Sign In  UC 4.3 - View All Accounts |
| Normal Course | 1. From the ‘Accounts’ tab the user will be presented with a text box for entering in search keywords. 2. After entering in the keywords, the user will press the ‘Search’ button. 3. The system will then filter all the users by the search keyword and present the user with a new list of users whose account information meets the search keywords. |
| Alternative Course(s) | No Results   1. From the ‘Accounts’ tab the user will be presented with a text box for entering in search keywords. 2. After entering in the keywords, the user will press the ‘Search’ button. 3. The system will then search through the users for any information matching the keywords. 4. The user is then presented with a message explaining that no results were found matching the keywords chosen. |

|  |  |
| --- | --- |
| UC 4.5 | View Account |
| Description | A specific account and the account information is displayed. |
| Actors | * Administrator * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with a qualifying account. * The account being viewed has already been created. |
| Postcondition(s) | * Specific details of an account is displayed to the user. |
| Related Use Case(s) | UC 2.1 - Sign In  UC 4.6 - Anonymize Account  UC 4.7 - Deactivate Member Account |
| Normal Course | 1. User clicks on the ‘Accounts’ tab. 2. Users searches for a specific account. 3. User clicks on the specific account. 4. A web page of detailed information of that account is displayed. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 4.6 | Anonymize Account |
| Description | Account being anonymized has all of their information set to null in the database. |
| Actors | * Administrator * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with a qualifying account. * User is viewing the individual account page whose account is going to be anonymized. |
| Postcondition(s) | * All information in the database for the specific account is set to null. |
| Related Use Case(s) | UC 2.1 - Sign In  UC 4.5 - View Account |
| Normal Course | 1. User is currently viewing the account he/she wishes to anonymize. 2. User clicks the ‘Anonymize’ button. 3. User is asked to verify that they wish to anonymize the account. 4. User clicks ‘Yes’. 5. User is then redirected back to the view all accounts screen with the anonymized user not present. |
| Alternative Course(s) | User Decides Not to Anonymize Account   1. User is currently viewing the account he/she wishes to anonymize. 2. User clicks the ‘Anonymize’ button. 3. User is asked to verify that they wish to anonymize the account. 4. User clicks ‘No’. 5. User is then redirected back to the view account page of the current user in question. |

|  |  |
| --- | --- |
| UC 4.7 | Deactivate Member Account |
| Description | A member’s account can be deactivated by an Administrator if the member is not displaying appropriate conduct within the website or at the events. |
| Actors | * Administrator * Owner |
| Priority | Want |
| Preconditions(s) | * User has signed in with a qualifying account. * Member’s account is active. |
| Postcondition(s) | * Member’s account is deactivated. |
| Related Use Case(s) | UC 2.1 - Sign In  UC 4.5 - View Account |
| Normal Course | 1. User clicks on the ‘Users’ tab. 2. A list of all members are displayed in the top section of the webpage. 3. User searches for the specific member in the list. 4. User selects the checkbox next to the member’s name. 5. User clicks the ‘Deactivate Account’ button. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 4.8 | View Personal Information |
| Description | User is able to view all their personal account information. |
| Actors | * Member * Standard Employee * Event Manager * Accountant * Administrator * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with their own account. |
| Postcondition(s) | * User is viewing their personal information that is stored the database. |
| Related Use Case(s) | UC 1.2 - Sign In |
| Normal Course | 1. Once user is logged in to their account they will be on the homepage of the website. 2. User clicks the ‘Account’ tab. 3. User is redirected to a page showing them all the current information stored as part of their account. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 4.9 | Update Personal Information |
| Description | Member personal information changes are made and saved successfully. |
| Actors | * Member * Standard Employee * Event Manager * Accountant * Administrator * Owner |
| Priority | Need |
| Precondition(s) | * Member account exists in the database. * Member is signed in and viewing his/her account profile page. |
| Postcondition(s) | * Account profile displays updated content. |
| Related Use Case(s) | UC 1.2 - Sign In  UC 4.8 - View Personal Information |
| Normal Course | 1. User clicks on ‘Edit’ button. 2. User updates information in fields available in the ‘Account’ tab. 3. Once finished with all the updates, user clicks on the ‘Save’ button to save the changes. 4. System verifies no incorrect or missing information is present. 5. ‘Account’ tab refreshes and displays the user’s profile with new information. |
| Alternative Course(s) | Error Saving Changes   1. User clicks on ‘Edit’ button. 2. User updates information in fields available in the ‘Account’ tab. 3. Once finished, user clicks the ‘Save’ button to save changes. 4. System finds incorrect or missing information present. 5. System reloads the form highlighting which information has been filled out incorrectly or is missing. . 6. User changes the incorrect information and resubmits the save request by clicking the ‘Save’ button again. 7. System verifies no incorrect or missing information is present. 8. ‘Account’ tab refreshes and displays a small the users profile with new information. |

|  |  |
| --- | --- |
| UC 4.10 | Request Account Deletion |
| Description | An email is sent to request that a user account is deleted from the system. |
| Actors | * Member |
| Priority | Need |
| Preconditions(s) | * User has an account created. * User is signed in. |
| Postcondition(s) | * An auto-generated email is sent to the company mailbox requesting account deletion. |
| Related Use Case(s) | UC 1.2 - Sign In  UC 4.6 - Anonymize Account |
| Normal Course | 1. User clicks on the ‘Profile’ tab. 2. User’s personal information is displayed. 3. User clicks on the link ‘Request Account Deletion’. 4. A pop-up window displays asking user to confirm request. 5. User clicks the ‘Ok’ button to confirm request. |
| Alternative Course(s) | Request Not Confirmed   1. User clicks on the ‘Profile’ tab. 2. User’s personal information is displayed. 3. User clicks on the link ‘Request Account Deletion’. 4. A pop-up window displays asking user to confirm request. 5. User clicks the ‘Cancel’ button in the pop-up window. 6. Pop-up window closes and user’s personal information is displayed. |

|  |  |
| --- | --- |
| UC 4.11 | View Virtual Garage Vehicles |
| Description | Vehicles that are in the Virtual Garage are displayed to the user. |
| Actors | * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Preconditions(s) | * User has signed in with a qualifying account. |
| Postcondition(s) | * Vehicles in user’s garage is displayed. |
| Related Use Case(s) | UC 1.2 Sign In |
| Normal Course | 1. User clicks on the ‘Profile’ tab. 2. One section of the page displayed will show the vehicles that the user has added to his/her profile. |
| Alternative Course(s) | No Vehicles Added   1. User clicks on the ‘Profile’ tab. 2. In the section of the page that shows the garage vehicles, the user will see a form with fields and a ‘Add Vehicle’ button. |

|  |  |
| --- | --- |
| UC 4.12 | Add Vehicle |
| Description | The user wants to add vehicles to their Virtual Garage. |
| Actors | * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. |
| Postcondition(s) | * The user has an additional vehicle in their garage. |
| Related Use Case(s) | UC 1.2 - Sign In  UC 4.11 - View Virtual Garage Vehicles |
| Normal Course | 1. User clicks on ‘Profile’ tab. 2. User clicks on the ‘Manage My Garage’ button. 3. User clicks the ‘Add New Vehicle’ button. 4. User enters in all required information. 5. User clicks the ‘Submit’ button 6. System verifies no missing or incorrect information. 7. Web page is refreshed to show the user’s garage with the newly added vehicle present. |
| Alternative Course(s) | Missing or Incorrect Information   1. User clicks on ‘Profile’ tab. 2. User clicks on the ‘Manage My Garage’ button. 3. User clicks the ‘Add New Vehicle’ button. 4. User enters in all required information. 5. User clicks the ‘Submit’ button 6. System verifies no missing or incorrect information. 7. System reloads current page showing which fields have missing or incorrect information. 8. User fixes information in specific text boxes. 9. User clicks ‘Submit’ button. 10. System verifies no missing or incorrect information. 11. Web page is refreshed to show the user’s garage with the newly added vehicle present. |

|  |  |
| --- | --- |
| UC 4.13 | Edit Vehicle |
| Description | The user wants to change properties of a vehicle listed in their Garage. |
| Actors | * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. * The vehicle the user wishes to update has already been created. |
| Postcondition(s) | * The changes are reflected in the user’s Garage. |
| Related Use Case(s) | UC 1.2 - Sign In  UC 4.11 - View Virtual Garage Vehicles |
| Normal Course | 1. User clicks on the ‘Manage My Garage’ button. 2. User clicks on the vehicle he/she wishes to update. 3. User clicks ‘Update’ button. 4. User is presented with a form for editing vehicle details which is prepopulated with the current information. 5. User edits any information and then clicks the ‘Submit’ button. 6. System verifies no missing or incorrect information. 7. User is redirected to their garage with the newly edited event page. |
| Alternative Course(s) | Missing or Incorrect Information   1. User clicks on the ‘Manage My Garage’ button. 2. User clicks on the vehicle he/she wishes to update. 3. User clicks ‘Update’ button. 4. User is presented with a form for editing vehicle details which is prepopulated with the current information. 5. User edits any information and then clicks the ‘Submit’ button. 6. System verifies no missing or incorrect information. 7. System reloads current page showing which fields have missing or incorrect information. 8. User fixes information in specific text boxes. 9. User clicks ‘Submit’ button. 10. System verifies no missing or incorrect information. 11. User is redirected to their garage with the newly edited event page. |

|  |  |
| --- | --- |
| UC 4.14 | Delete Vehicle |
| Description | The user wants to remove vehicles listed in their Virtual Garage. |
| Actors | * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. * The vehicle which the user wishes to delete has already been created. |
| Postcondition(s) | * The desired vehicles are no longer shown in the user’s Garage. |
| Related Use Case(s) | UC 1.2 - Sign In  UC 4.11 - View Virtual Garage Vehicles |
| Normal Course | 1. User clicks on ‘Profile’ tab. 2. User clicks on the ‘Manage My Garage’ button. 3. User clicks on the vehicle he/she wishes to delete. 4. System prompts user to verify deletion of vehicle. 5. User clicks ‘Ok’ button. 6. Page reloads to show all current vehicles in the user’s garage with the newly deleted vehicle no longer present. |
| Alternative Course(s) | User Decides Not to Delete Vehicle   1. User clicks on ‘Profile’ tab. 2. User clicks on the ‘Manage My Garage’ button. 3. User clicks on the vehicle he/she wishes to delete. 4. System prompts user to verify deletion of vehicle. 5. User clicks ‘Cancel’ button. 6. Page reloads to show all current vehicles in the user’s garage with no vehicles deleted. |

**3.1.5 View Announcements**

|  |  |
| --- | --- |
| UC 5.1 | Add Announcement |
| Description | The process by which the user creates a new announcement to the homepage of the GotTrackDays.com website. |
| Actors | * Standard Employee * Accountant * Administrator * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. |
| Postcondition(s) | * An announcement is added to the top of the announcements on the front page with the date and time of the posting. |
| Related Use Case(s) | UC 1.2 - Sign In |
| Normal Course | 1. User is on the homepage of the website. 2. User clicks on the ‘Add Announcement’ button. 3. The user is presented with text boxes for inputting a title and body of the announcement. 4. The user enters information into both fields. 5. The user clicks on ‘Submit’ button. 6. System verifies no missing or incorrect information entered. 7. Page reloads with newly created announcement on the top of the page. |
| Alternative Course(s) | Missing or Incorrect Information   1. User is on the homepage of the website. 2. User clicks on the ‘Add Announcement’ button. 3. The user is presented with text boxes for inputting a title and body of the announcement. 4. The user enters information into both fields. 5. The user clicks on ‘Submit’ button. 6. System verifies no missing or incorrect information entered. 7. System reloads page with text fields indicating which fields are not filled out correctly. 8. User enters in new text for the fields. 9. User clicks ‘Submit’ button. 10. System verifies no missing or incorrect information. 11. Page reloads with newly created announcement on the top of the page. |

|  |  |
| --- | --- |
| UC 5.2 | Edit Announcement |
| Description | The process by which a user edits an existing announcement. |
| Actors | * Standard Employee * Accountant * Administrator * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. * Announcement has previously been created and is displayed on homepage of website. * User is currently on the homepage of the website. |
| Postcondition(s) | * The announcement reappears on the front page with the newly updated title and body. |
| Related Use Case(s) | UC 1.2 - Sign In |
| Normal Course | 1. The user clicks on the update button next to an existing announcement on the homepage. 2. The user is presented with text boxes for inputting a title and body of the message, the text boxes are pre-populated with the existing announcement. 3. The user enters any new information into both fields. 4. The user clicks on ‘Submit’ button. 5. System verifies no missing or incorrect information. 6. Page reloads with newly edited announcement. |
| Alternative Course(s) | Missing or Incorrect Information   1. The user clicks on the update button next to an existing announcement on the homepage. 2. The user is presented with text boxes for inputting a title and body of the message, the text boxes are pre-populated with the existing announcement. 3. The user enters any new information into both fields. 4. The user clicks on ‘Submit’ button. 5. System verifies no missing or incorrect information. 6. System reloads page with text fields indicating which fields are not filled out correctly. 7. User enters in new text for the fields. 8. User clicks ‘Submit’ button. 9. System verifies no missing or incorrect information. 10. Page reloads with newly edited announcement. |

|  |  |
| --- | --- |
| UC 5.3 | Remove Announcement |
| Description | The process by which a user deletes an existing announcement. |
| Actors | * Standard Employee * Accountant * Administrator * Event Manager * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with a qualifying account. * Announcement has previously been created and is displayed on homepage of website. * User is currently on the homepage of the website. |
| Postcondition(s) | * The announcement is removed from the web page. |
| Related Use Case(s) | UC 1.2 - Sign In |
| Normal Course | 1. The user clicks on the ‘Delete’ button next to an existing announcement on the homepage. 2. A prompt asks the user to verify whether or not they actually want to delete the announcement. 3. User clicks ‘Yes’ button. 4. Page reloads with the announcement no longer present. |
| Alternative Course(s) | None |

**3.1.6 Participate in Events**

|  |  |
| --- | --- |
| UC 6.1 | View All Events |
| Description | A list of all events is displayed to the user. |
| Actors | * Guest * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Preconditions(s) | * Events have been created. |
| Postcondition(s) | * List of all events are displayed. |
| Related Use Case(s) | UC 3.1 - Create Event  UC 6.3 - Search Events |
| Normal Course | 1. User clicks on the ‘Events’ tab. 2. A list of all events are displayed to user in order of most current to least current. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 6.2 | Search Events |
| Description | View, retrieve and display a list of events that meet the search criteria. |
| Actors | * Guest * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * Search criteria can be inputted. |
| Postcondition(s) | * Search criteria that was previously inputted is still in the search criteria fields. * Search results are displayed. |
| Related Use Case(s) | UC 6.1 View All Events |
| Normal Course | 1. User clicks on the ‘Events’ tab on the website. 2. User inputs search criteria such as location and date range of events. 3. User clicks on ‘Search’ button. 4. Search results are displayed based on search criteria entered by the user. |
| Alternative Course(s) | No Search Criteria   1. User clicks on the ‘Events’ tab. 2. User does not input search criteria. 3. User clicks on ‘Search’ button. 4. A list of all events are displayed.   No Results   1. User clicks on the ‘Events’ tab. 2. User inputs search criteria such as location and date range of events if user wants to search by certain criteria, else user can leave search criteria blank. 3. User clicks the ‘Search’ button. 4. A pop-up window displays relaying the message that “No results found” to the user when no results match the search criteria. |

|  |  |
| --- | --- |
| UC 6.3 | View Event |
| Description | Details, such as cost, directions, and a list of all attendees, related to a specific event is displayed to viewers. |
| Actors | * Guest * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Preconditions(s) | * Event has been created. |
| Postcondition(s) | * Detailed information of a specific event is displayed. |
| Related Use Case(s) | UC 3.2 - Update Event  UC 3.3 - Cancel Event  UC 3.4 - View Event Manifest  UC 3.6 - View Forum Log  UC 6.1 - View All Events  UC 6.4 - Share Event |
| Normal Course | 1. User clicks on the ‘Events’ tab and a list of the most current events are displayed. 2. User clicks on a specific event from the list displayed. 3. A new page containing detailed information such as cost of attending event, directions to event location, and a list of all attendees currently registered for the event is displayed. |
| Alternative Course(s) | None |

|  |  |
| --- | --- |
| UC 6.4 | Share Event |
| Description | The user shares an event on a social media site (i.e. Facebook or Google+). |
| Actors | * Guests * Members * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Want |
| Precondition(s) | * The user is on the specific event page for which he/she wishes to share. |
| Postcondition(s) | * The event is posted to their profile in Facebook or Google+. |
| Included Use Case(s) | UC 6.3 - View Event |
| Normal Course | 1. User clicks the ‘Share on Facebook’ or ‘Share on Google+’ button. 2. User receives a popup from Facebook or Google+ to verify they want to post the event. 3. User uses the Facebook.com or Google+ interface to complete the post. 4. The popup is closed when the user finishes. |
| Alternative Course(s) | None |

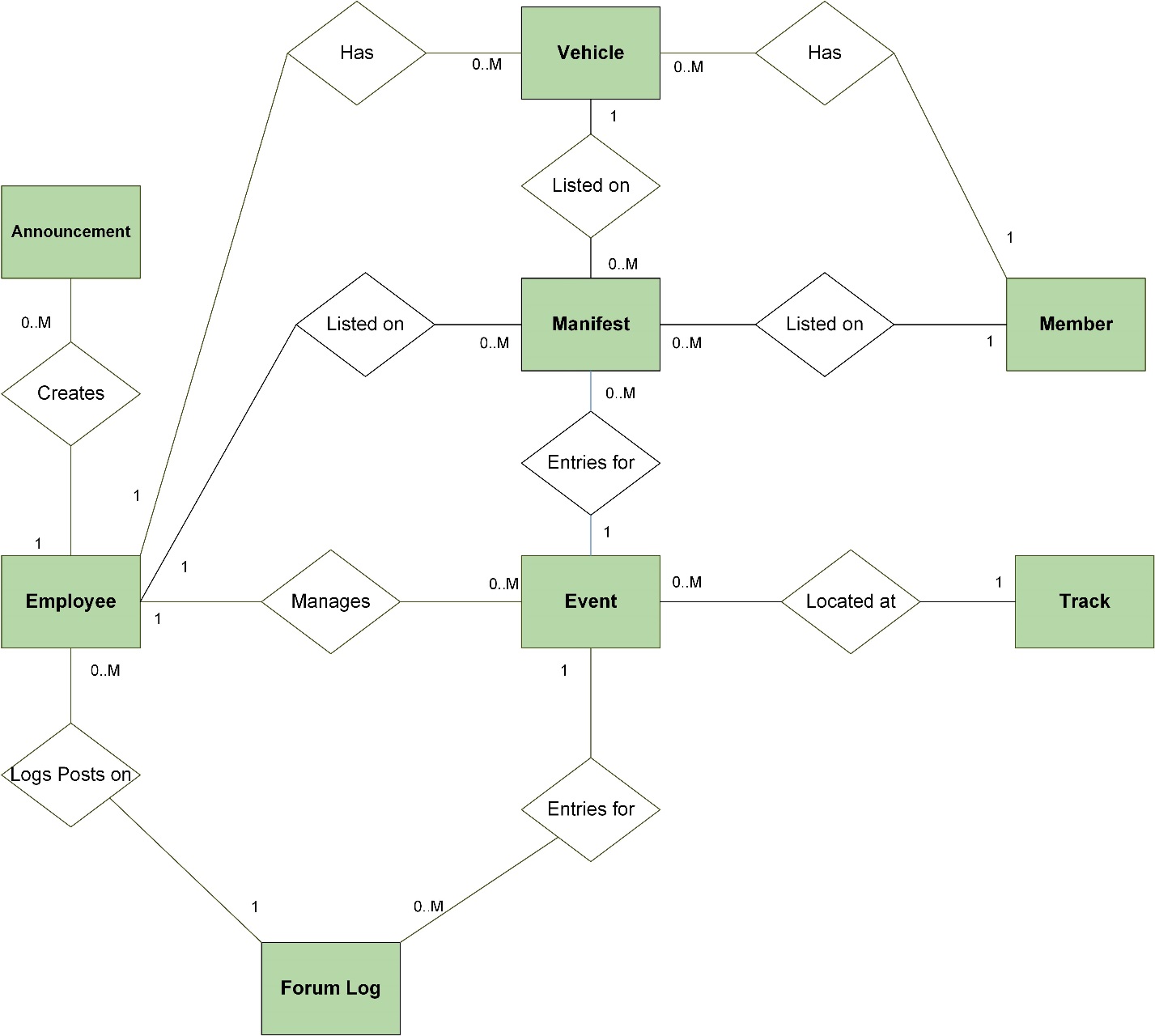
|  |  |
| --- | --- |
| UC 6.5 | Register for Event |
| Description | User registers and pays for a track day event and added to the manifest. |
| Actors | * Member * Standard Employee * Event Manager * Human Resource * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * The user has signed in with qualifying account. * User must be on the event search page or the single event page. |
| Postcondition(s) | * After the user pays for the event, a confirmation page will appear and the user will be added to the database as registered for the event |
| Related Use Case(s) | UC 1.2 - Sign In  UC 6.3 - View Event |
| Normal Course | 1. User clicks on the ‘Register for Event’ button. 2. User will then be redirected to the event registration page. 3. On the event registration page, the user will fill out all necessary information. 4. User will click the ‘Pay for Event’ button. 5. System will verify no missing or incorrect information. 6. System will redirect user to PayPal page to pay for the event. 7. User will then use the PayPal website to submit their payment. 8. After payment has been submitted, user will be redirected back to confirmation page. 9. Confirmation page will have a confirmation number and a small thank you section which thanks the customer for using GotTrackDays.com to register for their event. |
| Alternative Course(s) | Incorrect or missing information   1. User clicks ‘Register for Event’ button. 2. User will then be redirected to the event registration page. 3. On the event registration page, the user will fill out all necessary information. 4. User will click the ‘Pay for Event’ button. 5. System will verify no missing or incorrect information. 6. Because information is missing, the system will reload the current page with sections that need information highlighted. 7. User will fill out sections that had missing information. 8. User will click the ‘Pay for Event’ button. 9. System will verify no missing or incorrect information. 10. System will redirect user to PayPal page to pay for the event. 11. User will then use the PayPal website to submit their payment. 12. After payment has been submitted, user will be redirected back to confirmation page. 13. Confirmation page will have a confirmation number and a small thank you section which thanks the customer for using GotTrackDays.com to register for their event. |

|  |  |
| --- | --- |
| UC 6.6 | Request Refund |
| Description | User requests a refund for a registration fee that has already been paid because they can no longer attend the event. |
| Actors | * Member * Standard Employee * Event Manager * Administrator * Accountant * Owner |
| Priority | Need |
| Precondition(s) | * User has signed in with their account. * Registration to an event has been completed. * Payment has been paid. |
| Postcondition(s) | * Request awaits approval. * Approved requests will process the portion of the paid cost to the requestor. |
| Related Use Case(s) | UC 6.5 - Register for Event |
| Normal Course | 1. User clicks on the ‘Contact’ tab. 2. User clicks on the link ‘Request a refund’. 3. User selects the event that the refund is being requested for from the drop down of current ongoing events. 4. User inputs name and email into the corresponding fields. 5. User inputs a reason for the request. 6. User submits the request for approval. |
| Alternative Course(s) | None |

3.2 INFORMATION MODEL

This subsection contains the specifications for the database to be developed as part of the software system. An Entity Relationship Diagram (ERD) will show the various data entities in the system and the relationships between those entities.

**3.2.1 Entity Relationship Diagram**



**Figure 3.2.1 Entity Relationship Diagram**

**3.2.2 Use Case Relationship Matrix**

The Use Case Relationship Matrix maps the various relationships between data entities in the ERD to the Use Cases that those relationships are designed to support.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Foreign Table** | **Cardinality** | **Supported Use Cases** |
| Member | Vehicle | 1 to Many | UC 4.11 - 4.14, UC 6.6 |
| Member | Manifest | 1 to Many | UC 3.4, UC 3.5, UC 6.5 |
| Vehicle | Manifest | 1 to Many | UC 3.4, UC 3.5, UC 6.5 |
| Employee | Vehicle | 1 to Many | UC 4.11 - 4.14, UC 6.6 |
| Employee | Manifest | 1 to Many | UC 3.4, UC 3.5, UC 6.5 |
| Employee | Forum Log | 1 to Many | UC 3.6, UC 3.7 |
| Employee | Announcement | 1 to Many | UC 5.1-5.3 |
| Track | Event | 1 to Many | UC 3.1, UC 3.2 |
| Event | Forum Log | 1 to 1 | UC 3.6, UC 3.7 |

**Table 3.2.1 Use Case Relationship Matrix**

3.3 Performance Requirements

This section details any quantitative performance requirements that may impact the final design of the system. They represent attributes the system must have to meet the sponsor’s business needs and limitations on the system that may be imposed by his available resources.

**3.3.1 User Capacity**

The number of simultaneous users supported without violating quality attributes or other performance requirements.

* Minimum: 0 users
* Maximum: 25 users

**3.3.2 Data Stores**

The number of data stores required to support the system, the initial size of the data stores, and the max rate of growth.

* Initial Size: 10 Gigabytes
* Max Growth Rate: 10 Gigabytes/Year

**3.3.3 CPU/Memory**

The minimum CPU performance and physical memory levels to be used without violating quality attributes or other performance attributes.

* CPU: A least a dual core processor.
* Memory: At least 4 Gigabytes of RAM

**3.3.4 Bandwidth**

The maximum network bandwidth (incoming and outgoing) to be used by the system during operation.

* Maximum: 1 Terabyte/Month
* Maximum Growth Rate: 1 Terabyte/Month per Year

3.4 Design Constraints

This subsection specifies the design constraints that are imposed by standards and hardware limitations. It will describe any existing standards for report formats, data naming, coding, documentation, etc. that the sponsor may have already in place.

Other than designing the project as a Web Application, there are currently no known design constraints on the GotTrackDays.com project. Since this project will be designed and built from the ground up, there are no existing standards that will affect the scope of this project. However, this document may be amended to accommodate such discovery at a later time.

3.5 Quality Attributes

This subsection contains the non-functional requirements, that is, requirements that apply to the software system being developed and not just to a single feature, function, or use.

**3.5.1 Reliability**

The software is expected to last until the number of concurrent users has reached over 25 or have up to 20,000 registered users. In addition, the system is expected to fail to respond no more than 1 out of 100 users accessing the website.

**3.5.2 Maintainability**

The client expects to expand the company outside of Northern California in five years. Therefore, the software will be designed to accommodate events that will take place outside of the initial scope. In addition, jBehaving will provide documentation for editing and expanding on the current database without compromising existing data.

**3.5.3 Program Quality Attributes**

jBehaving will be extensively documenting both the design and implementation of the software for anyone who will be maintaining or extending the existing code base. Code will formatted for readability and well-commented to help with future modifications.

Each form on the website will check to make sure that all inputs are valid. This accounts for fields with incorrect characters and required fields with no input.

The client expects that the website will not be handling any serious traffic for the immediate future, therefore the system should have little trouble serving users in a timely manner.

**3.5.4 Security**

A login is required for any feature that accesses the database and security questions will be set up in case of a forgotten password. Each user may only access features of the website that their permission level allows. Because the system is web-based, measures will be taken to protect the personal information of all users. A member’s personal information will not be seen by any member or guest unless the member allows it. No sensitive financial information (credit card information) will be stored in the database.

**3.5.5 Transferability / Conversion**

Due to being web-based, the site will be accessible by any of three web browsers: Internet Explorer, Firefox, or Chrome. The database will be stored using an off-site hosting solution and can easily be copied and moved if need be.

**3.5.6 Operational Quality Attributes**

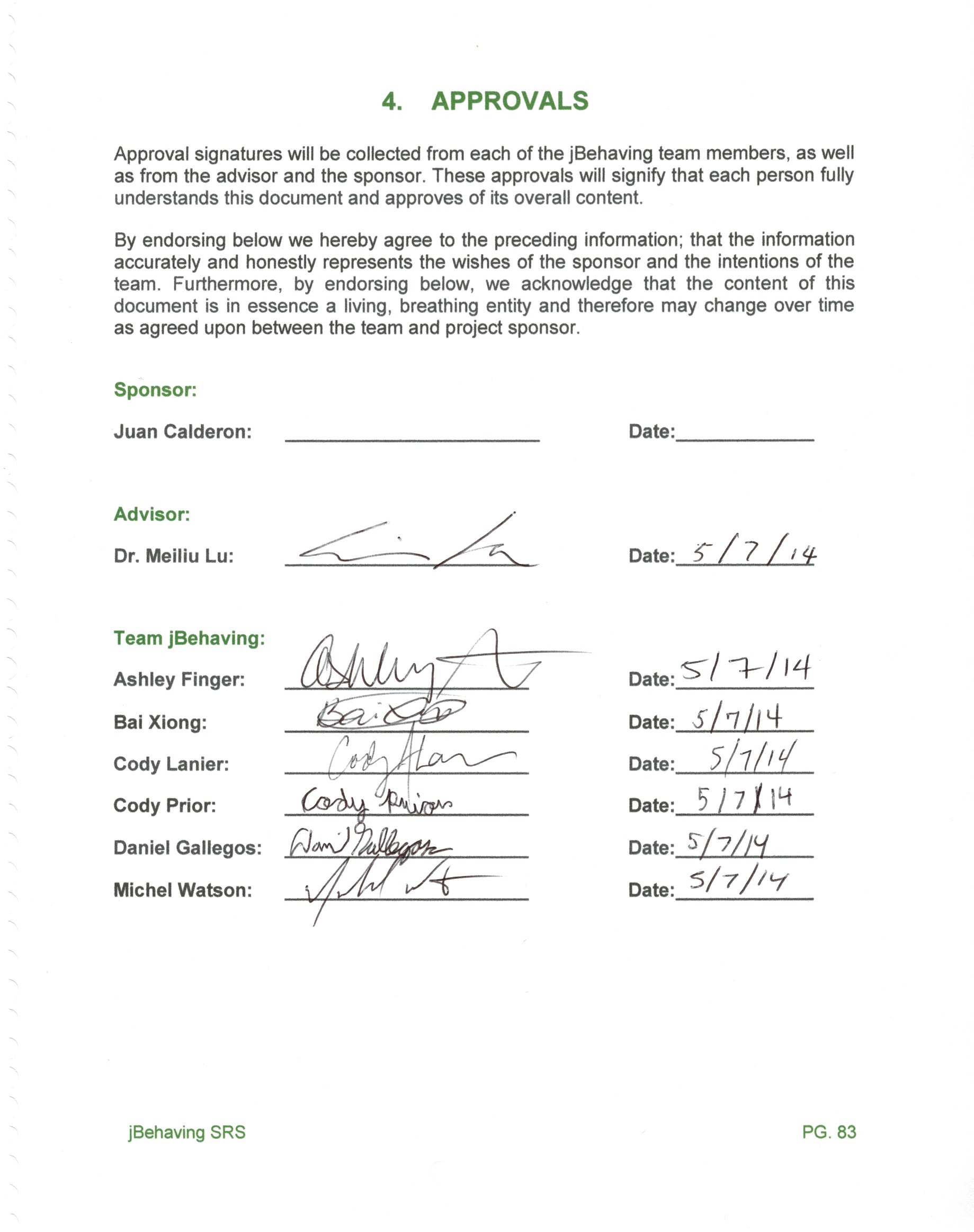
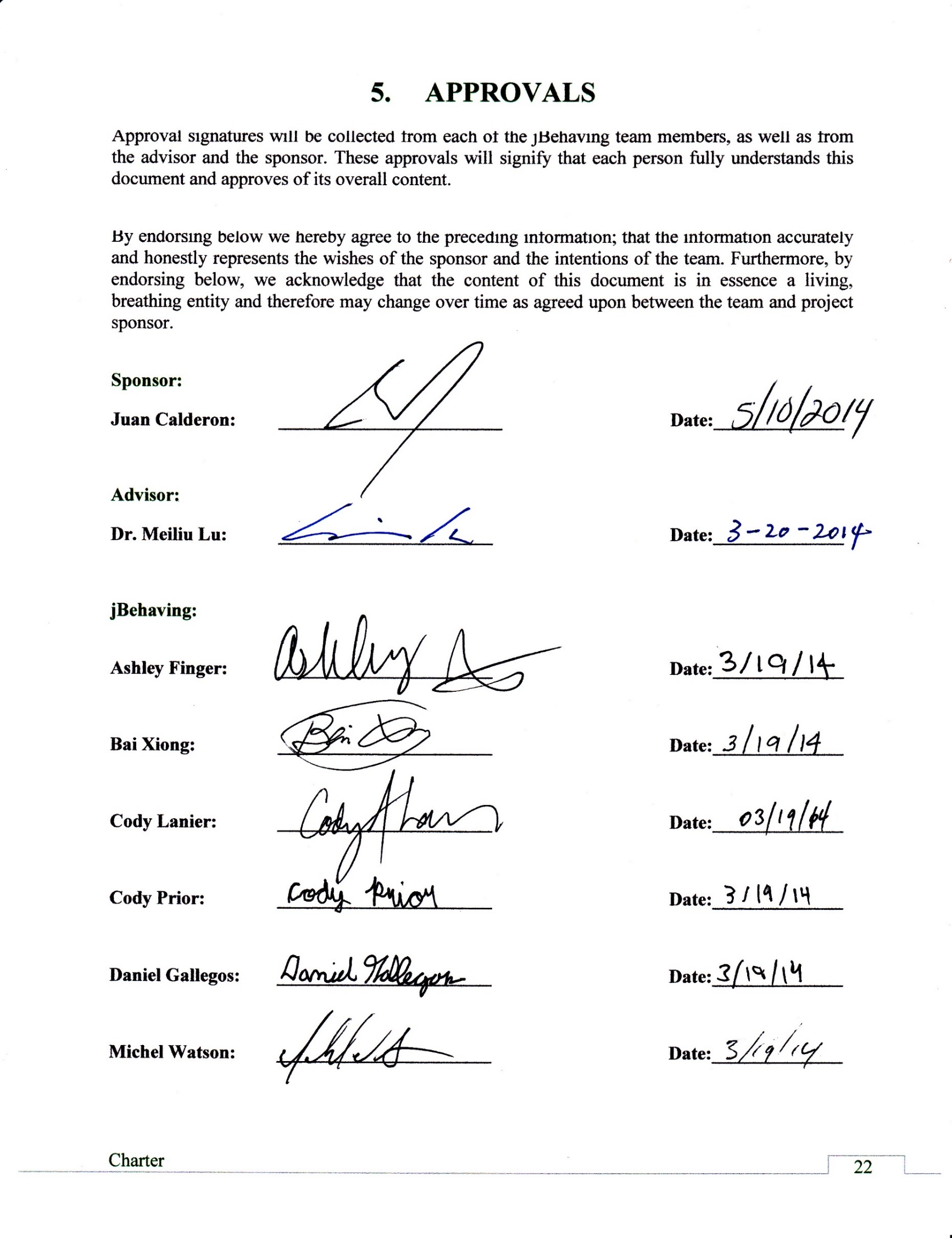
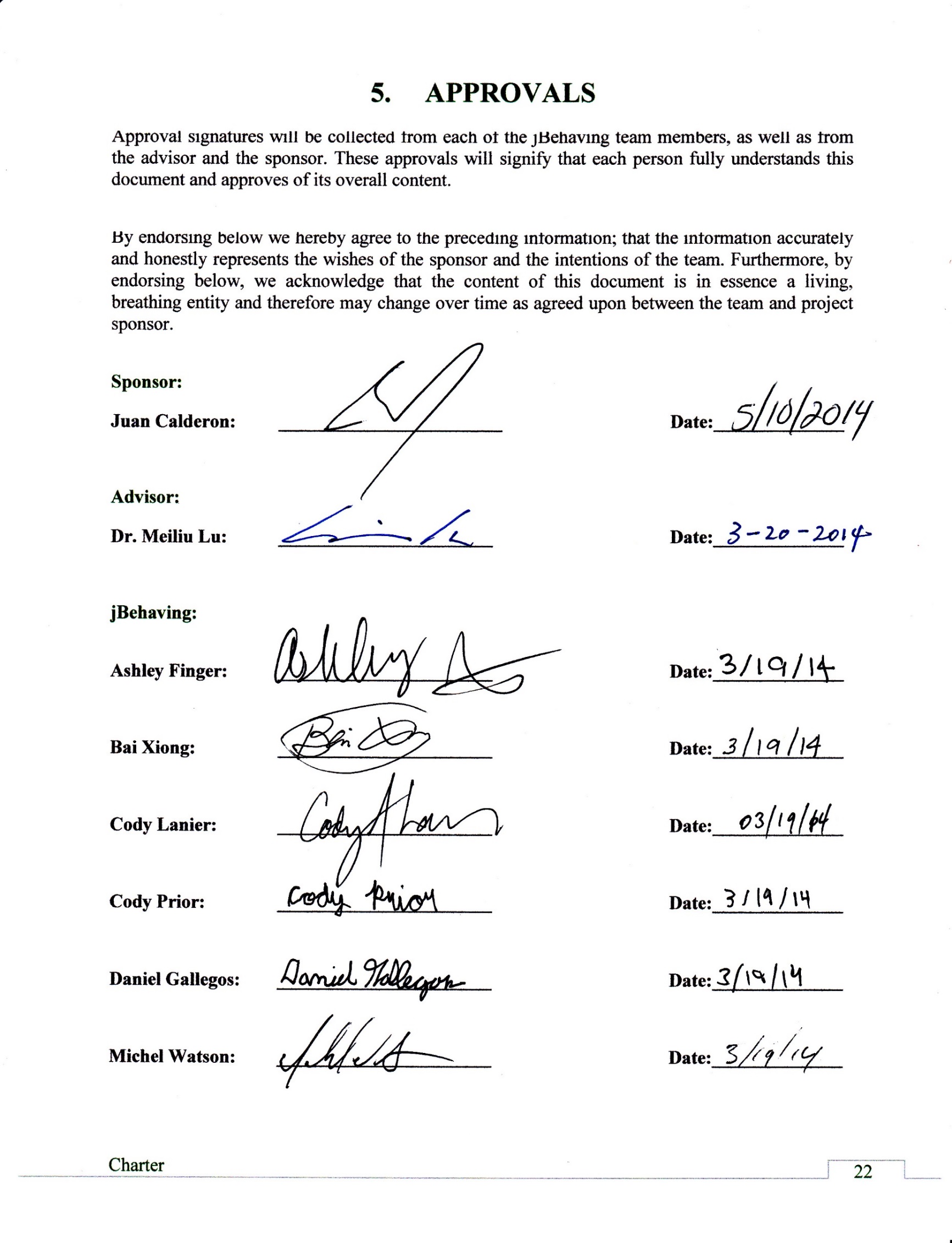
The website interface will be simple and intuitive. All pages on the website will use a similar color scheme and layout. Fields will be labeled with clear, understandable language. More complex fields will be accompanied by a brief explanation. Required fields in all forms will be clearly marked as such.

**3.5.7 Operations**

The owner will be able to access all features of the website. Other users will be limited by their permission level or user role.

**3.5.8 Site Adaptations**

Financial reports are required to output a .csv file that can be viewed in Microsoft Excel. Any user who needs to use the financial report feature will need to have a working copy of Microsoft Excel 2010 (or later preferred), OpenOffice or similar open source Office Suite.

****

APPENDIX A - Data Dictionary

ELEMENTS:

**advancedCap**

* The max size of the advanced driving class for an event.
* Set by validated user input.
* Used within event record to indicate the maximum amount of advanced drivers for a given event.
* Default value: 0.
* Valid if non negative integer.

**albumLink**

* A link to an album of photos related to the event on an external website.
* Set by user input.
* Used in event record.
* Default value: none.
* Valid if hyperlink text URL.

**amtPaid**

* The amount of money the user paid for registration.
* Set automatically after payment confirmation is received.
* Used in the registration record.
* Default value: 0.
* Valid if greater than or equal to zero, monetary format (0000.00).

**basePrice**

* The base un-discounted price in USD of registering for an event.
* Set by user input.
* Used in Registration record.
* No default value.
* Valid if greater than zero, monetary format (0000.00).

**confirmationID**

* The payment confirmation ID number.
* Automatically generated.
* Used in refund record, manifest record.
* Default value: none.
* Valid if not null.

**deactiveFlag**

* Whether or not the track is no longer in service.
* Set by user input.
* Used in track record.
* Default value: false.
* Must be a boolean value.

**decision**

* The final decision regarding whether or not a refund request was accepted.
* Set by user input.
* Used in refund record.
* Boolean value.
* Valid if not null.

**decisionReason**

* The reasoning behind the refund request decision.
* Set by user input.
* Used in Refund record.
* Text Value.
* Valid if not null.

**deletionFlag**

* A flag for whether or not the account is to be deleted.
* Set by user input.
* Used in Member record.
* Default value: False.
* Must be a boolean value.

**emergencyContactName**

* The emergency contact’s name.
* Set by user input at account registration.
* Used in employee record, member record.
* Default value: null.
* Valid if not null.

**empRole**

* Denotes the role of the employee.
* Set by user input.
* Used in the Employee record.
* Default value: none.
* Must be either: “Owner”, “StandardEmployee”, “EventManager”, “Administrator”, or “Accountant”

**eventDesc**

* A short text description of the event.
* Set by user input.
* Used within the event record.
* Default value: “ ”.
* Valid if not null.

**eventEndDate**

* The date at which the event ends.
* Set by validated user input.
* Used to store the ending date of an event.
* Default value: “00/00/00”
* Any date in standard date format that is today or greater.
* Valid if not null, valid date format. Cannot be before eventStartDate.

**eventEndTime**

* The time of day at which the event ends.
* Set by validated user input.
* Used to store the end time of an event.
* Default value: “00:00”.
* Valid if not null, must be valid time format (Any date in 24 hour time format that is now or greater).

**eventID**

* A unique, automatically generated number that identifies the event.
* Set during creation of an event record.
* Used to link an event to another entity.
* Included in no Data Structures.
* Default value: none.
* Valid if not null, unique.

**eventStartDate**

* The date at which the event starts.
* Set by validated user input.
* Used to store the beginning date of an event.
* Default value: “00/00/00”.
* Any date in standard date format that is today or greater.
* Valid if not null, valid date format.

**eventStartTime**

* The time at which the event starts.
* Set by validated user input.
* Used to store the begin time of an event.
* Default value: “00:00”.
* Any date in 24 hour time format that is now or greater.
* Valid if not null, valid time format.

**inactiveFlag**

* Whether or not the employee account is inactive.
* Set by user input.
* Used in employee record.
* Default value: false.
* Must be a boolean value.

**intermediateCap**

* The max size of the intermediate driving class for an event.
* Set by validated user input.
* Used within event record to indicate the maximum amount of intermediate drivers for a given event.
* Default value: 0.
* Valid if non negative integer.

**newsTitle**

* The title text of the announcement.
* Set by user input.
* Used in announcement record.
* Default value: null.
* Valid if not null.

**newsBody**

* The body text of the announcement.
* Set by user input.
* Used in announcement record.
* No default value.
* Valid if not null.

**newsEditedDate**

* The date that the announcement was edited.
* Set automatically when changes are accepted.
* Used in the announcement record.
* Default value 00/00/00.
* Must be in format “mm/dd/yyyy”.

**newsEditedTime**

* The time that the announcement was edited.
* Set automatically when changes are accepted.
* Used in the announcement record.
* Default value 00:00.
* Valid if not null.

**newsEnteredDate**

* The date at which the announcement was added.
* Set automatically when an announcement is entered.
* Used in announcement record.
* Text Value.
* Must be in format “mm/dd/yyyy”.

**noviceCap**

* The max size of the novice driving class for an event.
* Set by validated user input.
* Used within event record to indicate the maximum amount of beginners for the event.
* Default value: 0.
* Valid if non-negative integer.

**password**

* The password of the user
* Set by user input
* Used in employee record, user record.
* Default value: none.
* Valid if not null, hashed value.

**personalEmail**

* The user’s personal email address.
* Set by user input at account registration.
* Used in member record, employee record.
* Default value: none.
* Must be in format: “X@X.X”.

**primaryEmergencyPhoneNumber**

* The primary emergency phone number for the user.
* Set by user input at account registration.
* Used in Member record.
* Default value: none.
* Must be in format (XXX) XXX-XXXX.

**primaryPhoneNumber**

* The primary phone number for the internal user.
* Set by user input at account registration.
* Used in user record.
* Default value: none.
* Must be in format (XXX) XXX-XXXX.

**rideAlong**

* Whether or not the user is willing to ride alongside another driver.
* Set by user input at event registration.
* Used within manifest record.
* Default value: False.
* Must be a boolean value.

**riderWanted**

* Whether or not the user wants someone to ride alongside them.
* Set by user input at event registration.
* Used within manifest record.
* Default value: False.
* Must be a boolean value.

**secondaryEmergencyPhoneNumber**

* The secondary emergency phone number for the internal user.
* Set by user input.
* Used in Member record.
* Default value: none.
* Must be in format (XXX) XXX-XXXX.

**secondaryPhoneNumber**

* The secondary phone number for the internal user.
* Set by user input.
* Used in Employee record.
* Default value: none.
* Must be in format (XXX) XXX-XXXX.

**skillClass**

* The skill level at which the user is registered.
* Set by employee user during event registration.
* Used in registration record.
* Discrete values: “novice”, ”intermediate”, or “advanced” .
* Valid if not null.

**state**

* The state where the user lives.
* Set by user input at registration.
* Used in member record, employee record
* Text value
* Must be a valid state within the US.

**streetAddress**

* The street number and name of a user’s home address.
* Set by user input at registration.
* Used in member record, employee record.
* Text value.
* Must be a valid street address.

**trackID**

* The unique identifier assigned to each distinct track in the system.
* Set automatically during creation.
* Used in the Track and Event records.
* Included in no Data Structures.
* Default value: none.
* Valid if not null, integer.

**trackName**

* The official name of the track.
* Set by user input.
* Used in Track record.
* Default value: none.
* Valid if not null.

**trackAddress**

* The physical address of the track.
* Set by user input.
* Used in Track record.
* Default value: none.
* Valid if not null, address format.

**trackContactName**

* The primary contact’s name.
* Set by user input.
* Used in track record.
* Default value: null.
* Valid if text.

**trackPhoneNumber**

* The primary phone number for the track.
* Set by user input.
* Used in track record.
* Default value: null.
* Valid if phone number format

**userActiveFlag**

* A flag indicating the active or inactive status of a given user.
* Set programmatically and through user intervention.
* Used in Member and Employee record.
* Default value: “Y”.
* Discrete list of values: “Y”, “N”.

**userBirthday**

* The birthdate of the user.
* Set by user input.
* Used in the Member record.
* Default value: none.
* Must be in format (mm/dd/yyyy).

**userCreatedDate**

* The date that the user account was created.
* Set automatically when user account is created.
* Used in the Member and Employee records.
* Default value: none.
* Valid if not null.

**userCreatedTime**

* The time that the user account was created.
* Set automatically when user account is created.
* Used in the Member and Employee records.
* Default value: none.
* Valid if not null.

**userFirstName**

* The common name of the user.
* Set by user input.
* Used in the Member and Employee record.
* Default value: none.
* Valid if not null.

**userLastName**

* The surname of the member.
* Set by user input.
* Used in the Member and Employee records..
* Default value: none.
* Valid if not null.

**vehicleColor**

* The color designation for the hue of the vehicle.
* Set by user input.
* Used in Vehicle record.
* No default value.
* Discrete list of values: “Red”, “Orange”, “Yellow”, “Green”, “Blue”, “Purple”, “White”, “Black”, “Brown”, “Grey”, “Pink”, “Silver”.

**vehicleEnteredDate**

* The date that the car was created.
* Automatically generated on day of creation and set when the car is record is created.
* Used in Vehicle record.
* No default value.
* Valid if not null, valid text in date format (mm/dd/yyyy).

**vehicleID**

* Identifier for the car which is to be used in the event.
* Automatically generated at car creation.
* Used in registration and car records.
* No default value.
* Valid if not null, unique.

**vehicleModel**

* The model designation of the car.
* Set by user input.
* Used in Car record.
* Default value: Null.
* Discrete list of values depends on make.

**vehicleMake**

* The Make describing the car.
* Set by user input.
* Used in Car record.
* Default value: null.
* Discrete List of Values: common car makes.
* Valid if not null.

**workEmail**

* An employee’s work email address.
* Set by user input.
* Used in employee record.
* Default value: none.
* Must be in format: X@X.X.

**zip**

* The ZIP code of the user.
* Set by user input.
* Used in member record, employee record.
* Integer Value.
* Must be in format XXXXX or XXXXX-XXXX.

TABLES:

**Event**

Description: A table containing all the information pertaining to each event. It is used to inform prospective customers about upcoming events.

How it is set: User input via the “Add Event” page.

How it is used: Search Event, View Event.

Number of Records: One record per event.

Expected Record Growth: 15 records for the first year, growing to about 150 new records by year five.

Composition:

* advancedCap
* albumLink
* basePrice
* eventDesc
* eventEndTime
* eventID
* eventStartDate
* eventStartTime
* noviceCap
* trackID

**Manifest**

Description: A table containing details about individual event registrations.

How it is set: User input via the “Register for event” page.

How it is used: Register for Event, View Event, and Generate Financial Report.

Number of records: Multiple records per user, up to the maximum event capacity for each unique event.

Expected Record Growth: Approximately 40-120 per event (In five years this will mean 6000 – 18000 registrations a year).

Composition:

* amtPaid
* conformationID
* eventID
* rideAlong
* riderWanted
* skillClass
* userID
* vehicleID

**Member**

Description: A table containing all the information pertaining to each member. It is used to build a profile for the user as well as linking their registrations to the account.

How it is set: Created when a new member registers, updated through user input at the “Profile” page.

How it is used: View Profile

Number of records: One record per member.

Expected Record Growth: Expected upper bound of 20,000 in five years.

Composition:

* city
* deletionFlag
* emergencyContactName
* password
* personalEmail
* primaryEmergencyPhoneNumber
* primaryPhoneNumber
* secondaryEmergencyPhoneNumber
* secondaryPhoneNumber
* state
* streetAddress
* userCreatedDate
* userFirstName
* userID
* userLastName
* ZIP

**Employee**

Description: A table containing all the information pertaining to each employee. It is used to build a profile and designate what website features an employee can access.

How it is set: User input via the “Accounts” page

How it is used: View Profile

Number of records: One record per employee.

Expected Record Growth: Very Low, less than 50 records in the first five years.

Composition:

* city
* inactiveFlag
* emergencyContactName
* password
* personalEmail
* primaryEmergencyPhoneNumber
* primaryPhoneNumber
* secondaryEmergencyPhoneNumber
* secondaryPhoneNumber
* state
* streetAddress
* userCreatedDate
* userFirstName
* userID
* userLastName
* workEmail
* ZIP

**Vehicle**

Description: A table containing all information about member/employee cars.

How it is set: User input via the Garage section in the “Profile” page.

How it is used: View Garage, View Event.

Number of records: Multiple records per user

Expected Record Growth: Slightly higher than the member record growth rate.

Composition:

* userID
* vehicleID
* vehicleModel
* vehicleMake
* vehicleColor

**Announcement**

Description: A table containing all information about news announcements. It is used to communicate deals, changes, and upcoming events to customers on the front page.

How it is set: User input via the “Announcements” section on the “Home” page.

How it is used: GotTrackDays.com Front page.

Number of records: Multiple records per employee.

Expected Record Growth: Most likely low, depending on frequency of offers and updates.

Composition:

* newsBody
* newsEditedDate
* newsEditedTime
* newsEnteredDate
* newsTitle
* userID

**Forum Log**

Description: A table containing the details of each message written by employees. These messages are used to communicate employee activities on various forums to advertise an event.

How it is set: User input.

How it is used: View event

Number of records: Multiple per employee, per event

Expected Record Growth: Medium, no more than 20 messages per event.

Composition:

* comment
* dateEntered
* eventID
* forumLink
* userID

**Refund**

Description: A table detailing each refund request.

How it is set: User Input via “Request Refund”.

How it is used: Used in the processing requests.

Number of records: Multiple per event.

Expected Record Growth: Low, anywhere from zero to two per event.

Composition:

* amtPaid
* conformationID
* decision
* decisionReason
* eventID
* requestReason
* userID

**Track**

Description: A table detailing each individual track.

How it is set: User Input.

How it is used: Add Event.

Number of records: One per unique track.

Expected Record Growth: Low

Composition:

* deactivatedFlag
* trackAddress
* trackContactName
* trackPhoneNumber
* trackName