Mibrary

Technical Report

Contents

Motivation

The motivation of the Mibrary project is to provide an avenue for students to donate or exchange their textbooks. This will help students who have a financial barrier to their education, and in turn be of community service.

User Stories

User Story 1: There are many textbooks provided by our traders. When a user needs to know what textbooks we offer, the application displays the list of the textbooks exist in the system. Assuming that the user does not give any filter criteria for listing the textbook, so all of the textbooks will be shown.

Estimate time: 1 hour

Actual time: 1 hour

User Story 2: When a user clicks on a textbook from the textbook list displayed on the screen, the system will show the user information of that textbook includes its ISBN number, title, author, edition, and cover photo.

Estimate time: 1 hour

Actual time: 1 hour

User Story 3: Users are able to know what classes they can take at UT and information about the classes such as the classes' numbers, titles, and required textbooks. The system will display the list of classes and users can get any class's information by clicking on the class from the displayed list. Assuming users do not give any filter criteria for listing available classes.

Estimate time: 1 hour

Actual time: 1 hour

User Story 4: Users are able to know who is trading (donating or requesting) textbooks across the system and which textbooks are corresponding to each trader. Assuming traders are human.

Estimate time: 1 hour

Actual time: 1 hour

User Story 5: Users are able to know the methods to contact traders via contact information that traders post in the system. Users can get any trader's contact information when they click on the trader from the displayed user list.

Estimate time: 1 hour

Actual time: 1 hour

RESTful API

The parameters of the API design requirement were vague and ambiguous at first. It was unclear if we were required to map the API calls that we were consuming to build our database, or the ones that will ultimately be provided to the front end and other users of Mibrary, or both. Once the requirements were cleared up, this part became relatively simple.

To begin, the Mibrary API contains a couple of get requests for each of our models: Book, Review, Meeting, User, and Class. These requests return JSON files with the attributes of the specific instance of the model requested.

For Book, the API provides access to getting a single book by ISBN or query, getting the location of a book, and getting a list of all books. The list of books returns a list of ISBN numbers and names. The individual book returns the name, ISBN number, locations it can be found, and users with requests on the book.

For Review, the API provides the ability to get a review of a given book, taking ISBN as a parameter. The review returns a description, rating, and username.

For Meeting, the API provides the ability to get the time of a meeting and get the location of a meeting. Both of these can be found based on the user involved in the meeting. The meeting returns a time, location, and the users involved.

For User, the API provides the ability to get a single user by username, or a list of all users. The user returns the name, list of involved meetings, list of reviews, and books currently borrowed.

For Class, the API provides the ability to get a single class by course number, or all courses by institution. The class contains the class name, class number, and list of required textbooks.

The Mibrary API is available here:
https://documenter.getpostman.com/view/4703640/RWEjqJFc

Models

- Book

    - Book or textbook available at the library
    - Name
    - ISBN number
    - Locations it can be found
    - Users requesting book
- Review
    - Book name
    - ISBN Number
    - Description
    - Rating
    - Username
- Meeting
    - Time
    - Location
    - Username
    - Book(s) being exchanged
- User
    - Username
    - List of meetings
    - List of reviews
    - List of books being borrowed
- Class
    - Class name
    - Class number
    - List of required textbooks

Tools

Postman

Postman is the tool used to create documentation for the Mibrary API. Although Postman was a complicated tool in the beginning, its usefulness became apparent after working with it for longer. Postman allows us to easily document a restful API through creating and categorizing requests. These requests include the type, URL, description including sample usage, and parameters. A sample request can also be created and shown in a JSON or XML format.

Bootstrap

Bootstrap is the JavaScript and CSS framework that we used to enhance the user experience of our website. In order to implement Bootstrap, we integrated small pieces of scripting code into our html files. This code provides the link to a “CDN,” which is essentially a server with Bootstrap installed. Applying Bootstrap to all of our web pages ensured that the fonts were aesthetically pleasing and easy-to-read. Furthermore, we were better able to organize our information.

Slack

Slack has proven essential for communication. Our entire team is available on a single slack channel, connected to our phones and computers, allowing us to be in constant communication. Through this mechanism, we are able to communicate, meet up, ask clarifying questions, and coordinate the shared assignment and division of tasks.

Git

Git has been an invaluable tool to allow multiple contributions to this project. We have set up a shared Mibrary repo with CI. We commit to the repo often so we can always be working with the latest code. This allows all of us to update the same website codebase frequently, whenever individually we see a change that should be made.

Hosting

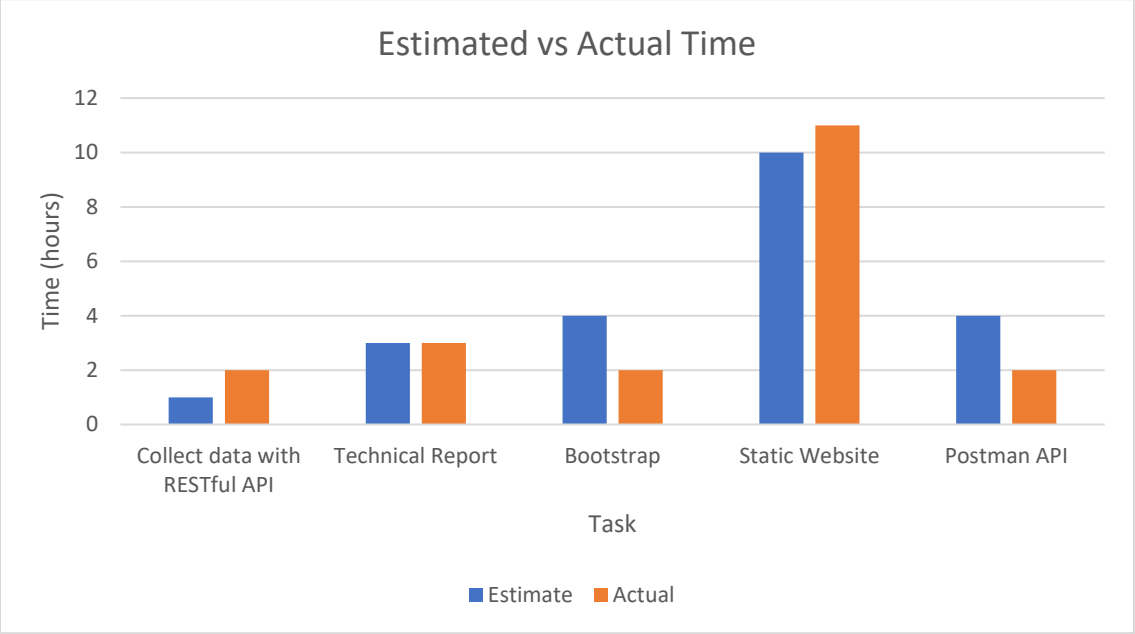
Namecheap

Our domain name was acquired from Namecheap. To use this domain with Amazon hosting, we used Namecheap's functionality to change the CNAME record to map our domain to the domain provided to us by AWS.

AWS

To host our static website, we used Amazon's AWS S3 service, a cloud-based static content storage service. There was some initial confusion regarding how to navigate Amazon's AWS service, but following the step by step process found online here was very helpful. After registering an account with AWS, the S3 console can be found at https://console.aws.amazon.com/s3/. We created buckets for both the mibrary.me and www.mibrary.me domains, and uploaded the same files to both. When uploading files, all the permissions must be set to public readable by everyone. Finally, to enable website hosting, static website hosting must be enabled in the properties of each bucket.

Charts


Estimated vs Actual Time
12
10
8
6
4
2
0
Time (hours)
Collect data with RESTful API
Technical Report
Bootstrap
Static Website
Postman API
Task
Estimate
Actual