FindACarFor.me

Group 7 - 11 AM

Introduction

Varun Jawarani
Front-End
Developed front-end tests and worked on front-end aspects of API requests. Wrote technical reports and handled code documentation.

Jatin Kulkarni
Full-Stack
Created Models, Model Instances, and Search Page. Worked on sorting, filtering, searching, and developer visualizations. Created scripts to populate our database in the backend from the various APIs

James Stuedemann
Backend
Created Backend with calls for searching, filtering, sorting, and pagination. Backend Postman and Testing. DevOps for Amplify and EB.

Trent Ho
Front-End
Created about page that dynamically updates. Worked on front-end design with react and CSS.

Taqi Hossain
Front-End
Created tests for front-end using Jest and splash page cards and banner. Worked on visualizations for our developer, ExploreAndGiveMore.

Our Website

https://developer.d7a6xirwxpml0.amplifyapp.com/

Our Postman

https://documenter.getpostman.com/view/23628441/2s83tJGqha

01

Self Critique

Self Critique

1. What did we do well?

- Back-End Development
- Testing
- Troubleshooting

2. What did we learn?

- The “Stack”
- Pipelines
- Product Design

3. What did we teach each other?

- Individual Skills
- Delegation
- Communication

Self Critique

4.

What can we do better?

- Time Management
- Task Organization
- Efficiency

5.

How were Peer Reviews?

- Accountability
- Direction

6.

What puzzles us?

- API Access
- Front-End UI

What did we do well?

1.

Back-end

- Client-side API functions neatly
- Database was designed for efficiency

2.

Testing

- Testing is thorough and future-proof

3.

Troubleshooting

- Communicated well when debugging
- Teamwork was strong when necessary

What did we learn?

1. The “Stack”

- How data moves across the stack
- The full process of API requests

2. Pipelines

- How to manage Docker images
- CI stages and test validation

3. Product Design

- Planning features fully before implementation
- Efficient database design

What did we teach each other?

1. Individual Skills

- React basics
- Working with the React-Bootstrap Library
- AWS Experience

2. Delegation

- Task management across a team
- How to give and ask for support

3. Communication

- Communicated well when debugging
- Teamwork was strong when necessary

What can we do better?

1. Time Management

- Didn't estimate time for tasks accurately
- Tasks assigned were not always equal

2. Task Organization

- Rubric fulfillment
- Issue tracking and GitLab boards usage

3. Efficiency

- Both in design and in programming
- Could certainly be more 'Agile'
- Further refactoring could be done

What effect did the peer reviews have?

1.
Accountability

- Allowed us to talk address any qualms we had
- Opened conversation for changes

2.
Direction

- Influenced what features we decided to include
- Motivated us when we achieved consumer goals

What puzzles us?

1.
API Access

- Automotive industry loathes student projects
- Certain features were missing or kept behind paywall

2.
Front-end UI

- Some clickable elements displayed/functioned improperly

02

Customer Critique

Customer Critique

1.

What did they do well?

- Very pretty UI/UX
- Intuitive website flow
- Info/sources were organized well

2.

How was their API?

- Well-documented request info
- Functioned exactly as documented

3.

Implementing user stories?

- Implemented nearly all effectively
- Most were within scope and were promptly replied to

Customer Critique

4.
What did we learn?

- The number of valuable charities in a city is not strictly correlated with the budget someone needs to live there or its safety rating

5.
What can they do better?

- Attributes could be located next to ratings for Attractions
- Charities and Attractions could follow the same layout

6.
What puzzles us?

- Some of the ratings aren't 'cited'
- Some of the information for cities is better suited for living permanently

Customer Critique

- **What did they do well?**
  - The UI/UX of their website was very nice
  - Overall website was very intuitive to use
  - Adjacent info/sources were organized well
- **How effective was their RESTful API?**
- **How well did they implement your user stories?**
  - They were able to implement nearly all of the user stories and the features function cleanly
- **What did we learn from their website?**
  -
- **What can they do better?**
  - Search feature
  - Duplicate instances/wrong images on instances
- **What puzzles us about their website?**

What did they do well?

1.

UI/UX

- Overall website had a clean layout and design with smooth functionality

2.

It’s Intuitive

- The use of tags made the website intuitive to use

3.

Adjacent Info

- Adjacent information such as charity websites, apartment finders, etc. were integrated well with each instance

How effective was their RESTful API?

1.

Back-end

- Client-side API functions neatly
- Database was designed smoothly

2.

Testing

- Testing is thorough and future-proof

3.

Troubleshooting

- Communicated well when debugging
- Teamwork was strong when necessary

How were our user stories implemented?

1.

Filter by category

- Searching for charity by broad category/terms

- Filtering by state

2.

Ranking system on instance pages

- Walking, Transit, Bike scores

What did we learn from their website?

1. Back-end

- Client-side API functions neatly
- Database was designed smoothly

2. Testing

- Testing is thorough and future-proof

3. Troubleshooting

- Communicated well when debugging
- Teamwork was strong when necessary

What can they do better?

1. Search

- Search feature is buggy when user inputs space characters

2. Duplicate Instances

- Some instances are either duplicates or have the same image as a different city with same name

3. Troubleshooting

- Communicated well when debugging
- Teamwork was strong when necessary

What puzzles us about their website?

1. Ratings

- Are the average ratings for cities based on user ratings?
- Some information about what the ratings (perhaps in a popover) are based on would be helpful

2. Testing

-

3. Troubleshooting

-

Thank you

Any questions