ParentAidATX Phase 1 Technical Report

Group Members: Amna Ali, Andrew Harvey, Rubi Rojas, Allison Still, Ethan Yu

Website Link: https://parentaidatx.me

CS373: Software Engineering
University of Texas at Austin

1. **Introduction**
    a. **Purpose**
        i. ParentAidATX empowers single parents in Austin, Texas, by simplifying access to essential resources, including family-related government assistance, affordable housing, and childcare services. Navigating support systems can be overwhelming, especially for single parents balancing work and family. ParentAidATX bridges this gap by connecting users to tailored programs and services, making it easier to understand eligibility, find nearby affordable housing, and locate quality childcare. We believe every parent deserves support, and every child deserves stability.

    b. **Questions Our Website Will Answer**
        i. What single-parent government programs am I eligible to apply for?
        ii. Where can I find affordable housing near my child's daycare or school?
        iii. What are the best-rated childcare centers that fit my budget?
        iv. What government assistance is available for housing and childcare in Austin?

2. **Models and Instances**
    c. **Model 1: Government Services**
        i. **Instances:** ~50
        ii. **Attributes**
            1. Program ID
            2. Program Name
            3. Government Administration/Department
            4. Government Scope (City, State, Federal)
            5. Category (Food, Housing, etc.)
            6. Description
            7. Application Link
        iii. **Media**
            1. Links
            2. Text descriptions of programs
            3. Government department image
        iv. **Connections to Other Models**
            1. Housing: Displays local programs based on location
            2. Affordable Childcare: Government programs related to childcare
    d. **Model 2: Housing**
        i. **Instances:** ~1000
        ii. **Attributes**
            1. Name
            2. Cost/Rate
            3. Rating
            4. Reviews
            5. Address
            6. Zip Code
            7. Style of Housing (Apartment, Condo, House)
            8. Crime Level
            9. Nearby Park
            10. Transportation
            11. Government Subsidized (Or Not)
        iii. **Media**
            1. Google Maps for location

2. Location Image
3. Text description of location, cost, and type of housing
4. User Reviews

iv. Connections to Other Models

1. Government Programs: Government programs pertaining to housing
2. Affordable Childcare: Displays affordable housing near the location of schools/daycares

e. Model 3: Childcare

i. Instances: ~8000

ii. Attributes

1. Name
2. Type of Childcare (Daycare, Afterschool, etc.)
3. Cost
4. Quality Rating
5. Age Range of Children
6. Zipcode
7. Address

iii. Media

1. Google Maps for location
2. Image of Childcare Location
3. Link to Childcare Website

iv. Connections to Other Models

1. Government Programs: Childcare provided by government programs
2. Housing: Affordable housing near the location of schools/daycare

3. **User Stories**
    f. “Add child care connection to housing page” - Sana Kohli
        i. We display affordable housing near the location of schools/daycares.
    g. “Add park as a housing attribute” - Sana Kohli
        i. We added nearby parks as an attribute that the user can search for on the housing page.
    h. “Add government program connection to housing” - Sana Kohli
        i. We added a connection that displays government programs that relate directly to housing.
    i. “Add public transportation as a housing attribute” - Sana Kohli
        i. We added transportation as an attribute that the user can search for on the housing page.
    j. “Crime level housing attribute” - Sana Kohli
        i. We added crime level as an attribute that the user can search for on the housing page.

4. **API Documentation**
    a. **GitLab API**
        i. The GitLab API enables us to display real-time data on commits, closed issues, and unit tests created by each site maintainer. We fetch issues via the https://gitlab.com/api/v4/projects/[PROJECT_ID]/issues endpoint. For each person, we count the issues assigned to them that are currently closed. Commits are retrieved through the https://gitlab.com/api/v4/projects/[PROJECT_ID]/repository/commits endpoint, where each commit associated with a person's email address contributes to their commit count.
        ii. Since GitLab's API support for unit tests is limited, tracking the correct number of unit tests per person involves a partially manual process. Contributors include the phrase "n unit tests" in their commit messages to denote the number of tests added. Each such message increments the contributor's unit test count accordingly.
        iii. All API calls to GitLab are made asynchronously. Given GitLab's limit of 100 items per page, we repeat API calls for each subsequent page, using the page query parameter to specify the desired page of data. We set the per_page parameter to 100 to maximize the amount of data received per call. For example, retrieving the second page of issues would involve a request to https://gitlab.com/api/v4/projects/[PROJECT_ID]/issues?per_page=100&page=2.

    b. **RESTful API**
        i. We have integrated Google Maps to provide location-based services and interactive maps, enhancing user navigation. Additionally, the Yelp API fetches and displays reviews, ratings, and detailed information about local businesses and services relevant to parents, helping them make informed decisions based on real user feedback.

c. **ParentAidAtx API**
    i. Our API utilizes GET calls to retrieve information about all models on the website, typically based on their attributes. This functionality allows users to either obtain all instances of a model at once or find a specific instance, depending on the nature of the call. For more detailed information, please refer to the API documentation available at https://documenter.getpostman.com/view/42442568/2sAYdZstBv.

5. **Tools**
    a. **Node.js:** Used for package management. Run ‘npm install’ to install the required dependencies from the package.json file.
    b. **React:** Frontend framework for building user interfaces
    c. **Bootstrap:** CSS framework for responsive styling and UI components
    d. **AWS Amplify:** Website hosting
    e. **Postman:** API Documentation

6. **Hosting**
    a. Our website is hosted and deployed with AWS Amplify. AWS Amplify ensures a stable user experience and allows our site to scale. Furthermore, with AWS Amplify, the website will automatically update when a push is successfully made to GitLab. The domain parentaidatx.me was obtained from Namecheap.

7. **Architecture**
    a. **System Components**
        i. **Frontend**
            1. The frontend of ParentAidATX is built using React, HTML, CSS, JavaScript (JSX), Node.js, and Bootstrap. Each page features a navigation bar created by the App.jsx and main.jsx files. All frontend files are located in the frontend folder.
        ii. **Backend**
            1. The system leverages AWS services, including AWS Amplify for serverless backend and database management. This ensures scalability and robustness of the platform.
        iii. **CI/CD Pipeline**
            1. The project is integrated with GitLab's CI/CD pipeline, allowing for continuous integration and deployment. This setup ensures that any code changes are automatically tested and deployed.

    b. **Folder Structure**
        i. **Frontend Folder**
            1. Contains all the frontend files, including the main components (App.jsx, main.jsx) that create the navigation bar.
            2. **frontend/public:** Stores all images used in the application
            3. **frontend/models:** Stores all the models utilized by the frontend components.

    c. **Data Attributes**
        i. Each instance within the system has five attributes and includes at least two sets of media, ensuring comprehensive data representation.

    d. **Data Flow**

i. Data flows seamlessly through the system, from collection and processing on the frontend, to storage and retrieval on the backend. AWS Amplify ensures efficient data handling and synchronization across the platform.

e. Communication Protocols

i. The system primarily uses REST APIs for communication between the frontend and backend components, ensuring smooth data exchange and interaction.

f. Deployment

i. The system is deployed and maintained using GitLab's CI/CD pipeline, enabling continuous integration and deployment. This ensures that updates and new features are rolled out seamlessly without disrupting the user experience.

8. **Challenges**
    a. **Hosting**
        i. One challenge we had with AWS Amplify was getting the routing configuration correct. Originally, we had an HTTP ERROR 404, meaning no webpage was found at the web address. Upon some more investigation, we realized AWS was looking for a build output file in the frontend directory of our repository. However, since we are using Vite + React, Vite outputs the build to frontend/dist/ by default. So, we resolved this issue by updating the base directory in amplify.yml by specifying baseDirectory: frontend/dist.
    b. **React**
        i. We were not familiar with React, specifically React components and JSX, so we had to spend some time learning it. However, in just a short period of time we were able to accomplish a lot and learn how to utilize React's powerful components which made building the website simpler and the code more readable and modular.