City Surf

CST-451 Capstone Project Final Architecture & Design

Scott Maxwell

Grand Canyon University

Instructor: Professor Michael Landreth

Revision: 1

Date: August 4th 2024

ABSTRACT

CitySurf provides an interface in the form of a web application to view various statistics about different cities in the United States. To create this experience, a full stack application will be created to provide a feature rich experience while having a minimal non-overwhelming interface to interact with the data. The goal is to provide a more consumable data visual tools that draw the eye to the conclusion.

This document summarizes the overall project design in how the application is built and implemented. The backend is built with an Express app that will be hosted on AWS while the user facing site will be built with React. The code for both will be stored in a Github repository where CI/CD methods can be used to automatically deploy when a commit is made impacting the respective directory. The city data is made up of Redfin Housing, Census Bureau, and NOAA weather data. The city data is hosted on MongoDB Atlas and delivered from the express application. The data is processed via scripts to be formatted and combined for each city before being uploaded to MongoDB.

This document specifies that the application will utilize various technologies such as Tailwind for CSS, Mapbox API for displaying an interactive map, and plotlyJS for displaying graphs. The applications are hosted in a mono repository where a commit to the main branch will trigger a deployment of the update to the hosted Amazon Web Services application. The issues found will also be reported via the "issues" area in the Github repository to keep track of the product backlog such that associated branches of code can be created to solve the found issues.

History and Signoff Sheet

Change Record

Date	Author	Revision Notes
08/01/2024	Scott Maxwell	Initial draft for review/discussion

Overall Instructor Feedback/Comments

Overall Instructor Feedback/Comments

Integrated Instructor Feedback into Project Documentation

☐ Yes ☐ No

TABLE OF CONTENTS

Design Introduction

Proposed solution: There is a lack of an easy-to-use application that provides consumable metrics for cities in the United States along with real community ratings for each city. CitySurf provides a way to view statistics for up to three different cities simultaneously.

Functionality: The user can add different cities via a search field or select them from an interactive map. If the user decides to create an account, they can save these cities to their own personal list to be viewed later. When they login, they also see additional bonus community metrics that are otherwise non viewable. When the user deselects (or removes) a city, the graphs will reload without the statistics of the loaded city. The user can add cities on the main city view page and the diagrams will update accordingly.

Data flow: When the user navigates to the city stat page, each city selected will result in an API call to the express application to retrieve the data for each city. The diagrams will then be created when the data returns from each API call.

Login: The user will be able to login. When doing so, the credentials are sent to the backend server, and a randomly generated CRSF token is saved to the client's JavaScript and the session Id is saved as a cookie. Both are sent when forms are submitted to the backend to perform an update or delete action.

Sign Up: Each user who signs will be prompted to rate their own city which will be retrieved via the Geolocation Web API. If the user blocks permission, the user will be allowed to skip this step. The user also must be verified with Altcha (a Captcha alternative) to ensure they are not robots. After they create the user, a user session and CSRF token will be created and the Modal used to fill out the form(s) will close automatically. See a flow chart here

Password Reset: If the user forgets their password, they can enter their email to be sent a one-time code to their email. They will then enter this code into the website to be given the ability to reset their password.

External Design Specifications:

ID	Deliverable Description	Comments	Evaluator (internal or external as applicable)
1	Postman API documentation	This would be used for developers to understand the different endpoints for the backend.	Internal: Development Team
2	Github Wiki	This would be used for developers to quickly access general documentation on the project.	Internal: Development Team

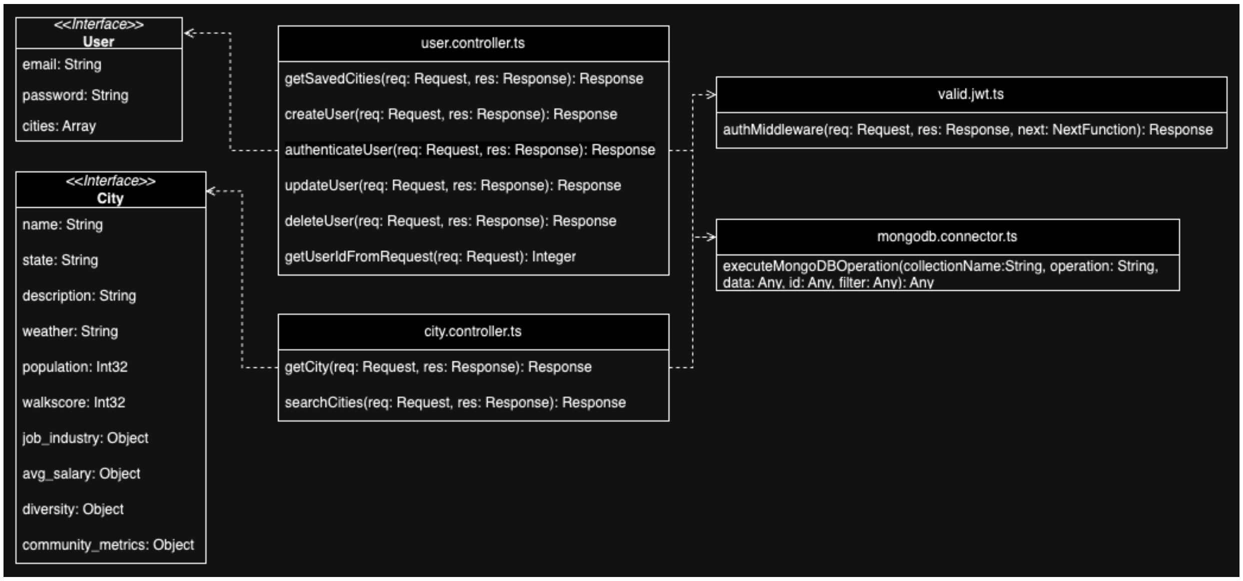
Detailed High-Level Solution Design

Functionality: The decoupled structure of a backend, frontend, and database allow for a separation of issues such that the frontend experiencing an issue will not impact the overall system including the database.

Security: CSRF Tokens will be used in addition to user session tokens to ensure CSRF attacks cannot be utilized against CitySurf. While the general risk of an attack is low as there is no sensitive user information, but the use of these tokens ensures account safety. The AWS services (Amplify and Elastic Beanstalk) will have rules to prevent DOSS attacks. Passwords will be encrypted using standard the Crypto JS library, ensuring that a database breach does not result in a harmful data leak.

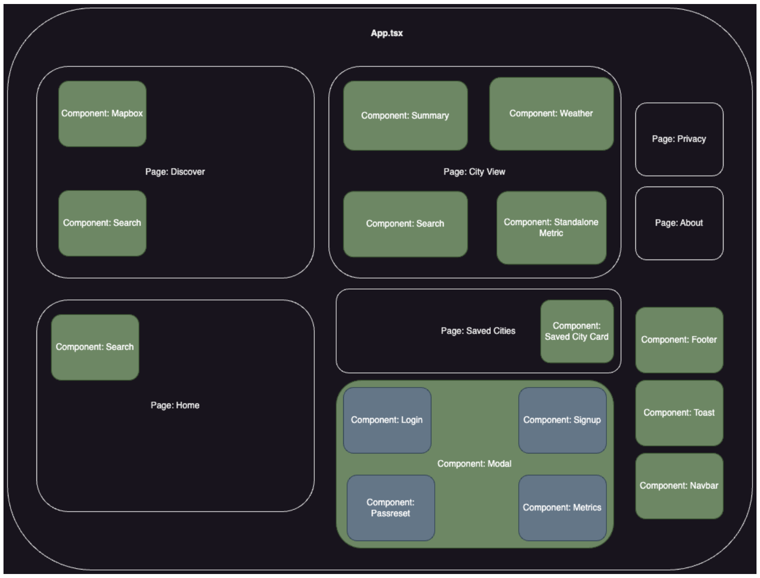
Performance and Availability: AWS Amplify and Elastic Beanstalk can be scaled as usage increases, allowing for high availability. MongoDB's finding methods are efficient for large data due to its use of indexes, which significantly speed up query performance by allowing the database to quickly locate and access the necessary documents without scanning the entire collection.

UML Diagram of Express App:


<<Interface>>
User
email: String
password: String
cities: Array
<<Interface>>
City
name: String
state: String
description: String
weather: String
population: Int32
walkscore: Int32
job_industry: Object
avg_salary: Object
diversity: Object
community_metrics: Object
user.controller.ts
getSavedCities(req: Request, res: Response): Response
createUser(req: Request, res: Response): Response
authenticateUser(req: Request, res: Response): Response
updateUser(req: Request, res: Response): Response
deleteUser(req: Request, res: Response): Response
getUserIdFromRequest(req: Request): Integer
city.controller.ts
getCity(req: Request, res: Response): Response
searchCities(req: Request, res: Response): Response
valid.jwt.ts
authMiddleware(req: Request, res: Response, next: NextFunction): Response
mongodb.connector.ts
executeMongoDBOperation(collectionName:String, operation: String, data: Any, id: Any, filter: Any): Any

Description: While a traditional UML diagram is not possible, the above diagram show the modularity of the backend in utilizing interfaces for structuring data received from the database, and using controllers for the different API calls.

React Pages and Components Diagram: Click here to view full image

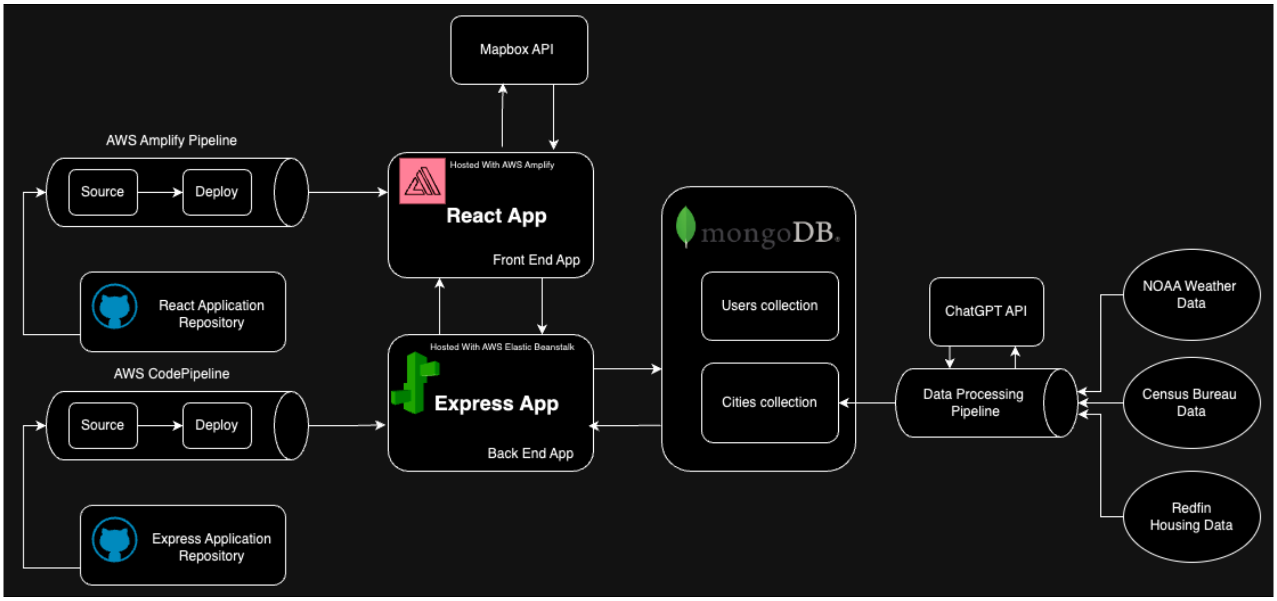


Description: The above shows the components (purple) and the pages (blue). This separation of elements can permit components to be re-placed on any page, allowing for future flexibility.

	Rationale of Technologies	
Software Technologies	**Description**	**Rationale**
ExpressJS	A minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications.	Keeping the language consistent for the backend as what is used for the frontend will help with transitioning development work between the two projects.
ReactJS	A JavaScript library for building user interfaces, particularly single-page applications, by creating reusable components and managing the state of web applications efficiently.	React is component based, allowing for a modular design that also allows for flexibility. This will be needed as requirements change and the overall design may change.
Mapbox	A platform offering customizable maps and location-based services, providing tools for creating interactive and visually appealing maps for web and mobile applications.	This API allows up to 200,000 monthly calls before any costs accumulate. This is a cost-effective solution in the event of little no usage but cost scales appropriately based on usage (Mapbox, n.d.).
FuseJS	A lightweight JavaScript library for performing fuzzy searching and text matching, enabling users to find relevant results in a dataset with approximate string matching.	This free library will allow users to make typing mistakes when entering a city name such that it auto corrects and provides suggestions to select (Fuse, n.d.).
Altcha	CAPTCHA alternative designed to provide secure, user-friendly verification solutions without relying on traditional CAPTCHA challenges.	This acts as a security measure to prevent fake accounts being created by bots. This solution can be self-hosted and free to use (Altcha, n.d.).
PlotlyJS	A modern JavaScript library for creating dynamic and interactive charts and visualizations, offering customizable and responsive charting options for data presentation.	This free library will allow customizing the look and feel of each chart and diagram allowing for flexibility in the altering of the site UI.
Python Pandas	A python library for data manipulation and analysis.	This free python library can be used to manipulate large data files and perform the refactoring and processing of data into the form needed for the MongoDB document.
MongoDB Atlas	A cloud service for hosting a non-relational document based database management system.	The free tier version allows up to 512MB which will more than accommodate the 20MB of city data currently estimated.

AWS Amplify	A set of tools and services that streamline the development and deployment of scalable full-stack web and mobile applications, providing features for authentication, APIs, and storage.	This service on AWS is optimized for single page navigation frameworks such as React. The build in CI/CD with Github repositories make this a reasonable option for deploying the frontend.
AWS Elastic Beanstalk	A fully managed service that simplifies the deployment, management, and scaling of web applications and services by automatically handling infrastructure provisioning, load balancing, and scaling.	This service allows for granular control over the infrastructure allowing for needed flexibility in changing the operation of the data flow.
AWS Code Pipeline	AWS CodePipeline is a fully managed continuous integration and continuous delivery (CI/CD) service that automates the build, test, and deploy phases of your release process to enable rapid and reliable software updates.	Streamlines and automates the software release process to Elastic Beanstalk, reducing manual effort and accelerating the delivery of high-quality applications with consistent and repeatable deployments.
Git + Github	A decentralized version control system allowing ease of collaboration and continual integration and deployment methods.	Using a Github repository creates a seamless experience where making a commit can trigger an action of unit tests to improve quality of code and automatic deployment.
Winston	A JS Library used for logging.	This free JS Library integrates with Loggly.com which can provide insights into usability and other key metrics.
Loggly	A cloud-based log management service that aggregates and analyzes log data from various sources to help monitor and troubleshoot applications.	Loggly’s free tier plan will provide efficient error tracking, real-time monitoring, and enhanced security, leading to a more reliable and user-friendly experience.
Uptime Robot	UptimeRobot is a monitoring service that tracks the uptime and performance of websites and servers, alerting users to any downtime or issues.	Uptime robot ensures continuous monitoring and quick alerts for any service interruptions, enhancing reliability and user trust. Their free tier plan offers basic functionality.
ChatGPT	A LLM used to generate text based on response provided to the model.	This LLM will have an initial low cost to generate accurate 3 sentence summaries for each city in United States that can be displayed.

Logical Solution Design:



Detailed Technical Design

General Technical Approach:

While some effort has been put into researching different data sources, building out the application's frontend and backend will be prioritized with the assumption of data retrieval being successful. Both the React and Express application will use a modular design for components, functions, and other key pieces to emphasize code reusability and maintainability. The data pipeline will also be setup such that each data source has a separated python script for processing and uploading the data to MongoDB.

Key Technical Design Decisions:

Backend Framework**:** ExpressJS, a minimal and flexible Node.js web application framework that provides robust features for building web and mobile applications. Keeping the language consistent for both backend and frontend will help streamline development work.

Frontend Framework: React, a JavaScript library for building user interfaces, especially single-page applications, using reusable components and efficient state management. React's component-based architecture allows for a modular design, offering flexibility as requirements and designs change.

Interactive Map: Mapbox, a platform offering customizable maps and location-based services, with tools for creating interactive and visually appealing maps for web and mobile applications. The API allows up to 200,000 monthly calls before incurring costs, making it a cost-effective solution with scalable pricing.

City Search Functionality: Fuse.js, A lightweight JavaScript library for fuzzy searching and text matching, enabling users to find relevant results with approximate string matching. This free library helps users correct typing mistakes when entering city names by providing suggestions.

Fake Account Protection: Altcha, a CAPTCHA alternative designed for secure, user-friendly verification without traditional challenges. It prevents fake account creation by bots and can be self-hosted and free to use.

Charts and Diagrams: PlotlyJS is a modern JavaScript library for creating dynamic and interactive charts and visualizations, offering customizable and responsive charting options. This free library allows for extensive customization of charts and diagrams, enhancing the site's UI.

Data Processing Tools: Pandas is a Python library for data manipulation and analysis. This free library can process large data files, refactoring and preparing data for MongoDB storage.

Database: MongoDB Atlas is a cloud service for hosting a non-relational document-based database management system. The free tier allows up to 512MB of storage, more than enough for the estimated 20MB of city data.

Front end deployment: AWS Amplify provides tools and services for streamlining the development and deployment of scalable full-stack web and mobile applications, offering authentication, APIs, and storage features. Optimized for single-page frameworks like React, with built-in CI/CD integration with GitHub.

Backend Deployment: AWS Elastic Beanstalk is fully managed service simplifying the deployment, management, and scaling of web applications by handling infrastructure provisioning, load balancing, and scaling. It offers flexibility in managing the infrastructure and data flow operations.

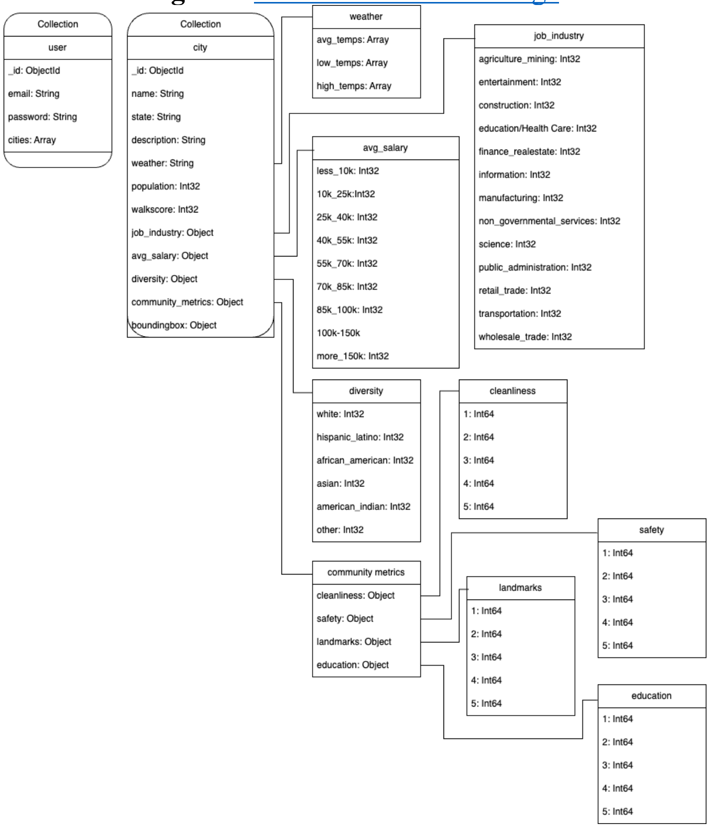
Backend Continual Deployment: AWS Code Pipeline fully managed CI/CD service automating the build, test, and deploy phases of software releases, enabling rapid and reliable updates. It streamlines and automates the release process to Elastic Beanstalk, ensuring consistent and repeatable deployments.

Version Control System: Github is a decentralized version control system facilitating collaboration and CI/CD methods. Using GitHub repositories enables seamless commits that can trigger unit tests and automatic deployments, improving code quality.

Logging Library: Winston is a JS library for logging. This free library integrates with Loggly, providing insights into usability and other key metrics.

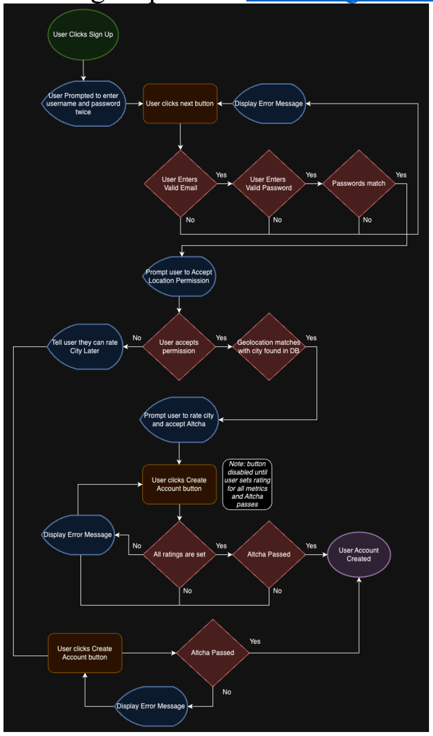
Logging Storage: Loggly is a cloud-based log management service that aggregates and analyzes log data to help monitor and troubleshoot applications. Loggly's free tier plan offers efficient error tracking, real-time monitoring, and enhanced security, ensuring a reliable and user-friendly experience.

Database Diagram: Click Here for Full Image

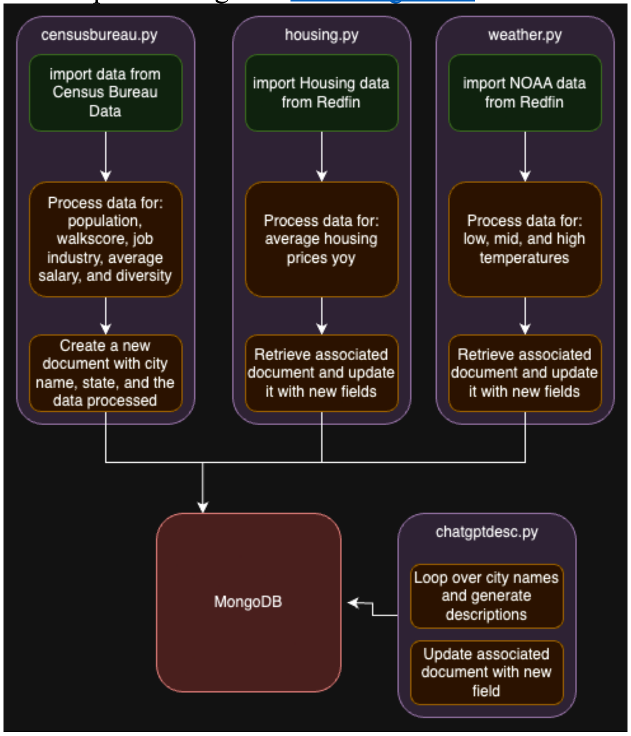


Flow Charts/Process Flows:

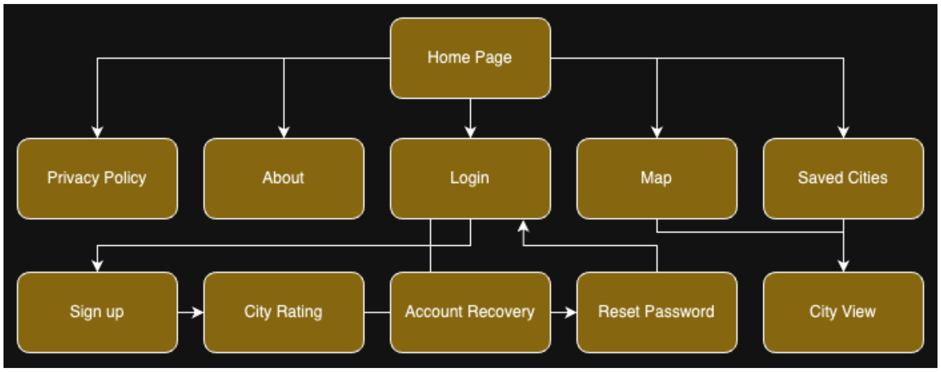
User Sign Up Flow: Full image here


User Clicks Sign Up
User Prompted to enter username and password twice
User clicks next button
Display Error Message
User Enters Valid Email
Yes
User Enters Valid Password
Yes
Passwords match
Yes
No
No
No
Prompt user to Accept Location Permission
Tell user they can rate City Later
No
User accepts permission
Yes
Geolocation matches with city found in DB
Yes
Prompt user to rate city and accept Altcha
User clicks Create Account button
Note: button disabled until user sets rating for all metrics and Altcha passes
Display Error Message
No
All ratings are set
Yes
Altcha Passed
Yes
User Account Created
No
No
User clicks Create Account button
Altcha Passed
Yes
No
Display Error Message

Data Pipeline Diagram: Full image here

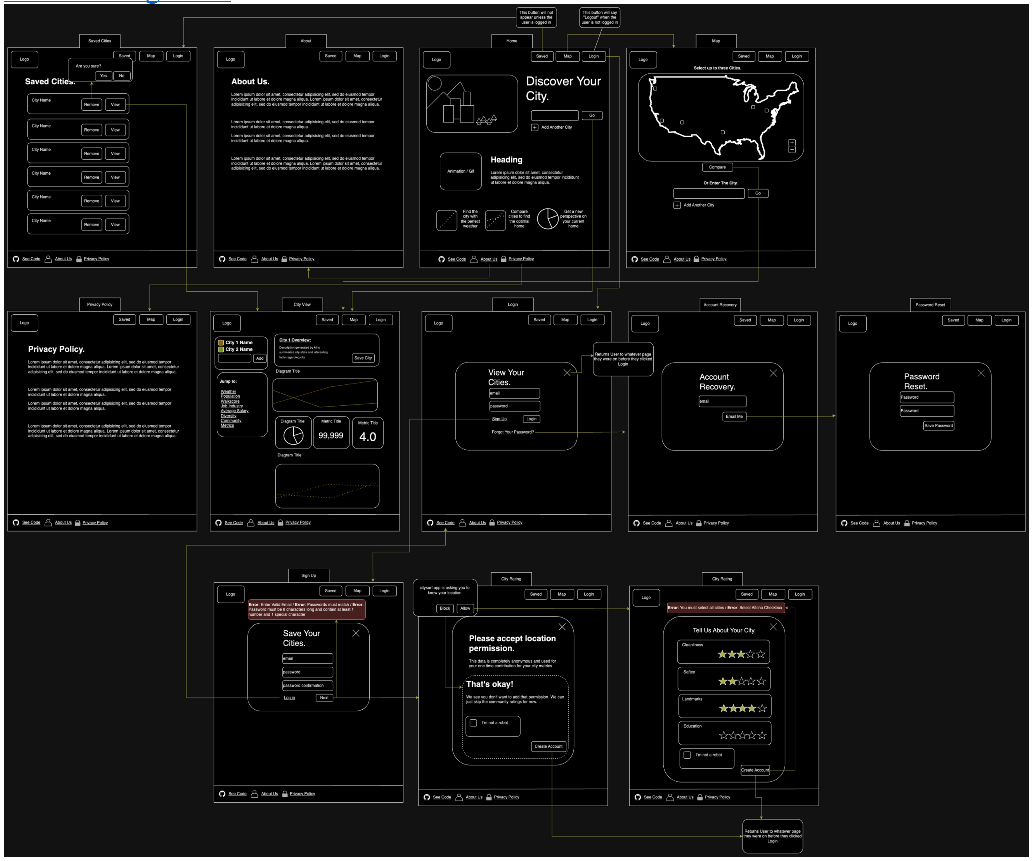

censusbureau.py
import data from Census Bureau Data
Process data for: population, walkscore, job industry, average salary, and diversity
Create a new document with city name, state, and the data processed
housing.py
import Housing data from Redfin
Process data for: average housing prices yoy
Retrieve associated document and update it with new fields
weather.py
import NOAA data from Redfin
Process data for: low, mid, and high temperatures
Retrieve associated document and update it with new fields
MongoDB
chatgptdesc.py
Loop over city names and generate descriptions
Update associated document with new field

Sitemap Diagram:



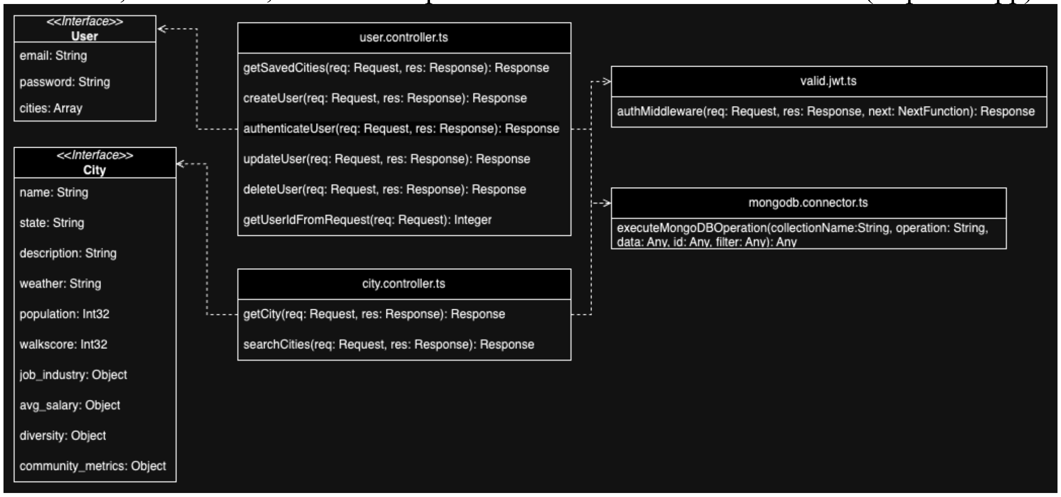
User Interface Diagrams:

See full image here



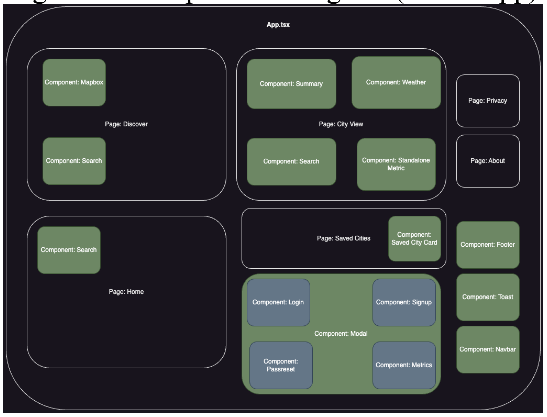
UML Diagram:

Interfaces, Controllers, and other exported functions used in the backend (Express App):



<<Interface>>
User
email: String
password: String
cities: Array
<<Interface>>
City
name: String
state: String
description: String
weather: String
population: Int32
walkscore: Int32
job_industry: Object
avg_salary: Object
diversity: Object
community_metrics: Object
user.controller.ts
getSavedCities(req: Request, res: Response): Response
createUser(req: Request, res: Response): Response
authenticateUser(req: Request, res: Response): Response
updateUser(req: Request, res: Response): Response
deleteUser(req: Request, res: Response): Response
getUserIdFromRequest(req: Request): Integer
city.controller.ts
getCity(req: Request, res: Response): Response
searchCities(req: Request, res: Response): Response
valid.jwt.ts
authMiddleware(req: Request, res: Response, next: NextFunction): Response
mongodb.connector.ts
executeMongoDBOperation(collectionName:String, operation: String, data: Any, id: Any, filter: Any): Any

Pages and Components Diagram (React App):



Service APIs:

MapBox API: This can be used to create selections in specific locations as detailed in the MapBox's API documentation: https://docs.mapbox.com/help/tutorials/add-points-pt-1/

Loggly: Loggly provides the necessary documentation for uploading data to their services via Winston here: https://www.loggly.com/blog/using-winston-3-for-logging-in-node-js/

ChatGPT: They provide their own API documentation for generating responses programmatically with python here: https://platform.openai.com/docs/quickstart

NFR's (Security Design, etc.):

CSRF Tokens will be used in addition to user session tokens to ensure CSRF attacks cannot be utilized against CitySurf. While the general risk of an attack is low as there is no sensitive user information, but the use of these tokens ensures account safety. The AWS services (Amplify and Elastic Beanstalk) will have rules to prevent DOSS attacks. Passwords will be encrypted using standard the Crypto JS library, ensuring that a database breach does not result in a harmful data leak.

Operational Support Design:

Uptime robot will be used to continually monitor the services for usability and alerts when services are not working as expected. Winston will be used to data regarding web requests made to each end point in the backend and specific functions accessed to Loggly for further analysis.

Other Documentation:

- Project Proposal
- Project Requirements

Appendix B – References

Altcha. (n.d.). Retrieved August 4, 2024, from https://altcha.org

Fuse.js. (n.d.). Retrieved August 4, 2024, from https://www.fusejs.io/

Mapbox. (n.d.). Add points to a map. Retrieved August 4, 2024, from https://docs.mapbox.com/help/tutorials/add-points-pt-1/

OpenAI. (n.d.). Quickstart. OpenAI. Retrieved August 4, 2024, from https://platform.openai.com/docs/quickstart

Appendix C – External Resources

GIT URL:	*https://github.com/scottwmaxwell/citysurf*
Hosting URL:	*N/A*