User/Developer Manual for Florida Tech History Tours

Project Website:
https://fit-history-app.github.io/

Team:
Tyler Zars
Grant Butler
Cameron Miskell
Matthew Tokarski

Faculty Advisor:
Dr. Fitzroy Nembhard (fnembhard@fit.edu)

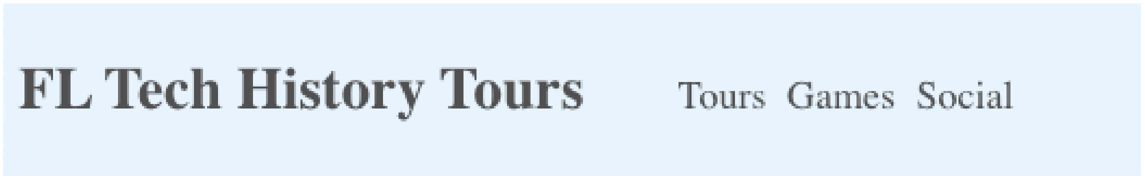
Client:
Dr. Ryan Stansifer (ryan@fit.edu)

Table Of Contents

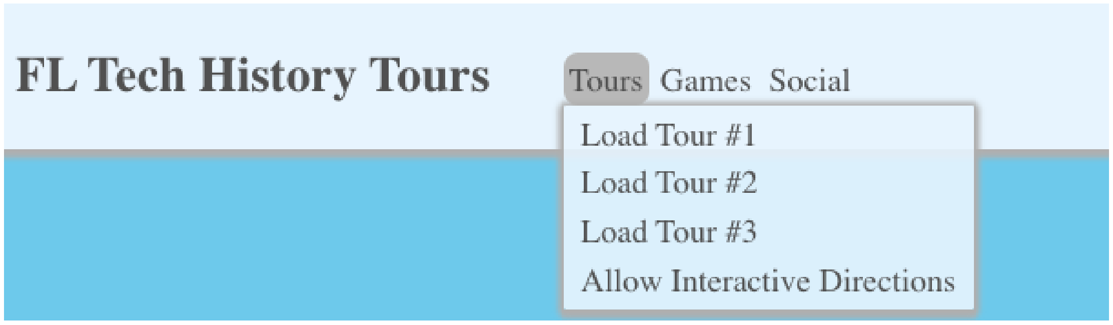
User Guide

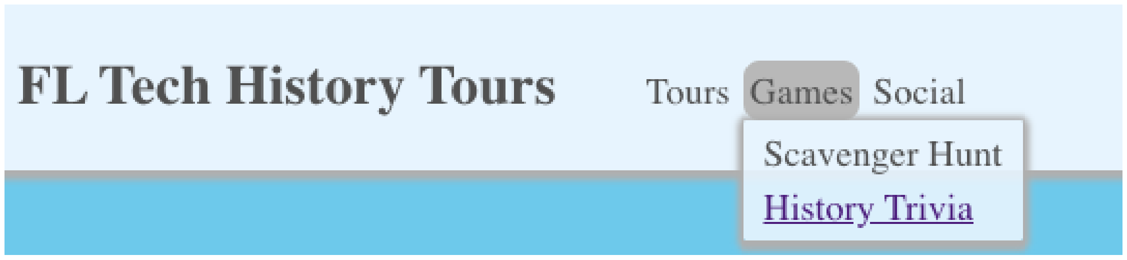
This web application provides the user with the ability to explore Florida Tech's grounds while learning about the history of the campus. Each piece of the application is broken down below to cover how it works and how it can be used.

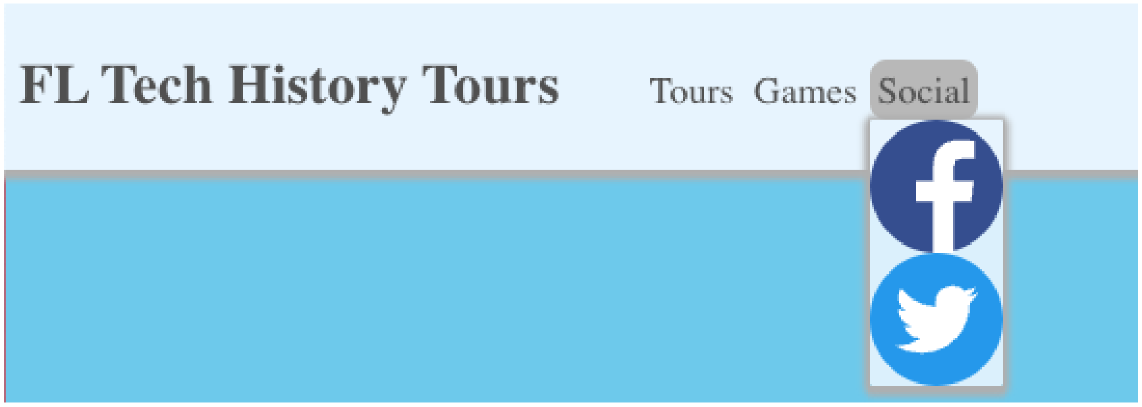
Menu Bar



The menu bar is presented to the user at the top of the page to allow for navigation between different pieces of the application. The main bolded text allows the user to ensure they are loaded on the correct site. Each of the following suboptions will bring up a dropdown as pictured below.







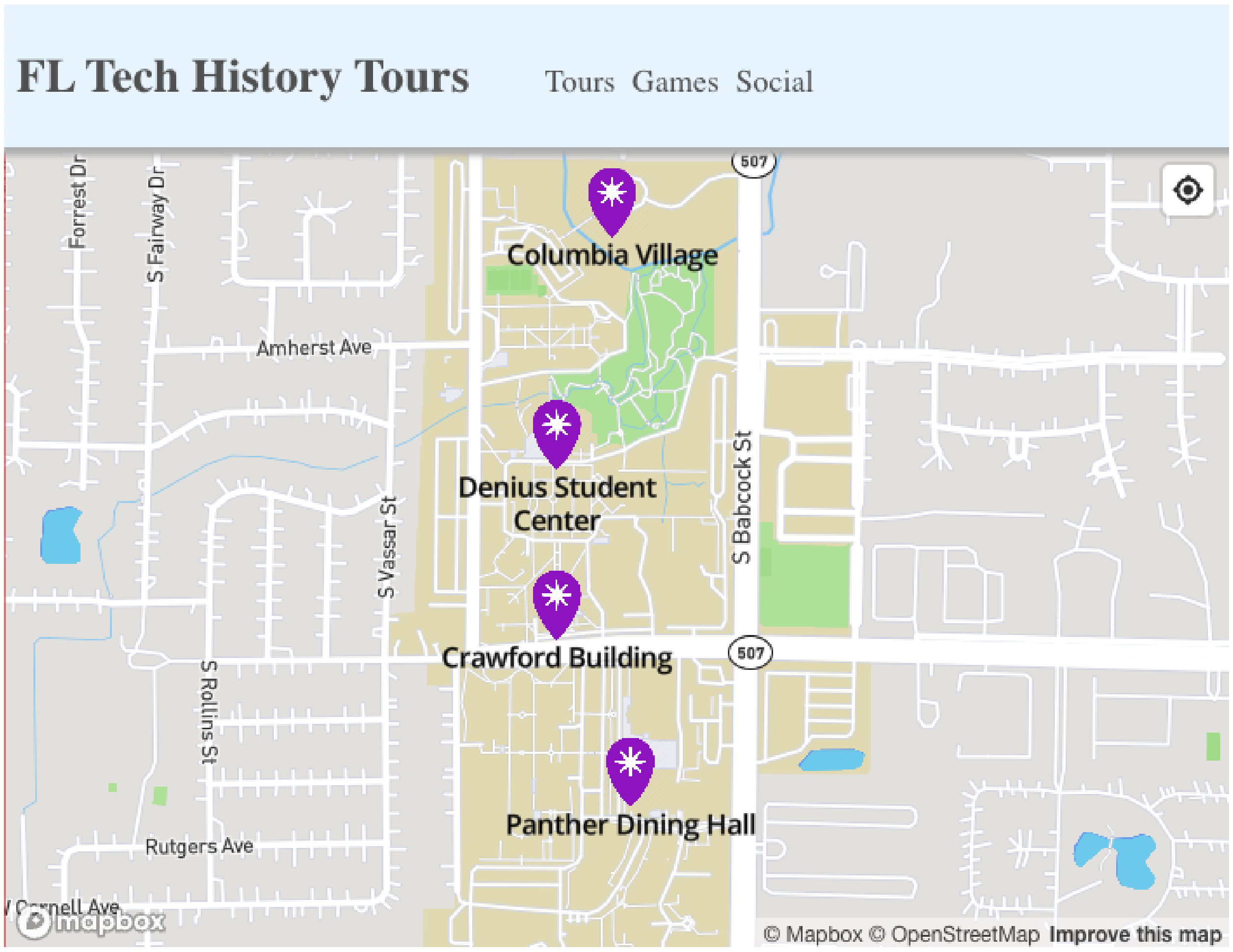
The tours navigation will open up and allow the user to choose from one of the applications tours. Upon clicking one, you can take a tour and this is covered at the “Tours” section of the user guide.

The games navigation will provide a dropdown menu to the two different interactive games included in the application. Upon clicking into one of the games, you will be directed to a new page with the games. From this page you’ll be able to interact with these games as outlined in the “Trivia” and “Scavenger Hunt” sections of the user guide.

The social navigation will provide a dropdown menu to the different social media applications linked to the Tours app. These buttons will redirect externally to the desired social media application as outlined in the “Social Media Integrations” section of the user guide.

Map

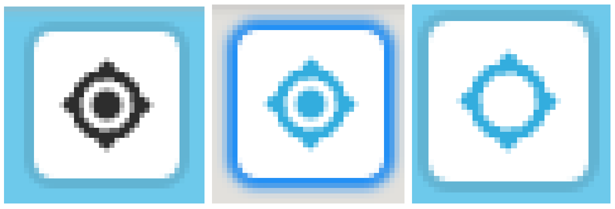
The Map component of the homepage is directly below the menu bar and provides the user with an interactive map to scroll around the campus with. The main map component can be seen below.



This map is fully interactive with the user being able to zoom in and out while scrolling around the map using touch gestures. The zoom can be done with either a touchscreen, mouse, or trackpad. Each of these devices require different gestures but they remain consistent with zoom across other applications. The user can also scroll on the map by clicking and dragging around. This will move the main map element around; to recenter yourself on the map, refresh the page and you will be recentered over Florida Tech's campus.

The users map will provide them with a layer of Point of Interests (POIs) symbolized by the purple icons. These are rendered as the user zooms and moves to show different POIs near their current location.

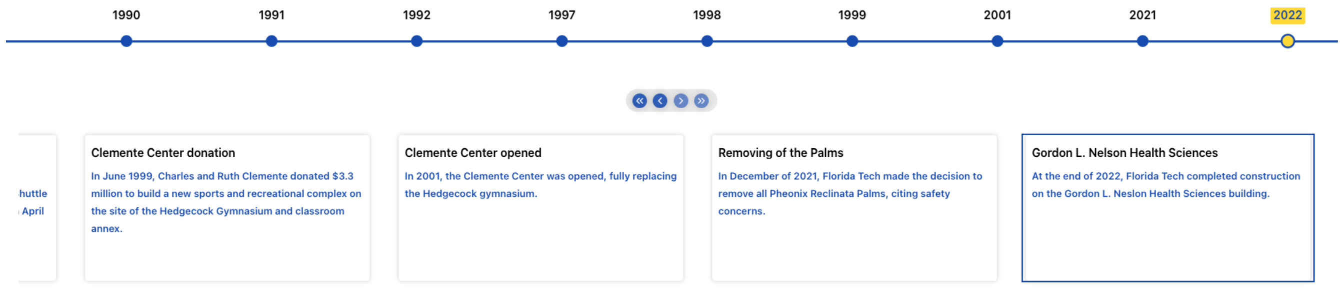
The map also houses a button in the top right corner to enable the user to have the application track their location. This is used on tours and for the dynamic timeline content. The button has three states pictured below.



The first example is the default state of the geolocation button where the user's location is not currently being tracked. This state is the default anytime the page is loaded by the user. Upon clicking the geolocation button, the icon will spin until it enters the state of the second image above. This icon means that your current location has been captured and the map will update as you move. If you scroll or move the map while your location is being used, the geolocation button will begin to look like the third image. In this state, you can click on the geolocation button to bring you back to your current location. When in the second state, you can click on the geolocation button again to stop your location from being shared.

Timeline

The timeline below the map provides the user with historical content about the university. This content is rendered in a linear timeline broken down by the year each event occurred. This timeline is dynamic to the user's location when the geolocation button is enabled (reference). An example of the timeline is shown below.



huttle
April

Clemente Center donation

In June 1999, Charles and Ruth Clemente donated $3.3 million to build a new sports and recreational complex on the site of the Hedgecock Gymnasium and classroom annex.

Clemente Center opened

In 2001, the Clemente Center was opened, fully replacing the Hedgecock gymnasium.

Removing of the Palms

In December of 2021, Florida Tech made the decision to remove all Pheonix Reclinata Palms, citing safety concerns.

Gordon L. Nelson Health Sciences

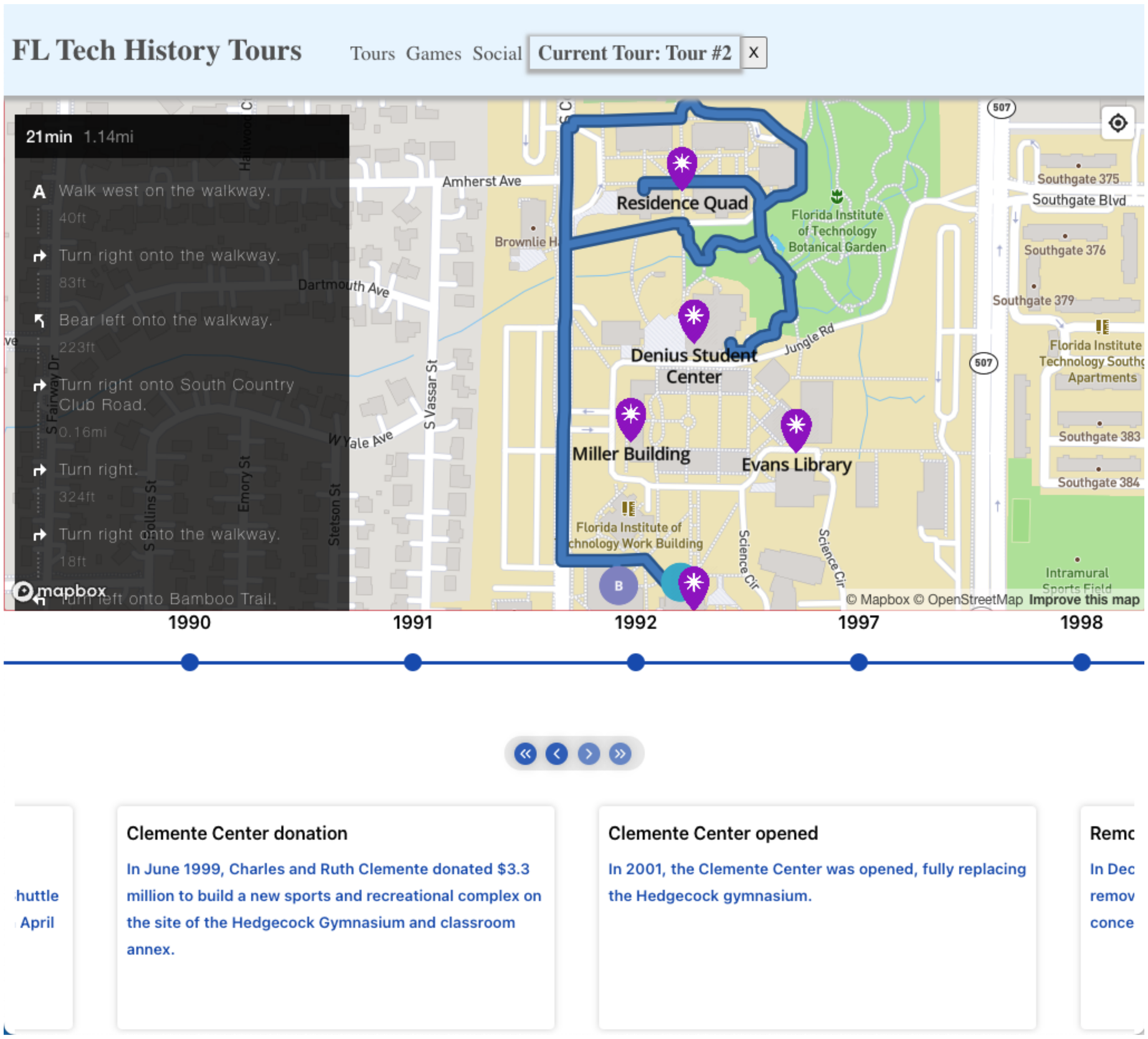
At the end of 2022, Florida Tech completed construction on the Gordon L. Neslon Health Sciences building.

This timeline has directional buttons to allow the user to scroll through the timeline and read the different facts presented by the web application. Users may click on the year they wish to see a fact for or they may utilize the directional buttons. These four buttons are summarized below.

- Jump to first timeline card: Clicking on this button will immediately move the timeline to the first card.
- Move to previous timeline card: Clicking on this button will move the timeline to the previous card in the sequence.
- Move to next timeline card: Clicking on this button will move the timeline to the next card in the sequence.
- Jump to last timeline card: Clicking on this button will immediately move the timeline to the last card.

Tours

With the intention of guided tours around campus, upon selecting a tour from the menu bar the application will load the map with a set of directions on the left to guide you along the tour. The menu bar will display the current tour you have selected. If you would like to cancel the tour, hitting the “X” button next to the tour name will remove it from the map and take the directions off the screen.



21 min 1.14mi

A Walk west on the walkway.
40ft

Turn right onto the walkway.
83ft

Bear left onto the walkway.
223ft

Turn right onto South Country Club Road.
0.16mi

Turn right.
324ft

Turn right onto the walkway.
18ft

Turn left onto Bamboo Trail.

huttle
April

Clemente Center donation

In June 1999, Charles and Ruth Clemente donated $3.3 million to build a new sports and recreational complex on the site of the Hedgecock Gymnasium and classroom annex.

Clemente Center opened

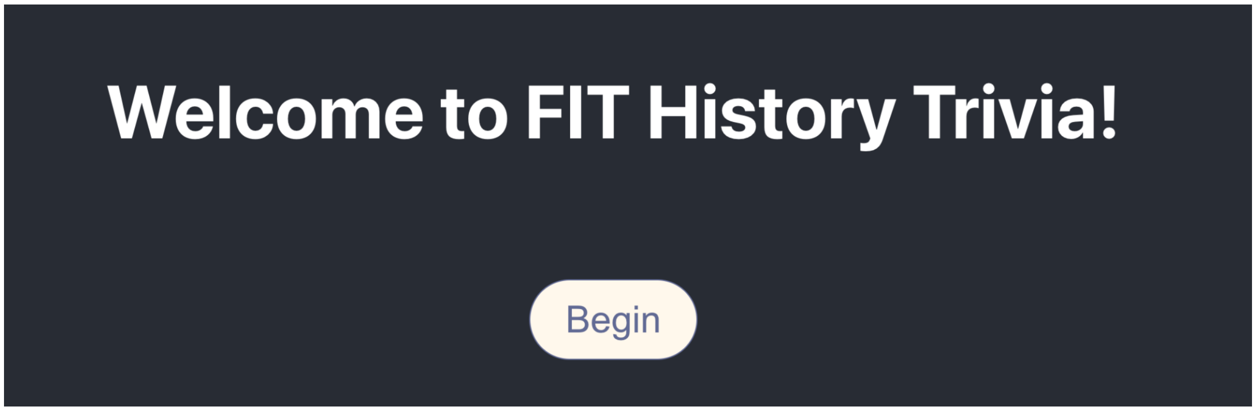
In 2001, the Clemente Center was opened, fully replacing the Hedgecock gymnasium.

Remo
In Dec
remov
conce

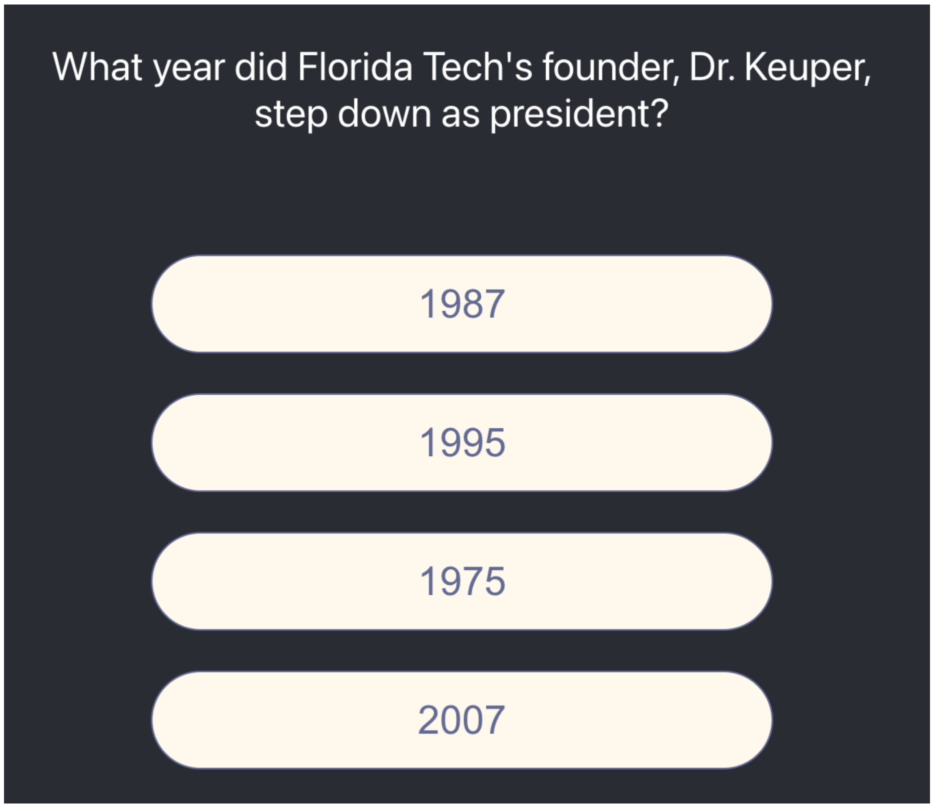
The directions on the left hand pane are enabled to guide you on a walking tour of campus based on the chosen tour. The directions pane is scrollable so you can scroll ahead to see where you are headed. There will be a point “A” and a point “B” on the map which is your starting point (“A”) and your primary destination (“B”). Point “A” is dynamically placed based on the user's current location. If the user doesn’t have their location enabled, the current center of the screen will be used instead.

Trivia

The trivia module can be loaded from the menu bar. Upon redirecting to the trivia route, you’ll be greeted with a screen to begin the tour game. Once the begin button is clicked, the game will present the user with a trivia question. The start screen can be seen below.

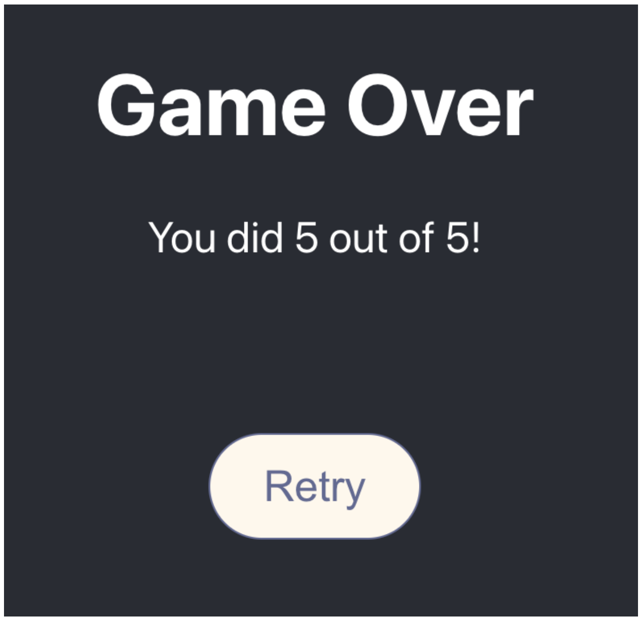


Upon starting, the quiz will randomly present you with five trivia questions in a wide range of topics related to Florida Tech. There are ~150 trivia questions in the application that will be randomly chosen each time that the quiz begins. The trivia questions load with four buttons and your objective is to click the button with the correct answer. The trivia page with the questions will look something like the image below.



What year did Florida Tech's founder, Dr. Keuper, step down as president?

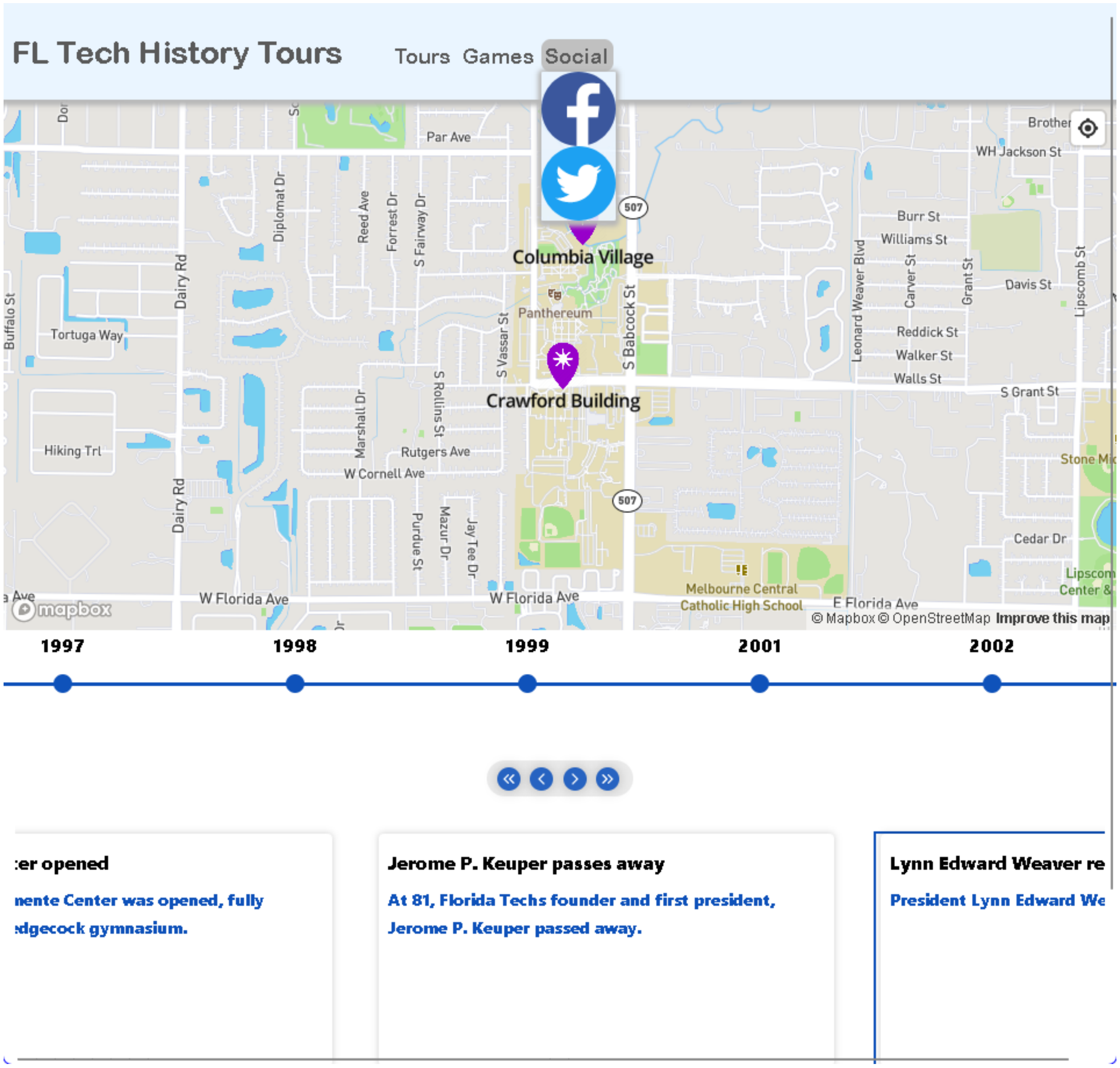
Once you have finished all five questions, the trivia game will provide you with a score screen to tell you how many answers you got correct out of the five rounds. There is also a retry button that will restart the game and load in a new set of questions.



Scavenger Hunt

Social Media Integrations

Our application has integrations linked to Facebook and Twitter for social media sharing of tour feedback, game performance, favorite facts, and more! By hovering over the social sub-menu in the menu bar at the top of the screen, users will be able to view and navigate to their desired social platform by clicking on the available icons linked to the corresponding website. Users should take relevant pictures or screenshots at their discretion for posting to a given platform, as the app will not track user activity automatically nor inject content to posts.



ter opened

nente Center was opened, fully
edgecock gymnasium.

Jerome P. Keuper passes away

At 81, Florida Techs founder and first president,
Jerome P. Keuper passed away.

Lynn Edward Weaver re

President Lynn Edward We

Twitter Integration

When a user chooses to share their thoughts about our app on Twitter, they will be sent to the default sharing action for that platform, writing a tweet to post on their feed. If using a browser that has not been signed into, the Twitter website will take over from that point on and request that the user log in to a registered account prior to authoring their tweet.

Facebook Integration

If a user would like to post about the History Tours experience in their Facebook communities, the integration link will take them to begin writing on their default feed. Similarly to Twitter, a user will also need to login before they are able to post to Facebook.

Developer Guide

Source Code Repository

The source code for fit-history-tours.live can be found at the following GitHub link: https://github.com/fit-history-app/fit-history-app. The main branch is currently the live deployment behind the running web application.

Downloading and Running

To install and run your own Florida Tech History Tours web application you will first need to download the code from the GitHub repository. Make sure you have git installed on your machine. We recommend navigating to the folder you want to install the program in your operating system's terminal and then running the following command:

```
> git clone https://github.com/fit-history-app/fit-history-app.git
```

Once you have downloaded the project files you will need to download the libraries needed to run the web server. First make sure that you have `node.js` installed on your machine. You can check if you have `node.js` installed on your machine by running the following command:

```
> node -v
```

The next step is to make sure you install all the necessary libraries so you will need to run the following command:

```
> npm install
```

The final step is to start the server. You can start the server with either of the following commands:

```
> npm start
```

You can then navigate to `localhost:3000` in your preferred web browser to see the server running. We recommend Google Chrome or Firefox for the best experience.

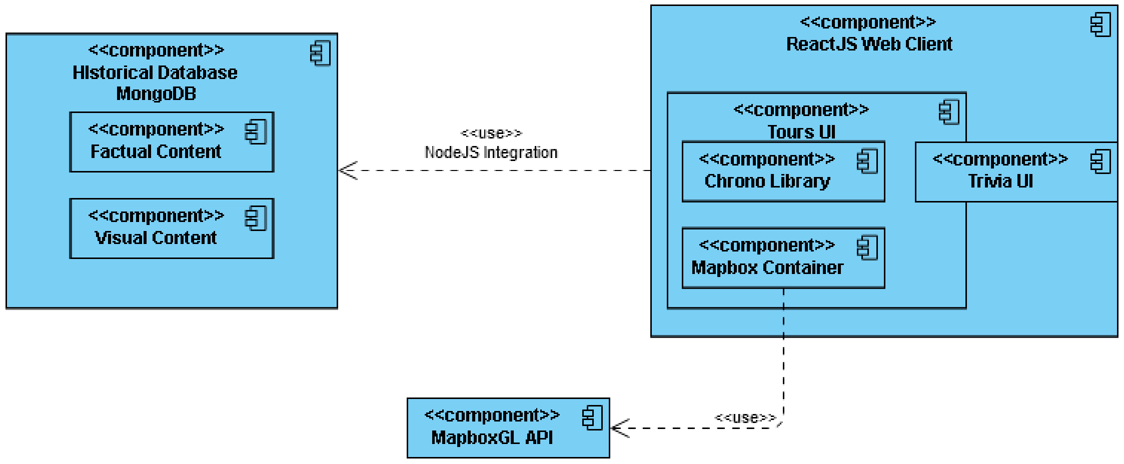
Deployment

Florida Tech History Tours is a statically built web application and is deployed utilizing a DigitalOcean 'app'. DigitialOceans 'app' is updated by making changes to the main branch of the central GitHub repository for the project. When any code is changed, this will trigger DigitalOceans hosting to rebuild and redeploy the web application with any new changes. Upon build errors, it will revert back to the last successful build of the web application. This web application is behind the DNS name of fit-history-tours.live which will remain active until October

13th, 2023. After this point, the web application will still be reachable from the default DigitalOcean domain name of https://urchin-app-4onk7.ondigitalocean.app/.

System Architecture

This application revolves around three main pieces. The main Web application is statically served from DigitalOcean with three components. The main page loads with the MapBox container, reactive timeline, and menubar. This page dynamically grabs data from the database to present it to the user. The map and directions use the MapBox API to display and route the directions. Lastly, the application can redirect from the trivia to the main component of the application by the internal React Router.



Source Files

Assets / Public

These files are mainly in relation to the provided images served by our application. Each image has its own purpose and all files in these folders are broken down in the table below.

facebook_svg_icon.svg	This provides the icon for the Facebook integration
twitter_svg_icon.svg	This provides the icon for the Twitter integration
favicon.ico	This is the icon that shows up in the browsers tab view.
index.html	This provides the landing page if there isn't JavaScript in the browser.
logo192.png	This provides the icon for web applications added to the home screen of mobile devices.
logo512.png	This provides the icon for web applications added to the

	home screen of mobile devices.
manifest.json	This JSON provides details about which images should be served when the page is rendered or saved.
robots.txt	This text file provides information for search engines
star-waypoint-icon.png	This image is used on the applications map to mark points of interest around the campus.

Routes

Root

The root JavaScript XML (JSX) file provides the main application view after the page is loaded. The menu bar, timeline, and map components are all rendered from this file. The functions in this file are broken down below with their use cases.

const [lng, setLng] = useState(-80.6239659);
 	
const [lat, setLat] = useState(28.0645427);
 	
const [zoom, setZoom] = useState(14);
 	
const [tour, setTour] = useState(null);
	These four variables are state variables defined using the "useState" hook from React. The hook returns an array with two elements: the current state value and a function to update the state value. The first element of the array (e.g. "lng", "lat", "zoom", and “tour”) holds the current value of the state, and the second element (e.g. "setLng", "setLat", "setZoom", and “setTour”) is a function that allows you to update the state.
	
	"lng", "lat", "zoom", and “tour” keep track of the user’s current longitude, latitude, zoom, and tour.

getData()
useEffect(
    getData,
    []
);
	The code defines a function named "getData" that fetches data from the trivia YAML file using the "fetch" function in JavaScript. Once the data is received, it is processed using the "load" function, which is assumed to be defined elsewhere. The resulting data is then logged to the console and returned from the function.
	
	This function is then used inside a "useEffect" hook, which is a React hook that allows you to run side effects in functional components. The "useEffect" hook takes two parameters:
	1. A function to run on every dependency update (in this case, the "getData" function).
	2. An array of dependencies that the function depends on (in this case, an empty array, indicating that the function should only run once on component mount).

const geolocate	The code initializes a variable named "geolocate" that is an instance of the Mapbox GL JS "GeolocateControl" class. This control provides a button on the map that, when clicked, centers the map on the user's current location and displays a dot indicating their location. The "GeolocateControl" constructor function takes an object with several options: ● "positionOptions": An object with options for the position of the user. In this case, it only has one option, "enableHighAccuracy", which is set to true. This indicates that the user's location should be determined with high accuracy, if possible. ● "trackUserLocation": A boolean value indicating whether the map should continuously track the user's location as they move. In this case, it is set to true, indicating that the map should continue to center on the user's location as they move.
var directions	The code initializes a variable named "directions" that is an instance of the Mapbox Directions API. The "Directions" constructor function takes several parameters: ● "accessToken": A string value representing the access token for Mapbox API. ● "unit": A string value indicating the unit system used for directions. In this case, it is set to 'imperial'. ● "profile": A string value indicating the type of profile used for directions. In this case, it is set to 'mapbox/walking'. ● "alternatives": A boolean value indicating whether to calculate alternative routes or not. In this case, it is set to false. ● "controls": An object with options that control the appearance of the UI controls for the directions API. In this case, it has two properties: "inputs" and "profileSwitcher". Both are set to false, indicating that the input fields and profile switcher UI controls are hidden. ● "interactive": A boolean value indicating whether the user can interact with the directions on the map or not.
useEffect()	Initialize map
useEffect()	This function keeps track of the user's latitude, longitude, and zoom by updating the according state variables.

	If the user is in the bounds of Florida Tech, the according state variable is updated.
load_tour1()	This function sets up a “Full Campus" tour route with multiple waypoints using the Mapbox API. The function removes all existing waypoints and then the tour header is set to "Full Campus Tour" using the "setTour" hook. The tour starts from the current position, which is defined by the "lng" and "lat" state. The "directions.setOrigin" function sets the starting point for the tour. The waypoints are then added to the route using the "directions.addWaypoint" function. Each waypoint is specified with an index and a latitude-longitude coordinate pair. Finally, the destination is set using the "directions.setDestination" function. Once the route is set up, it can be displayed to the user using the Mapbox Directions API.
load_tour2()	This function sets up a “Housing" tour route with multiple waypoints using the Mapbox API. The methodology and formatting is identical to “load_tour1()”.
load_tour3()	This function sets up a “Places to Eat" tour route with multiple waypoints using the Mapbox API.
load_tour4()	This function sets up a “Academic Buildings" tour route with multiple waypoints using the Mapbox API.
remove_tour()	The code defines a function called "remove_tour" that removes the current tour from the map using the MapBox Directions API. Next, it sets the tour header to null using the "setTour" function. This removes the tour header. The function also uses React Hooks, specifically the "useRef" and "useEffect" hooks. The "useEffect" hook listens for changes to the "tour" variable. If the value of "tour" is null, it removes the route using the "directions.removeRoutes" function. This function removes all routes and associated markers from the map.

Trivia

This route contains the components and module for the trivia module. In the components folder, the “Button.jsx”, “GameOver.jsx”, “Quiz.jsx” and “Start.jsx” contain the code to administer the different pieces of the trivia game. The main “trivia.jsx” file just redirects to the different components and has only a single executable line.

Button.jsx	This code exports the CSS for all buttons used in the trivia.
GameOver.jsx	This is a functional component in React that renders a "Game Over" message with the points scored and a "Retry" button to start the game again. The component accepts a single prop, pts, which is a number representing the points scored by the player in the game. Inside the return statement, the component is rendered with a Title component with the text "Game Over", a Points component with the text "You did {pts} out of 5!", and a Button component with the onClick event set to the refreshPage function.
Quiz.jsx	The code defines a functional component named Quiz that uses React hooks to manage state. The state variables are quiz, number, and pts, initialized as an empty array, 0, and 0 respectively. The component also defines a shuffle function to shuffle arrays, and a pickAnswer function to handle user answers and update the state variables accordingly. The component uses the useEffect hook to set the quiz state variable to an array of 5 questions from the my_trivia.questions array, with each question being an object containing the question text, shuffled answer options, and the correct answer. The component returns a QuizWindow div containing a conditional rendering of the question and options based on the current number state. If the number state is equal to 5, the component renders the GameOver component passing the pts state as a prop. Overall, this code is a React component that manages the state of a quiz game and renders the appropriate UI for each question.

Start.jsx	This is a functional component in React that renders an introductory message for a trivia game with a "Begin" button that starts the game. The component accepts a single prop, used to control the state of the quiz application. The component defines a function named startQuiz that sets the value of the props object to true when called. This function is passed as a callback to the onClick event of the Button component. It returns a header and button to start the game.

Scavenger Hunt

Data

These data files are formatted as JSON (JavaScript Object Notation) or YAML (YAML Ain't Markup Language) to contain different pieces of information related to our application. These data files are loaded during the first load and then cached on the user's device to prevent them from needing to be reloaded during each subsequent load of the web application.

history.yaml	This file provides all of the historical facts relating to the application. The dynamic timeline at the bottom of the application pulls historical facts from this YAML file. Each datapoint has an 'Event' which acts like the title, a year, and a description.
poi_waypoints.json	This JSON file outlines all of the major buildings on campus with a centered coordinate waypoint. This is used to render the 'star-waypoint-icon.png' onto each major building on campus.
pois.yaml	This file contains specific facts about each building on campus with a year field for the building's build date and description of interesting things about said point of interest.
trivia_questions.json	This JSON file contains the questions used for the trivia part of the application. The question is stored in the 'text' field, the correct answer is stored in its own field, and the incorrect answers are stored in a list under the 'incorrect_answers' field.
trivia.yaml	This YAML file contains extra questions that can be imported into the trivia_questions.json file at the last point.

	These questions don't have incorrect answers and will need to have those added when utilized.

Styles

These CSS files contain the styling for the web application's different pieces. Each file is broken down below with the pieces of the application it covers.

index.css	This CSS defines the styles for the main webpage upon loading into the web application. It defines the styling for the Map, timeline, menubar, and other pieces.
trivia.css	This CSS defines the styles for the trivia webpage. It defines the styling for the menu, quiz, and ending score page.