Amtrak Rail Planner: An itinerary planning solution for multi-segment train trips

Nicholas Alvarez
Dept. of Computer Science and Engineering
University of Nevada, Reno
Reno, United States
CS 620

***Abstract*—Traveling by Amtrak train with a Rail Pass is a cost-efficient way to see the country in one month, but planning the trip can be a time-consuming process, due to the limitations of the Amtrak website. The Amtrak Rail Planner will provide travelers an easy way to formulate their multi-segment journey to any of the destinations served by train in the United States. The manual process of searching Amtrak's website for each travel leg and inputting information into an itinerary spreadsheet will be eliminated. Novel features include "click once" searching on Amtrak's site, archival of previous searches for potential reference later, displaying multiple pages of results in a single list, links to relevant train maps, and photo displays of each selected cities. This Python application provides an efficient and easy way to search for trains and create a travel itinerary which can be exported to a file.**

***Index Terms*—Data Collection, Graphical User Interfaces, Python, Rail Transportation, Web Mining**

I. Introduction

Amtrak [1] is the national rail network with service to over 500 destinations in 46 states. It is possible to travel from coast to coast on as little as two trains. For those wanting to see the country by train, multiple train segments are required, and the cost can often add up quickly. For this reason, Amtrak offers a Rail Pass for a flat price, granting the traveler ten segments between any destinations they choose, provided their entire excursion is completed in one month.

Planning a trip at this scale can be difficult. With so many potential routes and destinations, it easy to get overwhelmed trying to pick where one wants to go in such a short time-frame. This also comes with the necessity to balance train connections, ample time in each destination, and feasibility of arrival and departure times. The Amtrak website's method for planning trips is searching by origin, destination, and departure date one at a time [2]. A train must be selected from the results and its information typed in elsewhere, such as a spreadsheet. This is a time-consuming process, and if previous search results need to be checked, the search information must be re-entered, and the search resubmitted. The Amtrak Rail Planner will provide a way for travelers to formulate a travel plan, likely with the purchase of a Rail Pass, navigating and parsing the Amtrak website so the user does not have to. Selected trip segments will be saved in an ordered spreadsheet with relevant information for each train.

The intended users of the application are travelers who have purchased a Rail Pass and need to plan their trip. In actuality, the user demographic expands beyond solely pass holders, as the underlying search functionality would still allow for shorter or longer trips to be planned. However, one-way only searches and the UI elements related to the number of segments may create issues for usage outside of what is intended and designed for. Potential users will benefit from the application due to its immense time saving advantages. The inclusion of all stations may help inspire new travel destinations, but if a user already has an idea of which places they want to visit, the application will assist in compiling their trip segments into a spreadsheet itinerary for later review. This organizational benefit is not insignificant, especially if multiple potential groups of destinations are planned but the feasibility is undetermined. The application should be both effective and efficient to use. That is, the desired results (a travel plan) should be achieved in as minimal time as possible. Any issues with a search or application function should be handled with an informational error message, as error handling should be another goal. Finally, the application should be easy to learn, with the driving component of this goal being its minimal interactable UI elements that will guide the user to a completed search and finished itinerary.

The application would have some impacts across global, environmental, economic, and societal contexts. Globally, both Americans and non-Americans may be inspired to take an Amtrak train trip, with or without the Rail Pass. The more people that choose to take a train trip over flying or driving, the less emissions are produced on a per-passenger basis, which showcases the environmental impact. On a societal level, it may promote train travel as a viable means for cross-country trips where urgency is not a concern. Travel by rail shines when destinations are too far to drive and to close to fly. Families or friends could plan a trip together. The application shines with its economic impact, however. Those unaware of the Rail Pass and its benefits may be prompted to investigate further and buy one, instead of paying for each segment individually. On a more abstract level, the amount of time saved planning the journey, and potentially, the time maximized during the journey with a plan allowing for the most time at destinations, would further contribute. Destinations served by Amtrak may experience the impact of tourists arriving and spending money.

This application provides multiple improvements over the existing method of manually searching the Amtrak website for each trip segment and inputting details into a spreadsheet. In summary, the paper makes the following contributions to Amtrak itinerary planning.

1) The act of searching by the user is entirely removed as the web driver runs all searches in the background and handles any errors as they arise.
2) All results are displayed at once in the application, instead of potentially multiple pages on Amtrak's site. Each train in the results table is shown with user-selected attributes, such as duration or prices.
3) Results from previous searches can be viewed without the need to perform another search.
4) A specific journey from search results can be saved to an itinerary, thus forming another leg in the user's multi-segment journey.
5) All trains in the itinerary, with the user's selected attributes, can be exported to spreadsheet form, thus eliminating the need for manual input when searching on Amtrak's site.

Other minor contributions include photo displays of currently selected cities, tourism information about each city, menu links to route maps and train status maps, and links to a selected train's information. These contributions are all a part of a standalone application, so the user would only need to navigate to Amtrak's website to book their trips once an itinerary is created.

II. Related Work

Web scraping is not a new field, but there does not seem to be much, if any, existing research using this technology with Amtrak's site. Gheorghe et al. discuss methods for scraping data from websites with varying layouts, such as pure HTML or with the inclusion of Javascript elements [3]. They discuss the use of Selenium, as is used in the application, and its capabilities with modern web browsers and element parsing abilities. There is some more work and available options in the area of schedule creation. Amtrak does offer a multi-city search function on their website [4], and while it does offer some of the functionality of the application, it is still limited in that only four segments can be searched for at one time, and results are still listed on multiple pages with a great deal of wasted space. Additionally, it does not offer any way to save trains for later to an itinerary. The most similar work to the Amtrak Rail Planner is the "Intelligent Traveling Service" from Navabpour et al. [5]. Their work is a travel planning service combing air, bus, and rail transportation. They did use a REST service with OpenKapow to retrieve Amtrak results for rail portions of the search. However, this paper was published in 2008, and the Amtrak service, along with OpenKapow, is now defunct and unreachable. However, the data returned from the service is similar to what the application retrieves from Amtrak's search results. Other itinerary planning research has been conducted, but does not necessarily relate to train travel. Roy et al. proposed an interactive itinerary planner focused on points-of-interest (POIs) in a location [6]. Users provide feedback on the proposed plans and the program creates new itineraries based on their feedback. In the context of the Amtrak Rail Planner, this would be akin to users providing a list of cities to visit and the application generating potential routes and trains to take. Given the overhead of performing an Amtrak search, even with an automated browser, generating an itinerary could be very lengthy.

However, web scraping is certainly not limited to Amtrak's website. It is often the technique chosen when APIs are limiting or nonexistent (in Amtrak's case). For the former case, Twitter is a prime example [7]. The authors note that Twitter's API only allows querying for tweets chronologically no more than three weeks old. Their solution bypasses this limitation by using Twitter's search function. They also built a web service to act as a GUI for their scraping solution. A similar issue arises with Instagram's API for account data collection [8]. Since the API has access restrictions, the authors propose a web scraping solution that can bypass the API. They include a media crawler to collect the data, storage of scraped data in a NoSQL database, and a Flask application acting as the interface between users and the underlying service.

III. Problem

There are no similar software solutions available to plan Amtrak Rail Pass trips. Manual work is required to search for and document each segment and its potential trains one can take. One existing method is Microsoft Excel, but this is a means for storing the data instead of finding it. The closest solution is simply Amtrak's website. While it is the absolute resource for finding trains, anything past purchasing the trip right then is severely limited due to its inability to save results or specific trip segments. Results can also be spread over several pages, requiring users to navigate through each to see all listings. For this reason, the application may present itself as a breakthrough application in the train travel field considering its novelty and use.

The application aims to simplify the travel planning process by eliminating the need to use the Amtrak website and the associated manual work. Every station in the network will be listed in alphabetical order to eliminate any guesswork about where to visit. The origin, destination, and departure date will be selected, and the user can begin the search. After results (if any) are found, a list of trains and their attributes (departure/arrival time, number of segments, duration, etc) will be displayed and the user will select their preferred trip. This selection is saved and the user can begin their next search, repeating this process until their journey is complete. The final travel plan will be saved in a spreadsheet with all associated information about each trip, ready for later consultation. The application will feature picture displays of cities where selected stations are located to provide context. Additionally, helpful options will be available to the user, such as enabling or disabling the display of certain trip attribute columns, and supplemental information such as the Amtrak system map.

IV. Solution

A. Requirements

The application had numerous initial requirements, listed in Table I. Extra requirements were gathered from a survey posted on the Amtrak subreddit, although not all were implemented in the prototype.

B. Backend

The application's backend primarily consists of data collection and processing. There are five major areas: the Amtrak Searcher, Image Searcher, Trains, Stations, and User Selections.

1) Amtrak Searcher: The `AmtrakSearch` class has its own webdriver, used for navigating Amtrak's website and initiating searches with Selenium. It provides a public method to prepare for a search, updating station and date information, and a method to start a one-way search. Numerous private helper functions assist in entering search parameters on the site and parsing the results for train information. The core functionality of this application resides in this class, as without it, no results could be obtained from Amtrak.

2) Image Searcher: The `ImageSearch` class is responsible for searching Google for city images and providing them to the appropriate Tkinter labels. It has a webdriver separate from the Amtrak searcher. A public method to load an image is used by the `ImageArea` class, and the searcher is responsible for cropping and converting the found picture to a Tkinter interpretable object. Threading and locks are implemented in the event the user changes images too quickly, as there is only one webdriver for both origin and destination image labels. The frontend can then retrieve the stored images from the `photo_city` objects and display them. A public method is also available to perform a city swap, which simply swaps the city objects. In this case, there is no need to perform searches for existing images, so the display update is instant.

3) Train: Trains is really two classes available in one file: `Train` and `RailPass`. The former is created for each train found from an Amtrak search, and the latter stores the user's saved segments. Train objects hold such information as arrival/departure datetimes, duration, and prices. The only public method is meant to return values for use in the results or itinerary tables, as Treeview widgets demand a specific format. The Rail Pass object, of which only one is created, provides a great deal of functionality. Specifically, this class handles searches and segments. Any modifications originating from the itinerary or results table will affect the Rail Pass, as its methods for segment modification will be called. This class also provides the method to create a spreadsheet file with the itinerary, so it is critical in accomplishing the end goal of the application's use.

4) Stations: The `Stations` class is created at application launch and retrieves a list of Amtrak stations from the corresponding Wikipedia page. BeautifulSoup is used to parse the response and creates dictionary entries for each station. This method was chosen rather than including pre-fetched list

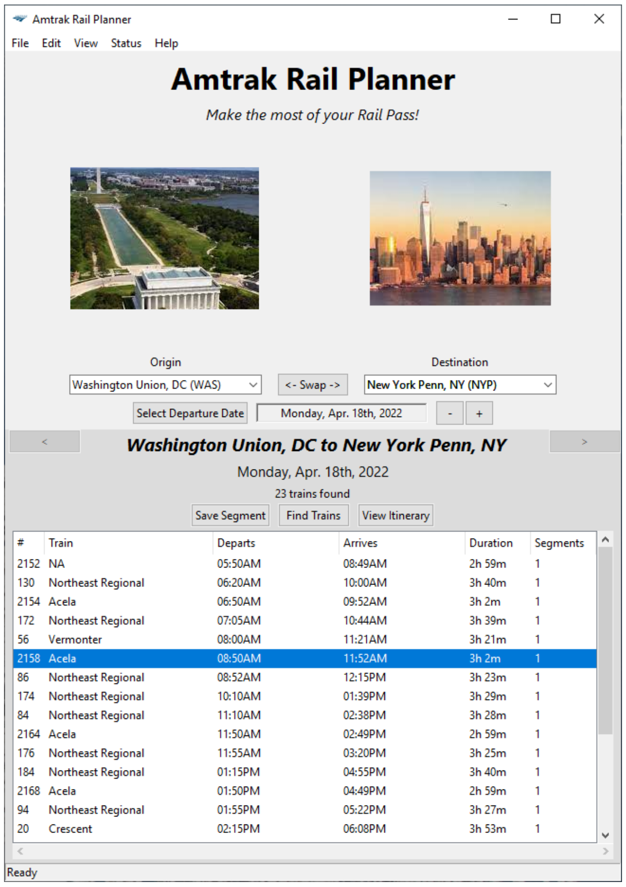
TABLE I
Functional requirements (condensed) for the Amtrak Rail Planner, sorted by priority.

R01	1	The user will be able to select the origin and destination from respective lists of stations, with the ability to search in the list if necessary.
R02	1	City photos will update themselves to a different location based on any new user station selections associated with the respective image.
R03	1	A Swap button will, when clicked, swap the origin and destination selections.
R04	1	Clicking the "Select Departure Date" button or the current date label will display a calendar selection popup. If the popup is already visible, clicking either will close the popup.
R05	1	Clicking the increment (+/-) buttons should increment the current date up or down by one day, respectively.
R06	1	A "Find Trains" button, when clicked, will begin the (background) search for trains based on selected origin, destination, and departure date. It will also create a search progress bar.
R07	1	An area for the train search results table will be visible, and, when a search is completed successfully, will populate with results and a scrollbar, if necessary.
R08	1	A heading area for the current search results, with "<Origin>to <Destination>", the date, and number of trains found, will update when a search is initiated.
R09	1	A status bar will be present at the bottom of the application, which normally displays "Ready." During a search, details of what is occurring will display to signify something is happening.
R10	1	Clicking a train in the results area will reveal an option to "Save Segment" so it can be recorded to the itinerary.
R11	1	The user will be able to export the itinerary (of some n saved segments) to a spreadsheet.
R12	1	If the Amtrak search returns an error (no service between stations, no trains on this day, etc), the user should be notified with an error popup box containing the contents of this message.
R13	1	Search validation should be performed in regards to station selection and departure date.
R14	2	Left/right arrows in the heading area will allow the user to move between all searches performed while the application is open, in case they need to check previous results.
R15	2	The user should be able to select which train attributes (columns such as departure time, price, travel time) are displayed in the search results table and exported spreadsheet.
R16	2	A popup window to list selected segments should open after clicking a "View Itinerary" button.
R17	2	The itinerary window should allow the user to reorder and delete segments, as well as view their search results.
R18	2	Clicking on the name of a train from within the results table should open an Amtrak page with details about the train, in case the user does not know its route.
R19	3	A map of the United States with pins/markers for the origin and destination should be displayed and dynamically updated as station selections change. A line should be drawn between the two points.
R20	3	With an existing app-generated itinerary, the user should be able to import this file into the app for further modification.
R21	3	Clicking on a photo of a city should bring up information about the city, such as its Wikipedia page.
R22	3	Allow the user to input each stop on their journey, and let the program figure out the travel dates.
R23	3	Display information related to each station, such as distance to the city center, availability of parking, or other transit connections.

of stations to ensure users can access every train station in the network, in the event a station was added or removed. Public methods are available to return a station's information in various string formats for different functions throughout the application. For example, the stations list uses the dictionary keys as values, while the image search requires the City and State attributes.

5) User Selections: Finally, the `UserSelections` class acts as an intermediary between multiple frontend and backend classes, storing search information and the `RailPass` object. Column information for tables is stored in this class, including the user's preferences for displayed columns. Most critical to the search feature is the validation method `isSearchOkay` which determines if the user search is acceptable based on the input stations and date, and, if not, prompts the user to continue depending on the violations.

Some frontend classes do contain logic for handling searches. Briefly, the results heading area manages navigating between previous search results and updating the results table accordingly. The stations area contains functionality for the autocomplete feature in the stations list. However, the train results area holds the ability to start a search of Amtrak's website and updates all necessary widgets in the process. A significant portion of the program's search results logic is within this frontend class.



Washington Union, DC to New York Penn, NY

Monday, Apr. 18th, 2022

23 trains found

#	Train	Departs	Arrives	Duration	Segments
2152	NA	05:50AM	08:49AM	2h 59m	1
130	Northeast Regional	06:20AM	10:00AM	3h 40m	1
2154	Acela	06:50AM	09:52AM	3h 2m	1
172	Northeast Regional	07:05AM	10:44AM	3h 39m	1
56	Vermonter	08:00AM	11:21AM	3h 21m	1
2158	Acela	08:50AM	11:52AM	3h 2m	1
86	Northeast Regional	08:52AM	12:15PM	3h 23m	1
174	Northeast Regional	10:10AM	01:39PM	3h 29m	1
84	Northeast Regional	11:10AM	02:38PM	3h 28m	1
2164	Acela	11:50AM	02:49PM	2h 59m	1
176	Northeast Regional	11:55AM	03:20PM	3h 25m	1
184	Northeast Regional	01:15PM	04:55PM	3h 40m	1
2168	Acela	01:50PM	04:49PM	2h 59m	1
94	Northeast Regional	01:55PM	05:22PM	3h 27m	1
20	Crescent	02:15PM	06:08PM	3h 53m	1

Fig. 1. Amtrak Rail Planner main window after the first search has been completed.

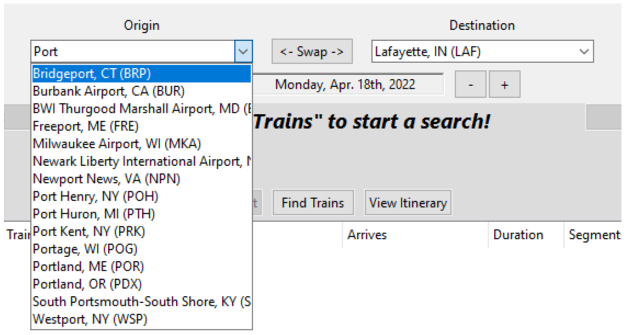
C. Frontend

The frontend classes all inherit from Tkinter Frame or Toplevel objects, aside from the `MainWindow` class which inherits from Tk (root). While they primarily hold UI elements, some backend functionality was still implemented in some classes. Each child class also accepts the root window as an argument so that other children can interface with each other. A complete view of the main window is shown in Fig. 1. At a high level, the `Ttk` module was heavily utilized due to its more modern look, despite somewhat limited customizability. Background colors for the application window were changed depending on the operating system to match the button borders. The application has been designed to guide the user down towards the search button only after they have filled out preceding fields.

1) Title Area: Only the application's name and a short message, or motto, about the Rail Pass is displayed.

2) Image Area: Photo displays of the origin and destination cities are shown. This class holds an `ImageSearch` instance to manage photo retrieval. Clicking on an image will open the city's official tourism site in most cases. However, smaller cities may direct the user to a government page. A query is sent to DuckDuckGo's equivalent of "I'm Feeling Lucky" to accomplish this with minimal user interaction.

3) Stations Area: Two combination boxes are displayed in this area with a "Swap" button between them, which will swap the boxes' selections. The user can type a query in the box and pressing Enter will only list stations with a substring of the query, shown in Fig. 2. Due to the lengthy list of train stations served by Amtrak, this feature cuts down on the amount of scrolling required to find a station.

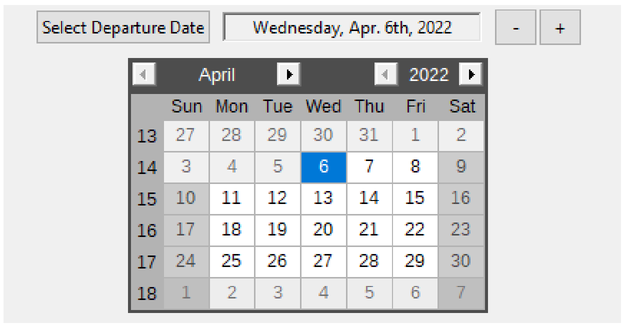


Bridgeport, CT (BRP)
Burbank Airport, CA (BUR)
BWI Thurgood Marshall Airport, MD (
Freeport, ME (FRE)
Milwaukee Airport, WI (MKA)
Newark Liberty International Airport,
Newport News, VA (NPN)
Port Henry, NY (POH)
Port Huron, MI (PTH)
Port Kent, NY (PRK)
Portage, WI (POG)
Portland, ME (POR)
Portland, OR (PDX)
South Portsmouth-South Shore, KY (S
Westport, NY (WSP)

Fig. 2. The Origin stations list shows all entries with "Port" (the user's query) in the name with the autocomplete feature.

4) Date Selection Area: The currently selected departure date is displayed in a label in this area, alongside increment buttons and a "Select Departure Date" button, seen in Fig. 3. Clicking the select button or the date label will show a TkCalendar calendar popup allowing for date selection. The user may only select a date greater than the current day's date. Increment buttons were included as most searches will only differ by one or two days.

5) Results Heading Area: A title for the currently displayed search's origin and destination, the search departure date, and



	Sun	Mon	Tue	Wed	Thu	Fri	Sat
13	27	28	29	30	31	1	2
14	3	4	5	6	7	8	9
15	10	11	12	13	14	15	16
16	17	18	19	20	21	22	23
17	24	25	26	27	28	29	30
18	1	2	3	4	5	6	7

Fig. 3. A calendar popup is available when the user clicks the ”Select Departure Date” button or the date label itself.

the number of trains found is shown in this area. Arrow buttons to the left and right of the title allow the user to navigate between previous searches.

6) Train Results Area: Buttons to view the itinerary, begin a search, and save a segment are present just below the results heading area and above the search results table. The results table displays all trains found in a successful search with the columns the user has selected. Arrival/departure columns only list the date if the train is an overnight journey or longer. Clicking a train enables the “Save Segment” button. Right-clicking a train (row) allows the user to save the segment or view information about the train on Amtrak’s website.

7) Itinerary: The itinerary is a separate window which displays any segments the user has saved, shown in Fig. 4. They can be reordered or deleted, as well as populating the results table with the search they were found in. Train information on Amtrak can be viewed, same as the train results context menu. The user may also export the itinerary to a spreadsheet (.csv) file with the attributes and location of their choice. Arrival/departure columns always list the full dates.

8) Column Settings: The user can select which train attributes they wish to see in the itinerary and results tables. At least one column must be selected. A separate column selection prompt is displayed when exporting the itinerary, for only the exported file’s columns.

9) Menu Options: Key functionality in the menu allows the user to edit display columns, view the system map, check historical on-time performance of trains, and view status maps for various regions around the United States.

V. Evaluation

A. Analysis of Results

The application, in its current state, provides travelers a way to create train travel itineraries in a much simpler fashion than current solutions. Of the 23 functional requirements in Table I, over 80% were satisfied, with the remaining unfulfilled requirements consisting of only priority level three. This means that the requirements critical to the application’s function were all implemented and met, and some of the “long-term” requirements were not. The primary goal of creating an itinerary, however, has been achieved. Along the way, the application evolved to include extra information about the trains and destinations, search validation, and saved results, all while keeping a relatively minimal design. Simplicity in the application was the goal, as people of all ages may use this application to plan their trips. An 8- or 80-year-old person should both equally be able to operate the Amtrak Rail Planner without much hassle. For this reason, the application is designed to guide the user down from station selection to the search button.

The time-saving aspects of the application, for a user of any age, cannot be emphasized enough. Two primary features, the Itinerary and saved searches, constitute most of the application’s worth. Besides acting as a liaison between the user and Amtrak, it eliminates the need for duplicate searches, were a user wanting to revisit a previous trip segment. The Itinerary ensures the user never needs to manually gather information about trains they find. The difficult aspects of train trip planning have been reduced to “point and click” work.

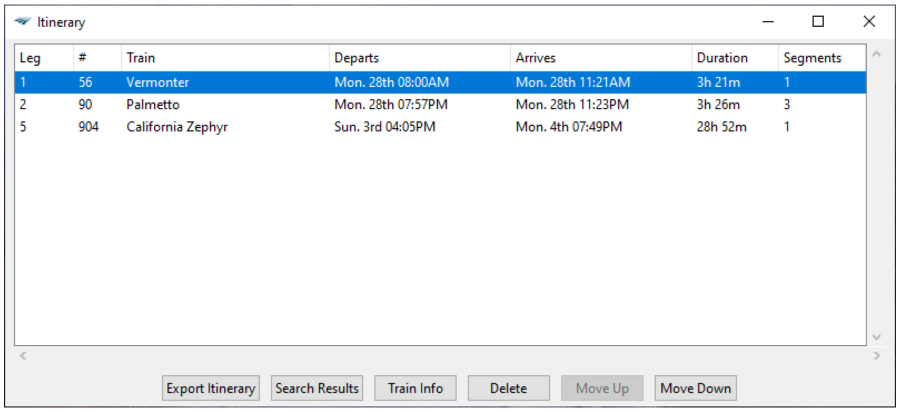
B. Limitations

The greatest inhibition towards successful app usage is failure of the webdriver when searching Amtrak’s site. The original method, simply using a headless browser, eventually failed as Amtrak blocked its use by refusing to return search results. For this reason, the Undetected Chromedriver [9] had to be used, which includes some additional logic to prevent detection by “anti-scraper” protection. Thus, a local Chrome installation is required to use the application. There is no guarantee that this solution will hold up indefinitely. Using a VPN in conjunction with the webdriver may help avoid detection as well. Alternative methods for retrieving search data were explored. Interestingly, all search results are stored as JSON data in the local session storage on the browser, which can be easily parsed by the application. However, the webdriver still needs to successfully complete a search to receive this information, and at that point, it can navigate the site as it is already programmed to.

Station information is also stored in the session storage; however, this list includes bus stations as well as train stations. While the Rail Pass can be used to travel to bus stations in conjunction with a train segment, the list of stations, at that point, may become unwieldly. Train information is also stored in a similar way, with the format `number: train_name`, but it is unclear what the use of this could be currently.

VI. Future Work and Conclusions

The Amtrak Rail Planner has interesting avenues for expansion. Highly requested features from survey respondents relate to station and train information. For example, station information could include details about the surrounding area, such as distance to the city center, available lodging or parking, places to eat, or amenities at the station. Train information includes its timetable and amenities on-board (dining car, WiFi, etc), the latter of which is available in information gleaned from the search results in the session storage. However, a significantly requested item was the inclusion of a map with the application, which would display the origin and destination


Itinerary

Leg	#	Train	Departs	Arrives	Duration	Segments
1	56	Vermonter	Mon. 28th 08:00AM	Mon. 28th 11:21AM	3h 21m	1
2	90	Palmetto	Mon. 28th 07:57PM	Mon. 28th 11:23PM	3h 26m	3
5	904	California Zephyr	Sun. 3rd 04:05PM	Mon. 4th 07:49PM	28h 52m	1

Fig. 4. The itinerary window. Two single-segment trips have been saved, and one three-segment trip is saved.

locations, potentially with a line drawn between the two. Ideally, all stops in the itinerary would be displayed on the map with actual rail lines between each, not just a straight line. In this way, the application could expand beyond a rail planning service and turn into a trip planning service, with the distinction coming from the inclusion of city information and lodging.

In conclusion, the application has provided an efficient way to plan a multi-city train trip and explore multiple potential routes. Saved search results allow for an easy review of travel options without the need to perform a search again, a significant time-saving improvement over existing methods. Train, city, and status information allows users to get the context of their journey. Itinerary views and exporting allow the user to save their trip for later, without the need to manually compile train information. While the application is an Amtrak website searcher underneath, its additional features bring it from an interface for the site to a feature-packed assistant for train travel planning.

References

[1] "Amtrak tickets, schedules and train routes." [Online]. Available: https://www.amtrak.com/home

[2] "Select your trip." [Online]. Available: https://www.amtrak.com/tickets/departure.html

[3] M. Gheorghe, F. Mihai, and M. Dârdal, "Modern techniques of web scraping for data scientists," 2019.

[4] "Multi-city tickets." [Online]. Available: https://tickets.amtrak.com/itd/amtrak/complexrail

[5] S. Navabpour, L. S. Ghoraie, A. A. Malayeri, J. Chen, and J. Lu, "An intelligent traveling service based on soa," in *2008 IEEE Congress on Services - Part I*, 2008, pp. 191–198.

[6] S. Basu Roy, G. Das, S. Amer-Yahia, and C. Yu, "Interactive itinerary planning," in *2011 IEEE 27th International Conference on Data Engineering*, 2011, pp. 15–26.

[7] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, V. Martinez-Hernandez, V. Sanchez, and H. Perez-Meana, "A web scraping methodology for bypassing twitter api restrictions," 2018. [Online]. Available: https://arxiv.org/abs/1803.09875

[8] A. Himawan, A. Priadana, and A. Murdiyanto, "Implementation of web scraping to build a web-based instagram account data downloader application," *IJID (International Journal on Informatics for Development)*, vol. 9, no. 2, p. 59–65, Dec. 2020. [Online]. Available: http://ejournal.uin-suka.ac.id/saintek/ijid/article/view/09201

[9] Ultrafunkamsterdam, "Ultrafunkamsterdam/undetected-chromedriver: Custom selenium chromedriver: Zero-config: Passes all bot mitigation systems (like distil / imperva/ datadadome / cloudflare iuam)." [Online]. Available: https://github.com/ultrafunkamsterdam/undetected-chromedriver