University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2022

Walmart Tech Bar Scheduler

Ben Hodges, Joe Tam, Austin Dixon, Cole Alvarado

Abstract

Walmart's currently designed IT (Information Technology) portal named "MyTech" is great for solving technology issues small enough to be done through instructions online. However, there are times when issues arise that are so complicated that an expert in the field must inspect the hardware physically. To solve this problem Walmart has implemented "Tech Bars" in many locations at which associates can bring their hardware with issues and tech support technicians can fix those issues. Our aim is to improve their Tech Bars by creating an end-to-end Tech Bar scheduler Web Application for Walmart associates and tech support technicians. This will allow associates to get into a queue at a Tech Bar location of their choosing and allow tech support technicians to remove associates from the queue when a resolution is reached.

Our approach to this is creating a web application that allows associates to sign in, find the nearest Tech Bar to their location, join the queue at that location, and see when they will be able to be helped by an IT professional at that Tech Bar. Technicians will be able to sign in, edit the Tech Bar locations, operating hours, status, as well as the current queue with the ability to add or remove associates. This web application has a large significance because it will improve the efficiency of Walmart's Tech Bars. By allowing online check-in, associates will know where they are in the queue and can thereupon know when to arrive at their chosen Tech Bar. This will alleviate the problem of a physically backed up line at a Tech Bar's location and save the time of both technicians and associates.

1.0 Problem

Walmart is creating and improving an IT portal named "MyTech." This IT portal allows Walmart associates to resolve Tech related issues that are minor enough to not require a physical hardware inspection. In the case that a physical hardware inspection is required, a Walmart Tech Bar will be needed. A Walmart Tech Bar is a walk-up desk at which associates can bring their physical technology that is having issues and tech support technicians can perform diagnostics on that device and solve issues that could not be solved by their traditional IT portal.

Physical Tech Bars have an issue that is not prominent in digital ones, physical lines. If an associate of Walmart is working and needs to go to a Tech Bar to fix an issue but there are others there with the same problem, they will have to wait and waste time that could be used doing more productive tasks. For a company like Walmart with over 2 million associates the possible time wasted on associates waiting in a physical queue is huge. There is also the possibility of an associate showing up to a Tech Bar that is closed which would also waste the time of that associate greatly. The MyTech portal will help resolve this issue by giving more detailed and clear solutions to tech support issues, but there will always be the need for certain issues to be solved by tech support technicians in person.

This problem also affects the technicians themselves. Without a digital check-in for their Tech Bars, technicians cannot be prepared for the issues that are going to be brought to them. This can also waste time because every person that comes up to the Tech Bar will have to have their problems assessed first while a digital check-in can allow that step to be done beforehand. This can improve the overall experience of technicians and implementing a digital check-in would improve both the associate experience and technician experience simultaneously.

2.0 Objective

The goal of this project is to create an end-to-end Tech Bar scheduler web application for Walmart associates and tech support technicians. This web application will be accessible through the MyTech platform but starting out will be an entirely separate application. This application will greatly help both associates and technicians by making the process of checking into and managing a Tech Bar a remote process which will save the time of associates who will no longer need to wait in a physical line and the time of the technicians who will be prepared for every associate who arrives at their Tech Bar. Most of this web application will be designed by the Capstone group with UI/UX input from the Walmart team. Within this wide objective we also have minor goals that will improve the feel and experience of the web application.

Within the web application associates should be able to see Walmart Tech Bar locations. These locations will have information associated with them telling the associate the Tech Bar's business hours and the real time status, that being whether it is closed, busy, or open. This will ensure that associates do not go to a Tech Bar that is closed, and it also allows the associate to choose to go to a Tech Bar at a less busy time if their technology issue is not pertinent. This will also help technicians if a Tech Bar closes temporarily during normal business hours because they can communicate to the associates that it is closed and ensure that no one will go there during those times.

These locations will also hold latitude and longitude values. When an associate signs in, their rough location will be pulled by the web application and using a calculation the application will allow the associates to show the Tech Bars that are closest to them. For the number of Bars currently open this function is not necessary, but over time as more open, this functionality will be needed so that associates do not need to manually search through the list of Bars to find one in their city.

Once finding a Tech Bar in their area, the web application should allow associates to enter a queue for that location. When an associate enters the queue, a technician at that location will be able to see information entered by the associate and be prepared for that associate's arrival. The queue will also allow associates to see what position they are in line and an estimated wait time for when they will arrive at the front of the queue. The page for the queue will also allow associates to leave it in case they no longer have time to get help at a Tech Bar.

Technicians also will have control over the queue at their Tech Bar. When a technician is free to help the next person at their location, they will be able to see every associate currently in the queue and pull out whichever one they choose and open a "ticket" for them. This "ticket" will represent an associate currently being helped and when they are finished being helped the technician will be able to close the "ticket." Closing a ticket will enter information into the database and allow technicians to see statistics of the number of tickets they have closed over different time periods at their location.

To ensure that technicians have an elevated level of control over the queue, when a new technician is entered into the database the admin of the system will raise a flag on their database entry which allows the application to know that they are a technician. Once a technician signs in on the application, the technician flag will be noticed, and they will be directed to the technician side of it. This flag will also ensure that associates are not able to get into the technician side of the application and will instead be redirected to their side if they try.

Our first optional goal is to implement a mobile view of the web application. Since many associates will have to travel to their closest Tech Bar, those users will most likely be checking into the queue and checking their place in the queue from a mobile device. Supporting a mobile view of the application would then be very advantageous because it would improve the UX greatly.

Another optional goal designed to improve user experience is support for dark mode. A lot of people prefer a dark mode on their devices because the brighter colors of usual web design can be irritating to the eyes. Designing a dark mode can also be a fairly simple process so with a small amount of extra time we hope this objective can be achieved.

Our last optional goal for this project is to give the associates the ability to reschedule their Tech Bar appointment for a different time or location. This would be a quality-of-life improvement for associates because commonly when making appointments another obligation may halt them from being able to travel to the Tech Bar. They also could end up closer to a different Tech Bar before the appointment and changing it to that one would be preferable. To solve this problem, we would prefer to implement the ability for associates to change the time and location of their appointment.

3.0 Background

3.1 Key Concepts

To build the implementation of the MyTech web application we will be using Ruby on Rails on the backend. This framework will allow mainstreamed communication with the PostgreSQL object-relational database which will allow the web app to be a better candidate for scalability and efficiency in the future. For the frontend we will be using React and JavaScript for web elements the user will see or interact with. Also, to be used are GraphQL and Apollo integration within the React framework which will be used to handle API requests. Other notable key components include HTML/CSS to give structure to and support the style of the system. These all will be used in tangent to make the user experience seamless and accessible for all users.

3.2 Related Work

Check-in systems like what we are building are prominent across the country. Stores like AT&T and Chick-Fil-A use them, and many even display the current queue on monitors around the store. Applying those similar systems' ideas to ours however would not work because they are far too specific for the place they are applied to. When looking at open-source or proprietary systems like ours we run into the opposite problem. Those systems are far too general to perform the tasks needed for this project.

One such system where that issue prevails is the Kinetic Tech Bar [1]. This system has similar features to the ones we are tasked with implementing. Those include advanced appointment

scheduling, in-person servicing, and a view of the user's position in the queue. The problem supplied above is also applicable to this system. Kinetic Tech Bar is far too broad of a system to be applied to Walmart's Tech Bars. This is because while it does have many of the same features that we will implement, there is an equal amount that it does not. One example of this is the control that tech support technicians have over the system. While the Kinetic Tech Bar does allow technicians to manipulate the queue, it does not allow them to manipulate any data related to their own Tech Bar. It is crucial in our system that the associates know whether the Tech Bar they are trying to schedule an appointment for is open at the time of the appointment, but the Kinetic Tech Bar has no such implementation. This system also has many features that would be useless to what we are building such as data aggregation.

Another system like ours is KASPA [2]. KASPA is a system that allows users to request a tech repair, drop it off in a locker unlocked using a corporate ID card, and then go back to that locker sometime later to pick up the fixed tech. Like Kinetic Tech Bar, KASPA has many features like our system but also a lot of differences. With KASPA when you have tech problems you can open the system and view the closest Tech Bar to you, which is a feature we hope to implement in our system. The main problem that arises with KASPA is that when you drop your technology off it will take a day or longer for it to be returned. For a company like Walmart, having associates unable to work a day or more would cost them a lot of money and significantly reduce efficiency especially with the substantial number of associates that they employ. One part of KASPA that we could use is their touchless system. While we are still amidst a global pandemic touchless methods of contact are preferred to physical. Implementing a way to make our system support touchless delivery would be greatly beneficial.

Lastly the Vizitor [3] application also shares similarities with our application. Vizitor is an application designed for meetings that allows users to check-in to a meeting before it, check-in when there physically, and notify the host of the meeting when they arrive. This application has similar features such as digital check-in, but since it is designed specifically for meetings it is harder to apply to our system. This system has the problem of not having location features. Without this it could leave the user confused as to where to go for their meeting. Our system solves this problem by having location support, and it gives the associates the location of the Tech Bar they are going to.

Overall, there are many systems in existence that share similarities with the web application we are implementing but none of them are good enough to help Walmart with exactly what it needs.

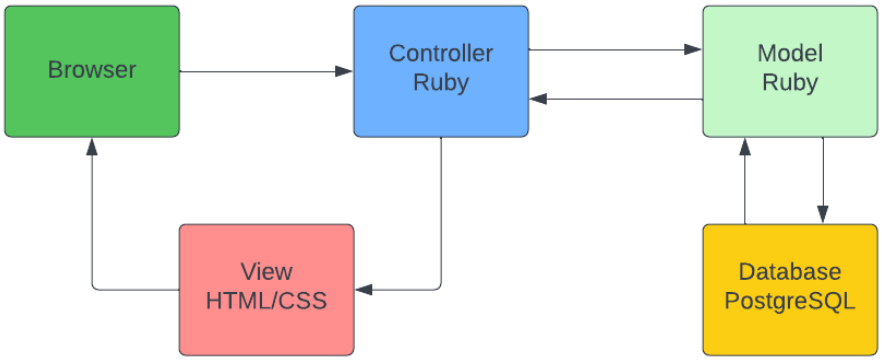
4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

- Create an end-to-end Tech Bar scheduler web application for Walmart associates and tech support technicians.
- Associates should be able to see all Tech Bar locations, their business hours and real-time status (open, closed, busy, etc.)
- Associates should also be able to get in queue at Tech Bar locations of their choice.
- Tech Support technicians should be able to configure Tech Bar locations, their business hours and status.
- Tech Support technicians should be able to view the queue and remove associates from it with a resolution.

- The ability to configure or change Tech Bar information and remove associates from queue should be password protected.
- Optional: The web application should get the associates' current location and recommend the nearest Tech Bar.
- Optional: The web application should have mobile responsiveness.
- Optional: The web application should have support for a dark mode.
- Optional The web application should allow associates to reschedule Tech Bar appointments.

4.2 High Level Design



Database

Our database uses PostgreSQL which works well with Ruby-on-Rails because migrations and changes to the database are made easy through the rails structure. Our database consists of 4 different tables. These tables are Users, Bars, Lines, and Tickets. These tables have relations to each other. First, Users has a One-to-One relationship with Lines and a One-to-Many relationship with Tickets. This is because when the user enters the queue, they are put into the Lines table and each user can only be in one queue at a time. A One-to-Many relationship applies to the Ticket table because each user can only have multiple closed tickets. The next relation we have at the table level is that Bars has a one-to-many relationship with Lines and Tickets. This relation is how we track the queue size of each Bar because each ticket and user in line will have a bar id associated with them such that the technician can view those values specifically for their Bar and the queue associated with each Bar can stay in the correct order. Below are how those relations are represented in Ruby-on-Rails along with a diagram that visualizes them.

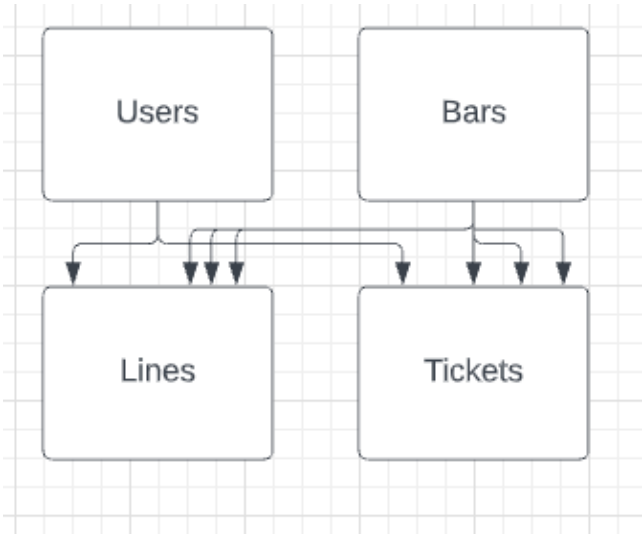
```ruby
class User < ApplicationRecord
    validates :username, uniqueness: true
    has_one :line
end
```

```ruby
class Line < ApplicationRecord
    belongs_to :bar
end
```

```ruby
class Bar < ApplicationRecord
    has_many :lines
    validates_presence_of :name
end

```

```ruby
class Ticket < ApplicationRecord
    belongs_to :bar
end
```



The Users table in the database holds information about the users in our web application. The first column implemented in this table is the string “username.” This column holds the username chosen by the user when they sign up to our web application. To avoid errors, no two usernames can be the same in this table. The next column implemented in this table is the string “password_digest.” This column holds the password that the user chooses when they sign up to our web application. This is encrypted to ensure a large amount of security in our system. Our last column in our table is the Boolean “admin.” This column holds a Boolean value that shows whether the user that is logging in is a Tech Bar technician or not. If they are, this value is true and if they are not, it is false. This will allow technicians to be routed directly into their view when logging in so that they can choose the bar they are working out of. Other than the ID, which is automatically generated, these are the three columns we have in our User table, and this implementation is below.

```ruby
class CreateUsers < ActiveRecord::Migration[7.0]
  def change
    create_table :users do |t|
      t.string :username
      t.string :password_digest
      t.boolean :admin, default: false
      t.timestamps
    end
  end
end
```

The Bars table in the database holds information about the Tech Bars that are in the system. The first column in this table is “name” which holds the name of the Tech Bar. The next column in this table is “address” which holds the address of the Bar. Along with address there are columns for “city,” “state,” “country,” and “zip.” These 5 columns hold the locational data of the Bar which will allow users to know where they need to go when they enter that Bar’s queue. The next column that the Bar table holds is the “status.” This column will show whether the Bar is closed, open, temporarily closed, or busy. Next are the “openTime” and “closeTime” columns which hold when the bar opens and closes. The next in this table is the “size” column. This holds the current size of the queue to allow technicians to see the size of their queue along with allowing the user to choose a queue with a smaller size if they have more than one Bar in their area. Lastly, we have latitude and longitude. When the location of a new bar is input into the system a geocoder in our application will automatically calculate this using the location columns. The implementation of this table is below.

```ruby
class CreateBars < ActiveRecord::Migration[7.0]
  def change
    create_table :bars do |t|
      t.column :name, :string, :null => false
      t.column :address, :string, :null => false
      t.column :city, :string, :null => false
      t.column :state, :string, :limit => 2, :null => false
      t.column :country, :string, :null => false
      t.column :zip, :string, :limit => 5, :null => false
      t.column :status, :string, :null => false
      t.column :openTime, :integer, :null => false
      t.column :closeTime, :integer, :null => false
      t.column :size, :integer, :null => false
      t.column :latitude, :float
      t.column :longitude, :float
      t.timestamps
    end
  end
end
```

The Lines table of the database holds the information about each user in the queue for a specific Bar. The first column in this table is the “bar_id.” This column shows which Bar the user is currently in queue for. The purpose of this is to allow technicians to only see the queue of the specific Bar they are managing. The next column in this table is “spot.” This value holds the place in line of the user at the Tech Bar they are currently in queue for. Then the next column is

“user_id.” This column holds the id of the user that is in the queue to keep users from entering more than one queue or entering the same queue twice. We also have the columns “name,” “device,” and “issue.” These columns hold information that is prompted from a user when they try to enter a queue. It is then shown to technicians which allows them to be prepared for users that are arriving to their Bars in the future. The implementation of this table is below.

```
class CreateLines < ActiveRecord::Migration[7.0]
  def change
    create_table :lines do |t|
      t.column :bar_id, :integer
      t.column :spot, :integer
      t.column :user_id, :integer
      t.column :name, :string
      t.column :device, :string
      t.column :issue, :text
      t.timestamps
    end
  end
end
```

The Tickets table of the table holds the information about the tickets for each of the Bars. The first column in Tickets will be “bar_id” which holds the bar that the ticket was created at. The next column is “user_id” which holds the user that the ticket was created for. After this we hold “name,” “device,” and “issue.” These hold the values passed from the Line table to the Ticket table. We then have a “status” column which holds whether the ticket is open or closed. The last relevant column we have is “dateResolved.” This column holds the time and date of when the ticket was closed. It is used to display the statistics that are present on the technician side of the application.

Controllers

To control and manipulate the database we used a series of functions inside of controllers written in Ruby. Many of these functions have a corresponding page in the interface and if not then redirect to another function that has a corresponding page. These all use the active record query interface in Ruby-on-Rails which executes SQL queries in Ruby-on-Rails. These queries are many times simpler than writing actual SQL queries and make reading and writing these functions much easier. The main functions we currently have implemented on the associate side of our project are list, show, createSpot, enterQueue, update, and leave.

The list function is used for the associate view of the Tech Bars. In the controller the function is basic. This function just pulls a list of every Tech Bar currently in the database that the view can use to display them.

The show function is what the user sees when they enter the queue for a Bar after inputting relevant information. It pulls the user’s line information from the Lines table in the database and the user’s Bar information from the Bars table in the database. The line information is used to display the spot in line the user is currently in, and the Bar information is used by the html to display the information about the Bar the user is currently in queue for. This function also helps the html display correct grammar based on information obtained in the database.

The createSpot and enterQueue functions are both very intertwined. When a user enters a queue, they are brought to a page where they enter data before being put into a queue. This page is

connected to the enterQueue function which creates a new line spot. When the user finishes entering the data and presses the button, the data that the user has entered in the page is then passed into the createSpot function. The createSpot function first checks if the user id is already in the Line table and redirects to the spot they are already in, if they are. This assures that the same user cannot enter the same queue twice or enter a different queue while already in one. If the user is not already in the Line a new Line row is created with the information the user inputted in the enterQueue page, and if the line is correctly saved then the user is redirected to their spot in the queue. If the user is being added this function also calculates their spot in the queue.

The update function is used after the leave function is called and updates the queue of the specific Bar that has been left. When the user leaves a queue the spot of every user behind them in that queue needs to be updated to move ahead. This function ensures that by getting a list of rows that have users that were behind them in the queue and incrementing their spot in line. This function then redirects to the list of Bars.

The leave function is used for when the user leaves the queue that they are in. This function finds the queue the user is currently inside of and then deletes their row from the Line table. If the deletion happens, the size attribute of the Bar the user was in is then incremented down and the application is redirected to the update function described above.

Along with functions on our associate side, there are also crucial controller functions on the technician side. These functions are chooseBar index, openTicket, closeTicket, createTicket, and changeStatus.

chooseBar is the function of the first page the technician is taken to when logging in. This page gives the technician the full list of Tech Bars and allows them to choose one. It also displays information about the bars so that the technician can choose the correct one they are assigned to. This information is pulled from the database in the function by creating a list of every Bar record.

The index function is used for the main page of the bar that the technician chooses. It pulls the information out of the database about the technician's bar and the currently open tickets at that bar.

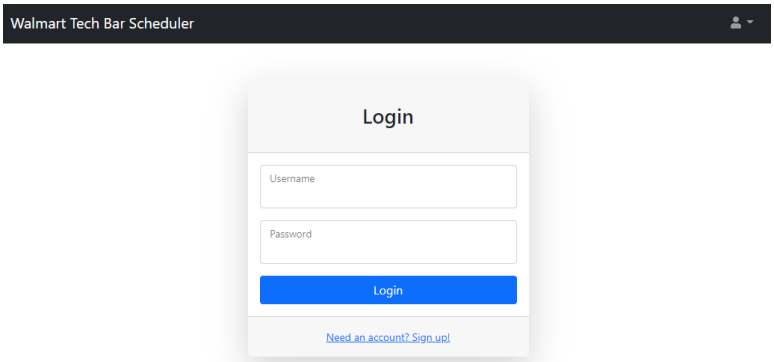
The openTicket function is used when a technician wants to open a new Ticket. It pulls every user currently in the queue of the technician's Bar out of the database. The closeTicket function has a similar role but it pulls the currently open tickets instead of the queue.

The createTicket function takes data from a record in the Line table and uses it to create a record in the Ticket table. The Line table entry is then destroyed, and the new Ticket record is represented by a new open ticket on the main page. The changeStatus function is simply used to change the status of the Tech Bar to closed when a button is pressed by the technician.

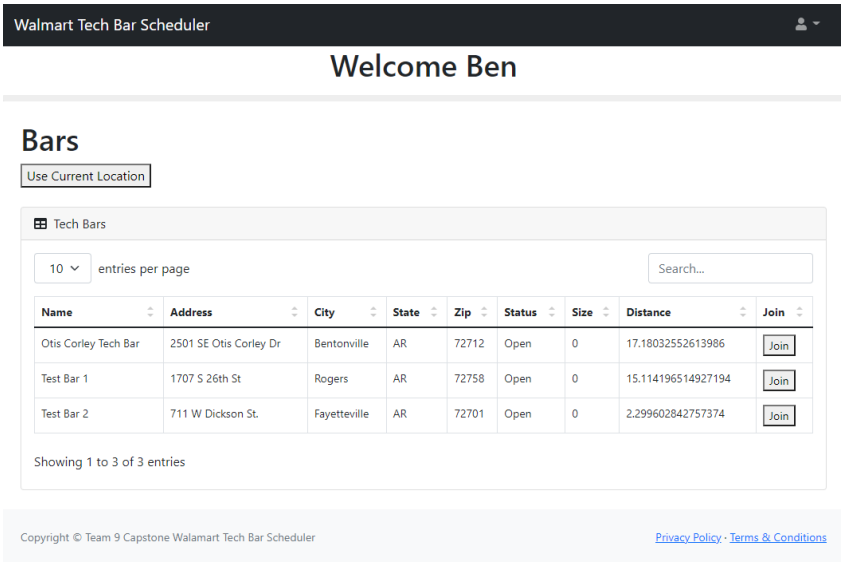
Interface

For our view of our system, we have html/CSS along with JavaScript working with our Ruby functions. There are many pages in our application, but on a higher level the 3 main parts we have are login, the associate side, and the technician side. For the explanation we will go in the respective order.

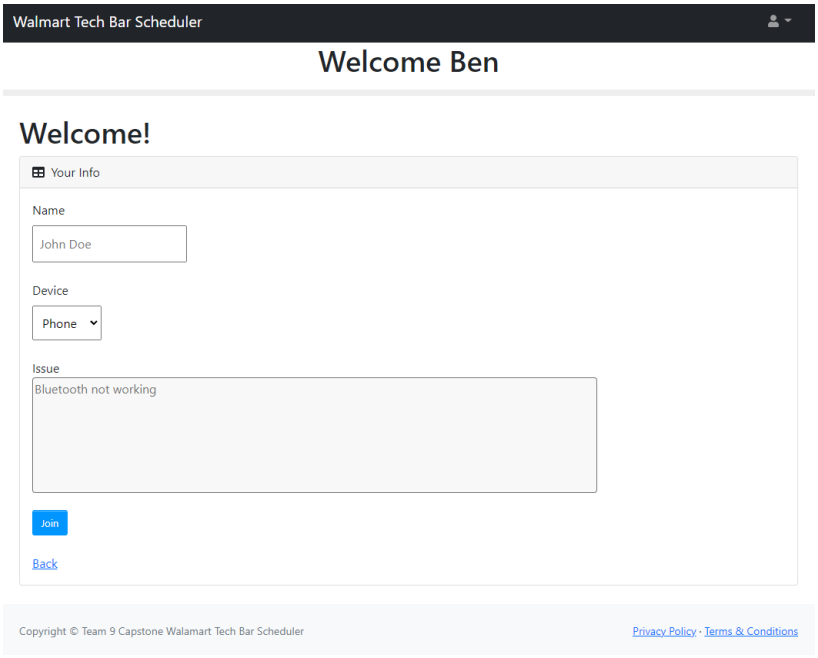
The first page the user will see when opening the web application will be the login screen. Our login screen is a basic login screen you would see in most modern web applications. It has a box for username, a box for password, and a sign-up button. Our login system encrypts the password for the user and using a flag in the User table the user is directed to either the associate or technician side of the application.



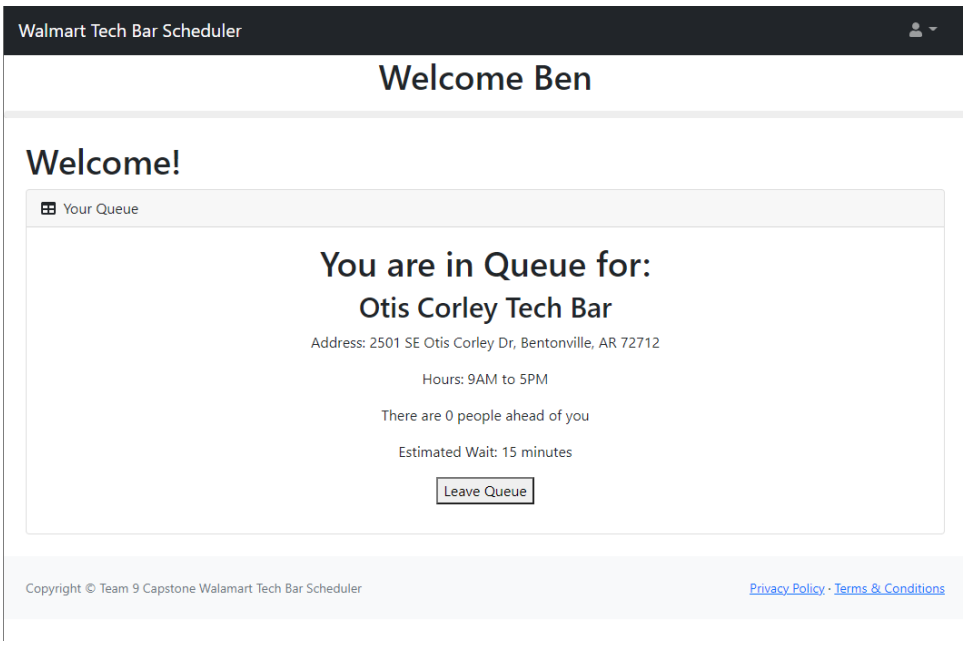
If the user is an associate, the next page the user is brought into is the list. This is the page where associates choose which Tech Bar they would like to enter the queue for. The list of Tech Bars holds the locational data of the Bar, the amount of people currently in the queue for the Bar, and a button that allows the user to join that queue. The view of this page is below.



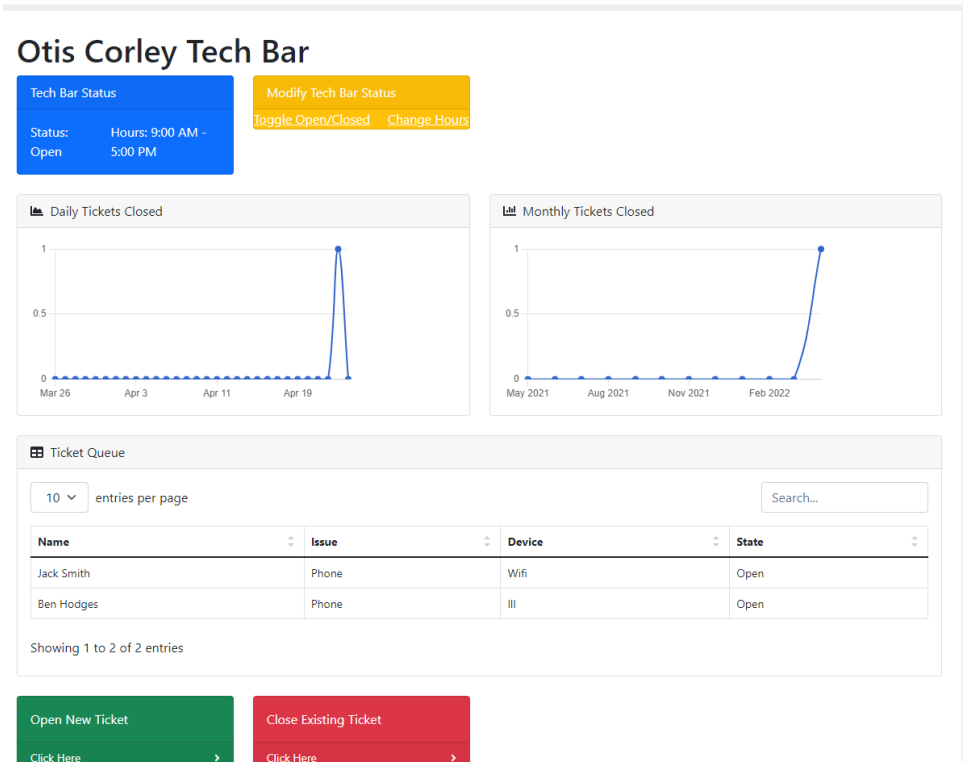
When the user clicks the join button to join a queue for a specific Tech Bar, they are then taken to the enterQueue page. The enterQueue page allows the user to enter data about their issue they are going to the Tech Bar for. The name box is where the user will enter their preferred name. The device box is where the user will enter the device they are having issues with. The problem box is where the user will enter a small description of the issue on their device. When the user has entered this information, they can then click the create button to fully join the queue.



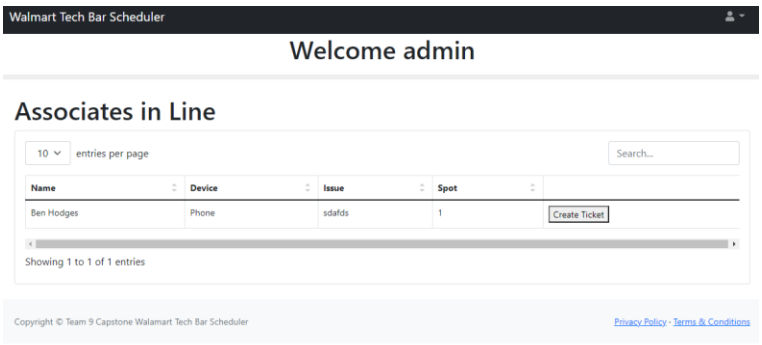
After fully joining the queue, they will be brought to a screen that shows them the current information about the queue, their place in line at that queue, and a prediction of wait time based on their place in line. We also have error checking to make sure the user does not join another queue while already in one, and if so, they are brought to an alternate version of this screen. The main screen for this is below.

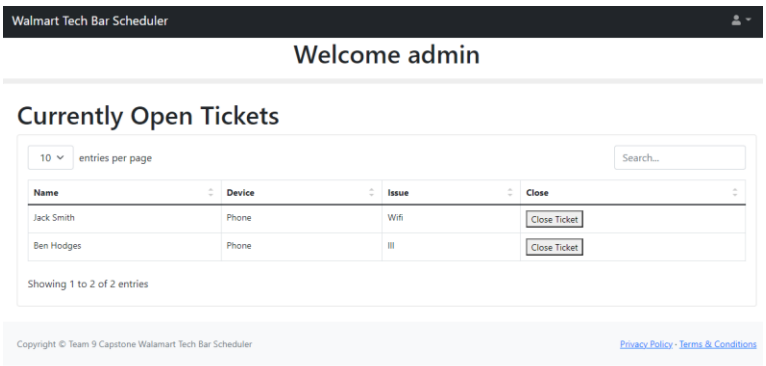


When the technician logs in they are brought to a screen that is basically the same as the page where the associate chooses what bar they are entering the queue for. However instead of being brought to an inqueue page after choosing one, the technicians will instead be directed to the main page of their bar. This page holds the users with currently open tickets and buttons to manage the queue of their Bar and the status of their Bar. This page is below.



If the technician then decides to either open or close a ticket, they are brought to the two pages below to do that.





External Technologies

Our web application uses 2 outside sources to help function. The first one is a bootstrap template that we based our html on. Starting out the project a lot of our implementation was done on basic white html screens to get the functionality to where we wanted it. We then integrated that functionality into a bootstrap template to improve our UI/UX. This template helped greatly because the work we had done in functionality was able to translate well into it and it is how we got our working demo. Within this template was also a JQuery which is what makes the data tables visually appealing.

Future Work

Since every functionality we were tasked with was finished in the result of our project, technically we personally do not have any future work. However, there are still improvements that could be made to this project if we were tasked with continuing development.

One change that could majorly improve the application is the improvement of our location function on the associate side of the application. Our current implementation of it only gets the longitude and latitude of the city that the associate is in which is helpful to find generally what bars are closest to them but getting the exact location would lessen confusion on our table. This was something we set out to do, but we found that we did not have enough technical knowledge to fully implement it.

Another improvement is security. Since it was many of our first times working in Ruby-on-Rails it is inevitable that we missed some general security practices that a more experienced developer might have implemented. So, if someone in the future revises our application or utilizes it, it may need some improvement on that front.

4.3 Risks

Risk	Risk Reduction
A malicious user getting into the technician side of the application and causing issues with the database.	Our password system will be single-sign-on and encrypted to ensure that only authorized technicians can access their side of the application. Passwords can be changed in case of a breach in security, and an admin will be able to remove compromised users.
A Tech Bar experiences an unforeseen event such as a fire or power outage closing it.	Technicians will be able to change the status of their Tech Bar at any time, and one of those statuses will be “Temporarily Closed” for events like these. This web application will also be accessible through a mobile device so in the case of a power outage the status can still be changed.
An associate having to suddenly cancel their appointment due to an event.	Our application will support the cancelling, changing of location, and rescheduling of the appointment.

An associate is unable to find where to access the Tech Bar check-in system	The webpage will be an embedded link in the MyTech tech support system, so if the associate knows how to navigate there then they will be able to find our system.
The queue has built up so large that the technician needs to stop taking people into the queue	A status for the Tech Bar labeled "Currently Busy" will be implemented for when events like this occur.
An associate's tech issue is much larger than expected and will take more time than expected.	Technicians are given the ability to add estimated time to the queue at any point to minimize this problem.
Associates not knowing how to correctly diagnose their technical problem on check-in	Our check-in system will be designed to be as easy to use as possible for non-technical users, including drop down menus and suggestions.
Technicians resolve the issues of associates at top of queue much faster than expected.	The web application will notify users when they are getting close to their scheduled time or when they are getting close to the front of the queue.

4.4 Tasks –

1. Work with our Walmart sponsors to review the requirements of the project and revise them until we have a full understanding of what we need to implement, and they understand what to expect from us.
2. Prepare a project plan similar to this proposal but specifically for our Walmart sponsors based on their guidelines.
3. Design a Wireframe mockup for the view of both sides of our system.
4. Create the database backend
5. Create the Associate Side's front end
6. Create the Technician Side's front end
7. Integrate the Login into both sides of the application
8. Final testing and bug fixing
9. Write source code documentation and user guide
10. Final Deliverable

4.5 Schedule –

Tasks	Dates
1. Work with our Walmart sponsors to review the requirements of the project and revise them until we have a full understanding of what we need to implement, and they understand what to expect from us.	11/18-1/15

2. Prepare a project plan similar to this proposal but specifically for our Walmart sponsors based on their guidelines. This will apply many of the descriptions from this paper into a different format to be more easily digestible.	1/18-2/1
3. Design a Wireframe mockup for the view of both sides of our system. This will be a basic PowerPoint template which we will use to guide the view of each page used in our application. We will then consult Walmart about our Wireframe and make revisions based on feedback.	2/1-2/15
4. Create the database backend. This will be written using PostgreSQL as our database but using migrations written in Ruby. This will also include writing seeds that will be used during testing.	2/15-3/1
5. Create the associate side frontend. This will involve first writing basic HTML to construct the view of this side, and then moving to writing controller functions in Ruby which will pass data from the database into the webpage.	3/1-3/8
6. Create the Technician Side's front end. We will start by writing HTML code to form the view based off the wireframe, and then move on to writing the controller functions in Ruby to display data from the database.	3/8-3/15
7. Integrate the Login into both sides of the application. This will involve developing functions to encrypt user passwords while also checking which side of the application the user needs to be directed to based off of their status in the database.	3/15-3/29
8. Final testing and bug fixing. This will involve testing to ensure that our security measures work as well as ensuring all the data in the database is manipulated properly.	3/29-4/15
9. Write source code documentation and user guide	4/15-5/2
10. Final Deliverable	5/2 - 5/5

4.6 Deliverables –

- Design Document: Holds a listing of each major feature of the website.
- Database scheme and initial data: The DB schema for the database holding mainly data for the users, agents, and tickets that are sent in and completed.
- Web site deliverable: working framework for web application

- Final project and report

5.0 Key Personnel

Ben Hodges – Hodges is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses. Mobile Programming is also a course enrolled in and could help if any ports would be necessary. He also has experience developing software in groups in hackathons and in a software development course.

Cole Alvarado – Alvarado is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. Relevant experience includes two summers as a software intern at Cerner Corporation as well as currently working for the University of Arkansas Athletics IT department which will supply valuable information on how this website can be best organized for both the customers and the agents.

Joe Tam - Tam is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He is currently enrolled in Information Retrieval which is about the creation of a search engine which should be beneficial for this project.

Austin Dixon - Dixon is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. Relevant experience includes working as an infrastructure intern at Tyson Foods in which he oversaw the creation of databases and front-end table accessibility which will prove highly informative during this project.

Sarah Friesell, Industry champion – Sarah is a Senior Technical Project Manager at Walmart.

6.0 Facilities and Equipment

No equipment or facilities needed other than the hardware we already own.

7.0 References

[1] Kinetic Tech Bar

https://www.kineticdata.com/kinetic-data-tech-bar

[2] KASPA

https://www.ktsl.com/kaspa/

[3] Vizitor
https://www.vizitorapp.com/