Wi-Find

Spring 2024

Group 9: Walid Abdullahi, Kevin Morales Funes, Kelsi Hill, Lisa Nguyen, Thomas Sigler

March 18th, 2024

Section 1: Planning, Schedule, and Peer Evaluation

Name	Email	Task	Duration (Hours)	Dependency	Due Date	Evaluation
Walid Abdullahi	Wabdullahi0529@gmail.com	1. Help Set Up Database 2. Behavior Modeling Diagrams 3. Test Cases 13-15	1. 2 hr 2. 1.5 hr 3. 1.5 hr (5 hr)	1.DB schema decision and project setup 2. N/A 3. N/A	1. 2/28 2. 3/18 3. 3/18	66%, Lisa did Test Cases 13-15
Kevin Morales Funes	Kevingustavo.kmf@gmail.com	1. Frontend Setup, Styling 2. Test Cases 7-9 3. Front end logic to server and debugging	1. 12hr 2. 1hr 3. 4.5hr (15.5hr)	1.VS Project Setup 2. N/A 3. API works	1. 3/18 2. 3/18 3. 3/18	100%
Kelsi Hill	2016k.hill@gmail.com	1. Test Cases 10.1-12.2 2. Frontend setup, Styling 3. Testing Frontend functions 4. Proofreading, and revision	1. 0.5hr 2. 8hr 3. 1hr 4. 0.5hr (10hr)	1.N/A 2.VS Project setup 3.Frontend setup 4.N/A	1. 3/18 2. 3/18 3. 3/18 4. 3/18	100%
Lisa Nguyen	Nguy3n.lisa@gmail.com	1. Set Up VS project 2. Database Schema, migrations, dummy data 3. Added more backend logic programming 4. Test Cases 4-6 5. Testing and debugging client and server integration	1. 2hr 2. 8hr 3. 12hr 4. 0.5hr 5. 4hr 6. 0.5hr (27hr)	3.Backend Logic set up 5.Front end functions available	1. 2/28 2. 3/18 3. 3/18 4. 3/18 5. 3/18 6. 3/18	100%

		6. Updated DB diagram				
Thomas Sigler	ts309435@gmail.com	1. Set up Backend Logic Programming - Listing, Login, Feedback, Tickets 2. Test Cases 1-3 3. Build Script 4. Github Screenshot/Setup 5. Class Diagram 6. Debugging	1. 4 hr 2. 1 hr 3. 2 hr 4. 0.5 hr 5. 1 hr 6. 2 hr (10.5 hr)	1.Database Setup– Partially Complete 2. N/A 3. Backend and Frontend Partially Set up 4. N/A 5. N/A	1. 2/28 2. 3/18 3. 3/18 4. 3/18 5. 3/18	100%

Section 2: Revised Problem Statement

Our product, at a high level, is a platform that connects users who want to rent out their internet connection (renters) to users who want to temporarily rent internet connections (rentees); the product is an e-commerce platform for a specific subset of products.

Those who want secure Wi-Fi on-the-go but do not need constant access to Wi-Fi will benefit from this product. Furthermore, users who have specific Wi-Fi criterias or need internet access but only have extremely limited and exorbitant viable options will find this product desirable. On the renter's side, this product will open a tangible opportunity to those who want to make extra money without breaking a sweat via renting out their Wi-Fi to others.

Our product presents a possible solution that is both elegant and profitable to the issue of free public Wi-Fi which is often unsecured as well as the issue of obtaining individualized Wi-Fi needs at an affordable rate.

Obvious alternatives are currently available that are viable competitors to our product. Many restaurants, cafes, and hotels now offer free Wi-Fi to their patrons; these competitors however, often do not enforce secure Wi-Fi. Internet Service Providers (ISPs) are the next upgrade from free public wifi and offers permanent Wi-Fi for a monthly fee; they are the largest competitor to our product but do have drawbacks. ISPs are far more expensive than what we expect our product will provide to consumers. They often have fine print contracts that binds users who may desire pay-as-you-use. Furthermore, ISPs availability varies by location, so a user's power to choose a rate and quality that suits them is limited if available ISPs in the area are those with pricey plans. For example, 5G with certain providers are not possible if their 5G tower is not in the area. Fiber also suffers from the same issue of physical limitations (no fiber underground in

areas where it is impossible to dig them). DSL (like NetZero) is much more affordable for slow internet but is not available in areas with no telephone poles. Breaking past ISPs that are limited physically or constructionally would be satellite internet such as ViaSat or StarLink, both exorbitant alternatives that each have their respective drawbacks. ViaSat is expensive and only as fast as DSL while StarLink's internet speed is sensitive to obstructions and vary depending on location. The final possible alternative is to commute to an internet cafe or establishments similar to it which could cost much more in both time and money just for commuting.

This project is compelling and worthy of developing since it could present a viable and competitive alternative to using free, unsafe public Wi-Fi, committing to an exorbitant internet service plan or trading invaluable time to get to an internet access point. Furthermore, our product could enable people who are traveling or are far from home to feel comfortable using a Wi-Fi connection for confidential and important actions over the Internet.

The top level objective is that our product shall provide a hub or centerpoint where clients can list their Wi-Fi up for rent, and other clients can visit this hub to rent Wi-Fi that suits their requirements. The means of creating this hub will be via a website. The website shall have geolocation functionality so renters can show where their Wi-Fi is and rentees can find available Wi-Fi based on location. The website should have a secure way for monetary transaction systems. There shall be a notification system for status and reminders for both renters and rentees.

What makes our product different compared to what is currently on the market is that our product conglomerates many options with varying characteristics for accessing internet service into one location so that clients who need quick, temporary Wi-Fi are able to gain access with ease. This in turn opens profit opportunities for clients, who are not an internet service provider company but have internet service, to temporarily rent out Wi-Fi that the client is not using to its maximum capacity.

The target customers are for those seeking to make profits as in clients who want to rent their internet service at a particular location temporarily. Our target customers are also for those seeking Wi-Fi as in c ents who want an alternative form of internet service temporarily to fulfill their current individualized need such as higher security, fast upload speed, uninterrupted internet access, fast download speed, high user capacity amount, etc. There could be many creative individualized needs. Our constraint for who can use our product is that all clients must be 18 years or older.

The scope of our product is to enable greater Wi-Fi quality that an individual is seeking given what he or she is willing to pay; furthermore, we envision that our product will contribute another diverse option in the pool of profit opportunities; clients who want to rent out their Wi-Fi easily can do so which in turn also increases diversity for individuals seeking Wi-Fi. The clients seeking internet access will have greater options. For clarification, the product is not selling Wi-Fi to clients; the product is merely a platform clients can use to do their business.

Our perceived competitors for clients seeking Wi-Fi are the existing ways of gaining internet access--hotspot, 5G, Satellite, Fiber, free public Wi-Fi (Starbucks, libraries, universities,

supermarkets, offices). The perceived competitors for clients seeking to rent out Wi-Fi for extra money are possibly Craigslist listings or sites similar to it and face-to-face contact between renter and rentee.

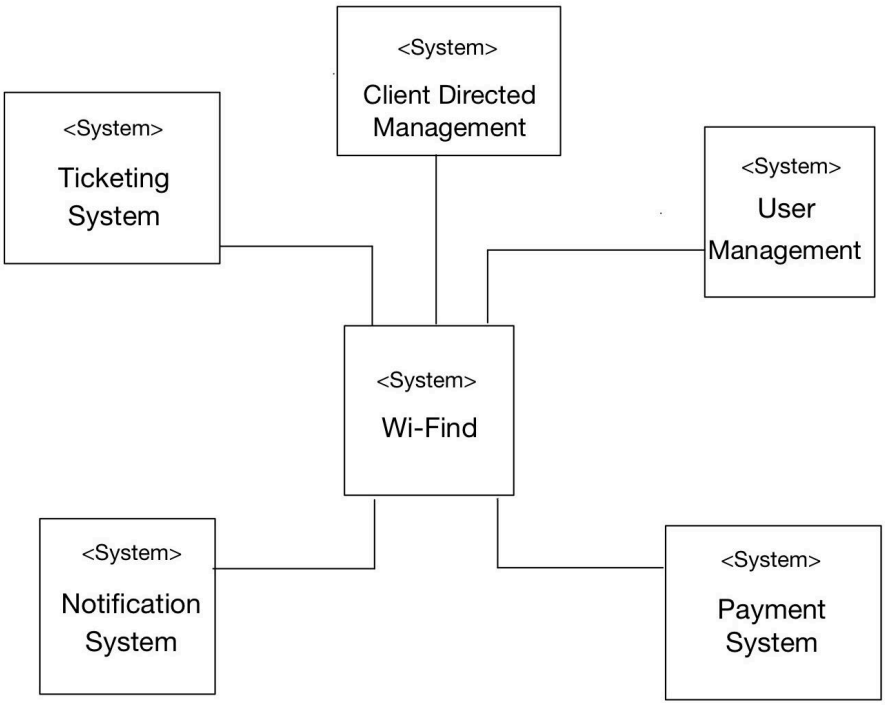
What is novel about our approach is that our hub allows both types of clients to get their needs in ease and break past their current limitations. For clients wishing to rent Wi-Fi, they may be limited physically since not all locations (such as new development areas or remote regions) have structures that enable internet service by a particular provider. For example, fiber, one of the best choices for high speed internet, is not available in every location especially if houses are far spread as laying underground fiber lines is costly. Coverage of Hotspot services such as T-Mobile have dead zones where no service is available or service is so poor that nothing loads. Public Wi-Fi has limitations beyond the monetary cost including cost of time to commute required to the destination, unmeasurable cost from speed and security limitations and cost of planning to accommodate limited time for access like business hours only. Clients may also obtain more buying power and avoid getting stuck in an internet service provider's limited plans. For clients wishing to rent out their Wi-Fi, the hub enables them to make profit by conglomerating the demand pool to just one location therefore increasing their number of possible clientele and formalizing transactions between renter and rentee better.

The system for our product can be built using an existing web stack that best matches the project's requirements and lifecycle. Existing webstacks include technologies and software that are compatible between frontend, backend, database, etc. The Google Maps API has geolocation functionality as well as libraries for calculation of points and area. Using this will allow renters to pin where their Wi-Fi location is and have the range calculated and shown on the map. Rentees can view rentable Wi-Fi's near them or around an address or coordinate they enter. Popular enterprise databases such as postgreSQL have easy-to-use encryption for data management and server storage for protecting sensitive user information such as passwords, purchases and cash exchange information. For payments, a third party established platform such as Stripe can be used as well. Reliable open source application security tools can be used to help make the hub more secure against malicious attacks.

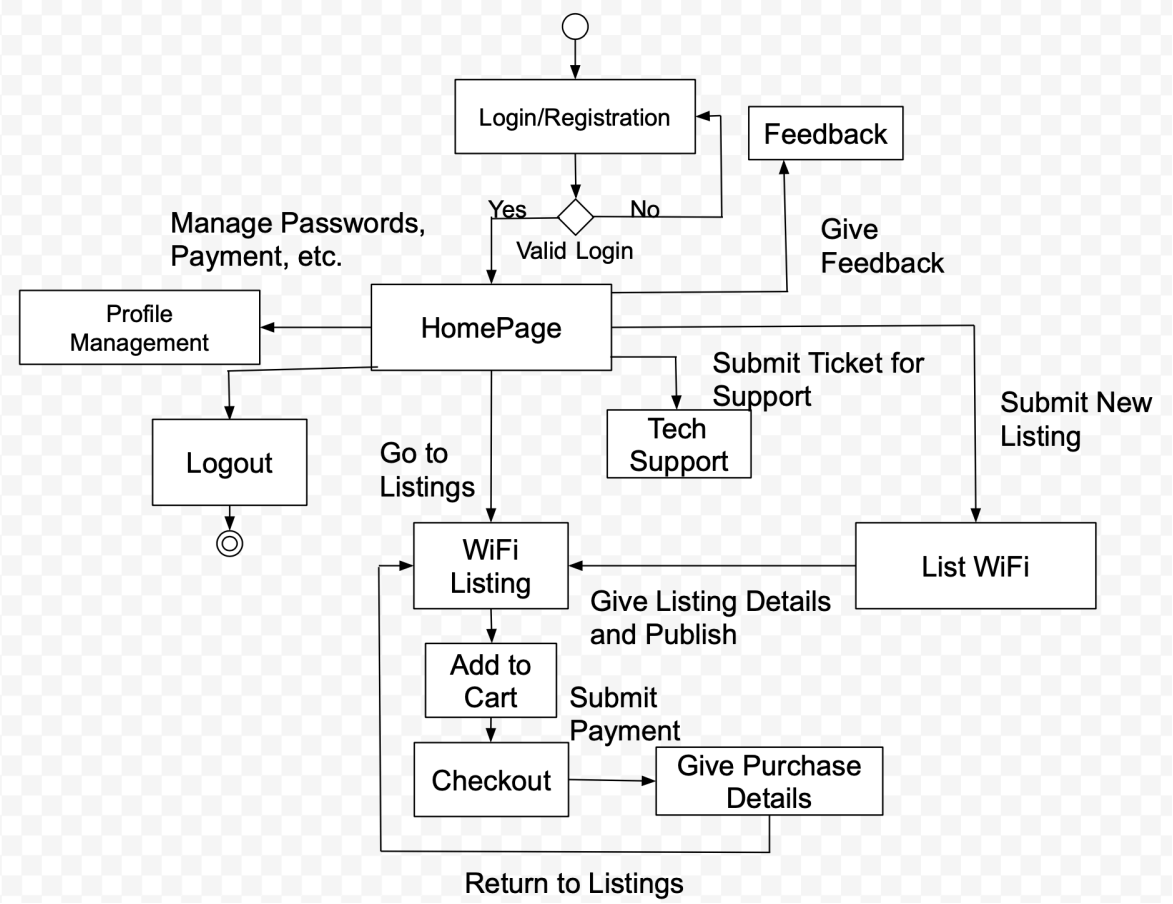
From a technical point of view, implementing appropriate validations for Wi-Fi, transactions, identity, etc may be tricky thus reaching an acceptable solution will be interesting.Another interesting point is figuring out a strategy and implementation to maintain an acceptable update that reflects to all users viewing the site.

There will be a client login and an admin login. The client login will consist of features pertaining to renting and renting out Wi-Fi. Clients should be able to do both if they wanted to (such as rent out their home Wi-Fi while they are on vacation in an area that has poor service leading them to simultaneously rent Wi-Fi from someone else in that area). A separate admin login will be used by the website developers, administrators and moderators. Depending on their role and permission level, actions and information will be limited to just what is necessary for completing the task or maintaining the site.

Section 3: Revised Context Diagram



Section 4: Revised Activity Diagram



Section 5: System Requirements:

- 5.1: Use Cases

  Use Case Number: 1
  - Use Case Name: User Login
  - Actors: User, Administrator, Database
  - Description:
    - On connection to the site, the user will be given a login page.
    - The user will enter an email and password(includes).
    - Recommended option is an excludes option
    - After the inputs are complete, the user will have to hit submit
    - If successful, the user will end up on the destination page
    - If unsuccessful, the user will be given a try again message
    - The information will be validated with the database
    - Administrators can change the look and feel of the website and anything in the database, such as email and password.
  - Alternate Path: There is only 1 path to the destination site, unless given the error to try again.

- Exception Path: If the user doesn't have an account, they will be prompted to sign up.
- Pre-Condition: Valid account must be made.
- Post-Condition: Reaching the destination page through the browser after the pre-condition has been met.

Use Case Number: 2

- Use Case Name: Payment Processing
- Actors: User, Administrators, Database, Third Party Companies
- Description:
    - On clicking checkout, User will be sent to a page asking for payment information
    - The user will fill in their card information, their card number, CVC, address, and name
    - When this information is filled out, the user will have to click a submit button
    - The information will be validated for legitimate cards with third party companies
    - The purchase information will be stored in the database
    - Administrators can change the look and feel of the website, along with any information in the database.
    - The user will be given information about their purchase and then sent to the destination page.
- Alternate Path: The user will be able to select already stored payment information rather than filling out their information manually. The result, the information page and return to
- Exception Path: If the payment information is not validated or incomplete, the user will be prompted to fix the information.
- Pre-Condition: Valid payment information must be entered and submitted
- Post-Condition: Reaching the destination page through the browser after the pre-condition has been met.

Use Case Number: 3

- Use Case Name: Ticket Submission
- Actors: User, Administration, Database
- Description:
    - On clicking Ticket Submission button, the user will be sent to the ticket submission page
    - The user will fill out the submission information
    - When the information is filled out, the user must press the submission button.
    - The ticket will be submitted to the data base.
    - Administrators will be notified of new tickets submitted to the database.
    - An email containing the ticket number and information will be sent to the email address associated with the account.
    - The database will store the ticket information.

    - The user will be sent to the destination page upon submission.
- Alternate Path: There is no alternate path, the only exception will be the user clicking the home button.
- Exception Path: If the ticket sent is blank, the user will be prompted to fill in information rather than submitting a blank ticket.
- Pre-Condition: The user having an account with the website.
- Post-Condition: The user arrives at the destination page after submission.

Use Case Number: 4

- Use Case Name: User Management Portal
- Actors: Administrator, Database
- Description:
    - The administrator will have a user management portal button or tab in the administrator view of the homepage.
    - When the button is clicked, there will be a check with the database for the administrator's permission level.
    - After the permission level is confirmed, the administrator will be directed to the user management portal homepage which only shows action cards or tabs and information that is available to that permission level.
- Alternate Path: There are no other paths to get to the user management portal homepage.
- Exception Path: If validation for checking that the person entering the portal is an administrator fails, the person is redirected to the login screen.
- Pre-Condition:
    - Administrator is logged in
    - Administrator clicked "User Management Portal" button on the homepage
- Post-Condition: The administrator will land on the user management portal tailored to his or her permission level after the pre-condition has been met.

Use Case Number: 5

- Use Case Name: Administrator Ticket Management
- Actors: Administrator, Database, User
- Description:
    - The administrator with a permission level that allows ticket management will have a ticket management button or tab available in his or her user management homepage.
    - When the button or tab is clicked, the administrator will be directed to a page or interface that has a list or table containing all tickets from the database and its attributes.
    - There will be options to sort and filter the table where the administrator can sort by columns and ascending or descending order as well as filter for keywords.
    - A button to assign will be available for tickets the administrator selects as well as a dropdown to assign the tickets to someone.

    - The dropdown contents in the dropdown will be other admins and self. Clicking the confirm button after choosing one option from the dropdown will update the ticket's assignment attribute in the database.
    - Administrators will be able to change their assigned ticket status by selecting the ticket and clicking update ticket status button.
    - Updates will be reflected in the database.
    - Tickets that go from unassigned to assigned status will automatically have an email sent to the user that made the ticket, so the user is informed of ticket acknowledgement.
- Alternate Path: There are no other paths to get to ticket management.
- Exception Path: If error with database occurs, administrator is redirected to user management homepage
- Pre-Condition:
    - Administrator is logged in
    - Administrator's permission level allows ticket management
    - Administrator clicked "Ticket Management" button or tab in the user management portal homepage
- Post-Condition:
    - After preconditions are met, the administrator will land on the ticket management interface.
    - Changes in the ticket management interface will be reflected into the database.
    - Tickets that go from unassigned to assigned status will trigger a notification email to the user that submitted the ticket.

Use Case Number: 6

- Use Case Name: Administrator Remove Inactive User
- Actors: Administrator, Database, User
- Description:
    - The administrator with a permission level that allows inactive user management will have a manage inactive user action card or tab available in his or her user management homepage.
    - When the card or tab is clicked, the administrator will be directed to a page or interface that has a list or table containing all users from the database that meet the definition of inactive.
    - There will be options to sort and filter the table where the administrator can sort by columns and ascending or descending order as well as filter for keywords.
    - The administrator can select users from the list or table and click 'delete'.
    - Clicking the button will trigger a confirmation popup. Confirmation will send a notification email to the user to warn the user of deletion if the user does not log on in a stipulated amount of time.
    - Deletion of the selected user will occur after the user does not log on in the stipulated time and will be reflected in the database.
- Alternate Path: There are no other paths to get to inactive user management.

- Exception Path: If error with database occurs, administrator is redirected to user management homepage
- Pre-Condition:
    - Administrator is logged in
    - Administrator's permission level allows inactive user management
    - Administrator clicked "Check for Inactive Users" button or tab in the user management portal homepage
- Post-Condition:
    - After preconditions are met, the administrator will land on the inactive user management interface.
    - Changes in the interface will be reflected into the database.
    - Inactive users will receive an email about imminent deletion before deletion occurs.

Use Case Number: 7

- Use Case Name: Renting
- Actors: User, Administration, Database
    - Description: When clicking on the "Rent" button, the user will be sent to a rent page of different wifi listings.
    - The User will have the option to browse different wifi levels and prices that best suit their needs, rental dates, and device settings.
    - The user will then select a time period that they would like to rent out their wifi for, similar to a calendar.
    - User will click "Rent now" after choosing a date and have the Wifi listing added to a cart.
    - Users will go through payment processing.
    - Then the user will agree to the rental policy and extra charges if not taken care of.
    - User will confirm the rental.
    - The user will then receive a confirmation email stating it was successful.
- Alternate Path: There is no alternate path, if the user chooses not to go with anything, they will only be able to click the home button.
- Exception Path: If the Wifi listing the user chooses is not available, then the user will be prompted to choose another one.
- Pre-Condition: A valid account must be made beforehand.
- Post-Condition: The user will be sent a confirmation email and sent back to the destination screen.

Use Case Number: 8

- Use Case Name: Wifi Listing
- Actors: User, Database
- Description:
    - User will log into their account and will navigate to the listing section
    - The user will create a new listing, inputting their device information.
    - The user will then add their rental date, availability, terms, and any additional fees.

    - User will agree to listing terms and conditions.
    - User will click “publish now” button and the listing will be added to the website.
- Alternate Path: There is no alternative path, if the user does not list anything when prompted, then they will just click the home button or other tabs.
- Exception Path: If the user chooses to enter nothing or partial information, the user will be prompted to either fill it out completely or discard the listing. .
- Pre-Condition: Users will be logged in with their credentials.
- Post-Condition: User successfully lists their wifi to rent on their website.
- 5.2: Requirement Number

Use Case Number: 9

- Use Case Name: Wifi Managing Listings
- Actors: User, Administration, Database
- Description:
    - User will log into their account and will navigate to listing management section in their dashboard
    - Users will view a list of their existing listing in their management section.
    - Users will be able to select which listing from their list they wish to update or edit.
    - They will click on their listing to access the edit screen to make their changes.
    - If desired, the user will be able to update their rental terms, device information, minimum durations, rental dates, and more.
    - Users will also, if need be, update their listing to unavailable.
    - Users will be able to review and save changes to their listing.
- Alternate Path: There is no alternative path, if the user wishes to not edit or change anything, then they will choose the home button.
- Exception Path: If the user chooses to enter nothing, then the listing will not change. If the user does not save their current changes, they will be prompted and asked to either save their changes or discard them.
- Pre-Condition: Users will be logged in with their credentials and have access to their listing management from their dashboard.
- Post-Condition: The changes the user inputs will be reflected on the Wifi listing on the website.

Use Case Number: 10

- Use Case Name: User Registration
- Actors: User, Administrator, Database
- Description:
    o User accesses the registration page provided by the website.
    o The registration form is displayed, prompting user to enter required information such as a username, email, and password.
    o User fills out the registration form with accurate information.
    o User clicks the "Submit" or "Register" button to proceed.

- The system validates the entered information for correctness and completeness.
- If any errors are detected, the system displays error messages indicating the fields that need to be corrected and prompts the user to resubmit the form.
- If the information provided is valid, the system stores the user's registration details in the database.
- User receives a confirmation message or email indicating successful registration.

- Alternate Path: May include optional steps such as profile customization.
- Exception Path: If the entered information does not meet the required format or criteria the system prompts the user to revise the information.
- Pre-Condition: User must be on the website.
- Post-Condition: User registration is successfully completed, and the user's account details are stored in the system

Use Case Number: 11

- Use Case Name: Payment confirmation
- Actors: User, Administrator, Database
- Description:
  - User initiates a payment transaction through the website.
  - After payment is processed, a confirmation message is generated.
  - The confirmation message includes details such as the transaction ID, payment amount, date and time of payment, and a summary of services.
  - The user receives the confirmation message and reviews the payment details for accuracy.
  - If the user has any questions or concerns regarding the payment, they can contact customer support for assistance.
  - The transaction details are logged in the database for record-keeping purposes.
- Alternate Path: There is only 1 path to the destination site.
- Exception Path:
- Pre-Condition: A payment transaction has been initiated and successfully processed
- Post-Condition: User receives payment confirmation message containing details of the transaction

Use Case Number: 12

- Use Case Name: User Feedback
- Actors: User, Administrator, Database
- Description:
  - The User interacts with the feedback submission interface provided by the website.
  - A Form or dialogue box is presented prompting the user to enter their feedback.
  - User provides feedback by typing text or selecting options from predefined categories.
  - The feedback is submitted,processed, and stored in the database.
  - Feedback may undergo categorization, or prioritization based on importance.

    - A confirmation message is displayed to the user acknowledging successful submission of feedback.
- Alternate Path: May prompt users for feedback at specific touchpoints, such as after completing a transaction.
- Exception Path: If there are technical issues or errors during the submission process, the system may display an error message and prompt the user to retry submitting their feedback.
- Pre-Condition: User is on the website and wishes to provide feedback.
- Post-Condition: User feedback is successfully submitted and stored in the system.

Use Case Number: 13

- Use Case Name: System Maintenance
- Actors: Administrator, System
- Description:
    - Conducting system maintenance to ensure optimal platform performance and reliability.
    - Software updates, updating software components to the latest versions or patches for bug fixes, performance improvements, and security enhancements.
    - Database optimization, improving database performance and efficiency through activities like indexing, defragmentation, and query optimization.
    - Server upgrades, upgrading hardware components or migrating to newer server technologies to enhance performance and scalability.
    - Applying security patches, updating the system with security patches released by software vendors to address known vulnerabilities and enhance security.
- Alternate Path: If the scheduled maintenance window is interrupted due to unforeseen circumstances, administrators reschedule the maintenance activity to minimize service disruptions.
- Exception Path: If critical issues arise during maintenance, such as software update failures or database corruption, administrators halt the process and initiate recovery procedures to restore system functionality.
- Pre-Condition: Administrator authorization and resources for maintenance, users informed in advance of planned downtime.
- Post-Condition: Successful completion of system maintenance, ensuring continued reliability and performance.

Use Case Number: 14

- Use Case Name: User Account Deactivation
- Actors: Administrator, Database
- Description:

    - Deactivating inactive or dormant user accounts to maintain security and manage platform access.
    - Identifying inactive accounts, Identifying user accounts that have been inactive or dormant for a certain period.
    - Changing their status to inactive or suspended, changing the status of inactive accounts to inactive or suspended to prevent unauthorized access.
    - Notifying users, notifying users of the deactivation of their accounts.
    - Providing reactivation instructions if necessary,

- Alternate Path: If an inactive account is mistakenly identified as active, administrators review the account status and verify user activity before deactivation.
- Exception Path: If an account deactivation request fails due to technical issues or database errors, administrators investigate the issue and resolve it promptly to ensure account security and functionality.
- Pre-Condition: Administrator authorization for account deactivation, periodic review of inactive accounts.
- Post-Condition: Successful deactivation of inactive user accounts, reducing security risks and managing platform access effectively.

Use Case Number: 15

- Use Case Name: Web Accessibility
- Actors: User, System
- Description:
    - Enables customers to choose how the site UI should look given a set of premade themes and font size
    - The user can change light and dark theme to suit visibility needs (such as if the user is viewing the site in dark area, the user can switch to a dark theme for less visual stimulation)
    - The user can change font size and font style to accommodate for vision impairments.
    - Changing web accessibility is not affected by login status
- Alternate Path: No alternative path
- Exception Path: If some exception error occurs for any reason, the theme and font is changed back to default.
- Pre-Condition: Customer is on any page in the site.
- Post-Condition: Site UI is updated to user’s accommodation choices.

- 5.2: Requirements

  Requirement Number: 1
  - Use Case Number: 1
  - Introduction: Easy feel and look for a login to go to the destination page, using a username and a login.
  - Inputs: Username and password
  - Requirement Description: The user must be able to sign up, given all requirements are met
    - All fields must be complete
    - Requirements from the system are met
    - Database validation is met
  - Outputs: Routed to destination page

  Requirement Number: 2
  - Use Case Number: 2
  - Introduction: Easy use and secure payment system
  - Inputs: Card Number, CVC, Expiration Date, Address, Name on Card
  - Requirement Description:
    - All fields must complete
    - The credit card information validation must be met
    - Database storage is completed
  - Outputs: Confirmation of payment and payment details

  Requirement Number: 3
  - Use Case Number: 3
  - Introduction: Quick, Easy, and versatile Ticket System
  - Inputs: Ticket Information
  - Requirement Description:
    - Information filled in
    - Submission button clicked
    - Ticket information storage in database
    - Ticket information emailed to customer
  - Outputs: Ticket Number emailed to customer and details of ticket

  Requirement Number: 4
  - Use Case Number: 4
  - Introduction: Easy organized look and feel page for administrators to manage users, view tasks and manage tickets. Ability to view and manage depends on the administrator's permission level.
  - Inputs: Administrator credentials
  - Requirement Description:
    - Given administrator is logged in,
    - Administrator’s permission level is checked then user management welcome page shows only actions that the permission level allows.
    - The administrator can click options available to see more.

- Outputs:
    - If action is clicked, administrator is navigated to the respective view.
    - Else, log administrator out if no activity is detected

Requirement Number: 5

- Use Case Number: 5
- Introduction: List of tickets with current status and assigned administrator is shown in a table that can be filtered and sorted. Administrator can assign ticket to self or assign to other administrators.
- Inputs: Administrator credentials, optional keywords for filtering table
- Requirement Description:
    - Given administrator is logged in and has appropriate permission level,
    - List or table is shown containing all tickets and their attributes from the database.
    - Administrator can filter and sort list or table and can type in keywords to search within table
    - Admin can assign a ticket to another administrator or reassign a ticket to self.
- Outputs:
    - If changes are made to tickets, respective ticket information in database is updated
    - If unassigned tickets become assigned, email notification to user is sent.

Requirement Number: 6

- Use Case Number: 6
- Introduction: Interface for administrator to handle deleting inactive users. Administrators can view users' last login in a table and, email on a table or list that the administrator can filter and sort.
- Inputs: Administrator credentials, optional keywords for filtering table
- Requirement Description:
    - Given administrator is logged in and has appropriate permission level,
    - List or table of users that meet a given inactivity time (such as last login date was over 6 months) is shown.
    - List or table can be filtered and sorted, and administrators can type in keywords to search within table
    - Administrator can click a button to email inactive users that the user's account will be deleted if last login date is not updated by a certain time.
- Outputs:
    - If action buttons were clicked, users are sent emails.
    - Database entries are updated if deletion criterias were met.

Requirement Number: 7

- Use Case Number: 7
- Introduction: This requirement outlines how users will rent Wifi from the website.
- Inputs:
    - User credentials

    - Selection of Wifi to rent
    - Specification of rental term
    - Rental agreements
    - Payment information
- Requirement Description: This system allows users to browse and select their wifi they wish to rent. Users will have to provide the rental date they wish to go with and payment information.
- Outputs:
    - Confirmation details in the form of email
    - Rental details as well as device information
    - Access to rented wifi to use during rental period

Requirement Number: 8

- Use Case Number: 8
- Introduction: This requirement outlines the process users will go through to list their wifi to rent on the website.
- Inputs:
    - User login
    - Wifi device details
    - Rental terms
    - Agreement to list terms/conditions.
- Requirement Description: The system should provide a listing interface to input their wifi details to list onto the website as well as their rental terms and agree to listing agreements.
- Outputs:
    - A confirmation message informing the user of a successful listing
    - Published listing for users browsing for available internet.

Requirement Number: 9

- Use Case Number: 9
- Introduction: This requirement showcases the functionality of managing their listings on the website. Users should be able to edit listing terms, change rental dates or take down their listings whenever they desire.
- Inputs:
    - User credentials
    - Access to listing management section in users dashboard
    - Selection of specific listing or edit
- Requirement Description: This system should be able to allow users to edit their listings whenever, update rental dates, device info, and more. On top of this as well, it also should allow the user to remove their listing from the website due to either damage, currently in use, or other reasons.
- Outputs:
    - A confirmation stating the changes and edits have been made and saved.
    - The wifi listing is updated to whatever the user has chosen to change.

Requirement Number: 10

- Use Case Number: 10

- Introduction: User-friendly registration process
- Inputs: Personal information (name, dob, etc), email address, password
- Requirement Description:
    - Mandatory fields in the registration form must be filled out by the user
    - Validated data securely stored within the system's database
- Outputs: Account details stored in the system's database, allowing them to log in.

Requirement Number: 11

- Use Case Number: 11
- Introduction: Payment confirmation involves verifying user-provided payment details and obtaining user confirmation to finalize a transaction.
- Inputs: clicking a "Confirm Payment" button or similar action
- Requirement Description:
    - Payment is confirmed, action serves as the input to initiate the finalization of the payment process.
- Outputs: Message or email containing details of the transaction.

Requirement Number: 12

- Use Case Number: 12
- Introduction: This process collects user input
- Inputs: Text input, ratings, or selection from predefined categories
- Requirement Description:
    - Validate submitted feedback to ensure it meets basic requirements
    - Store feedback in the system's database
- Outputs: Confirms successful submission of feedback to the user, through an on-screen notification or a confirmation message

Requirement Number: 13

- Use Case Number: 13
- Introduction: System Maintenance
- Inputs: System maintenance triggers, administrator credentials
- Requirement Description:
    - The system must undergo routine maintenance to ensure optimal performance and security.
    - Software updates, database optimization, server upgrades, and security patching.
    - Administrators are responsible for initiating and overseeing maintenance activities.
- Outputs: Confirmation of maintenance completion, system stability, and security enhancements.

Requirement Number: 14

- Use Case Number: 14
- Introduction: User Account Deactivation
- Inputs: Administrator credentials, inactive user accounts
- Requirement Description:

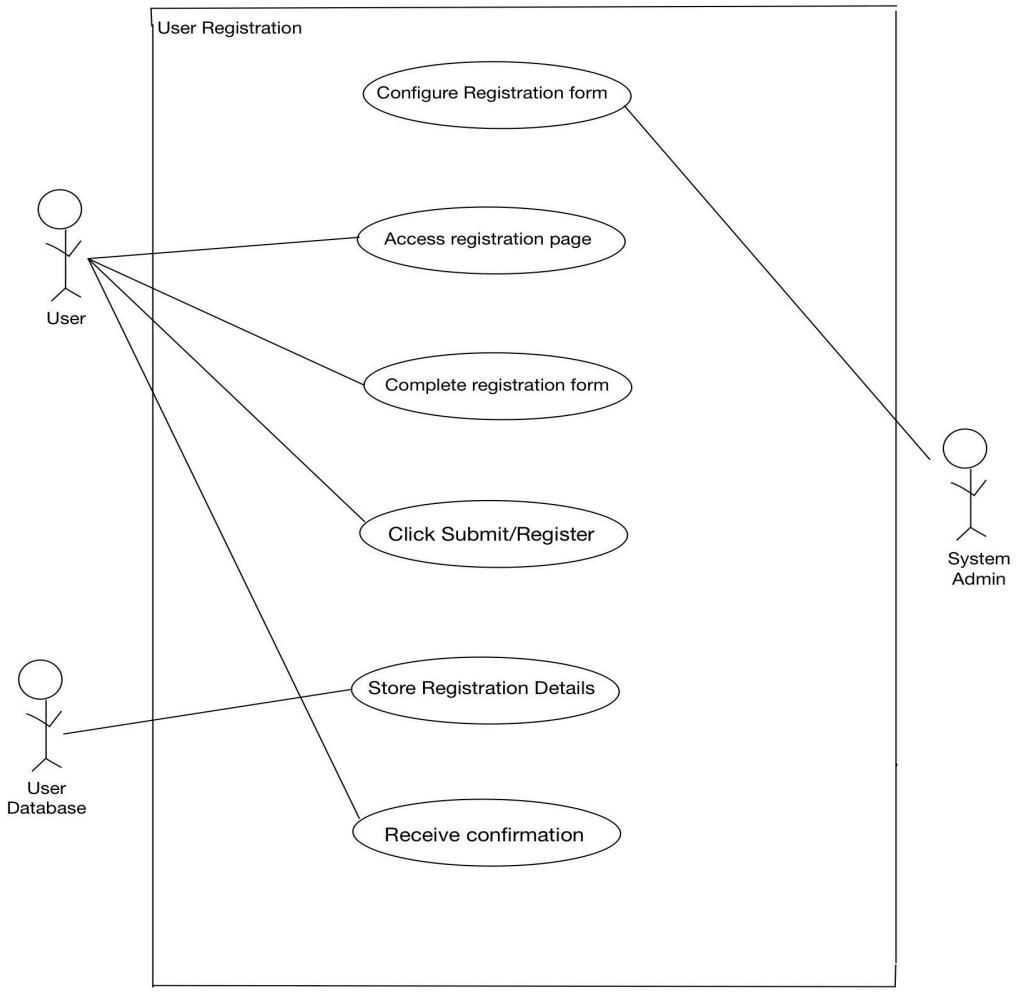
    - Inactive user accounts must be deactivated to enhance system security and manage access.
    - Administrators review and identify inactive accounts, change their status to inactive or suspended, and notify users accordingly.
    - Deactivated accounts may be reactivated upon user request or system validation.
- Outputs: Notification to users regarding account deactivation, enhanced system security.

Requirement Number: 15

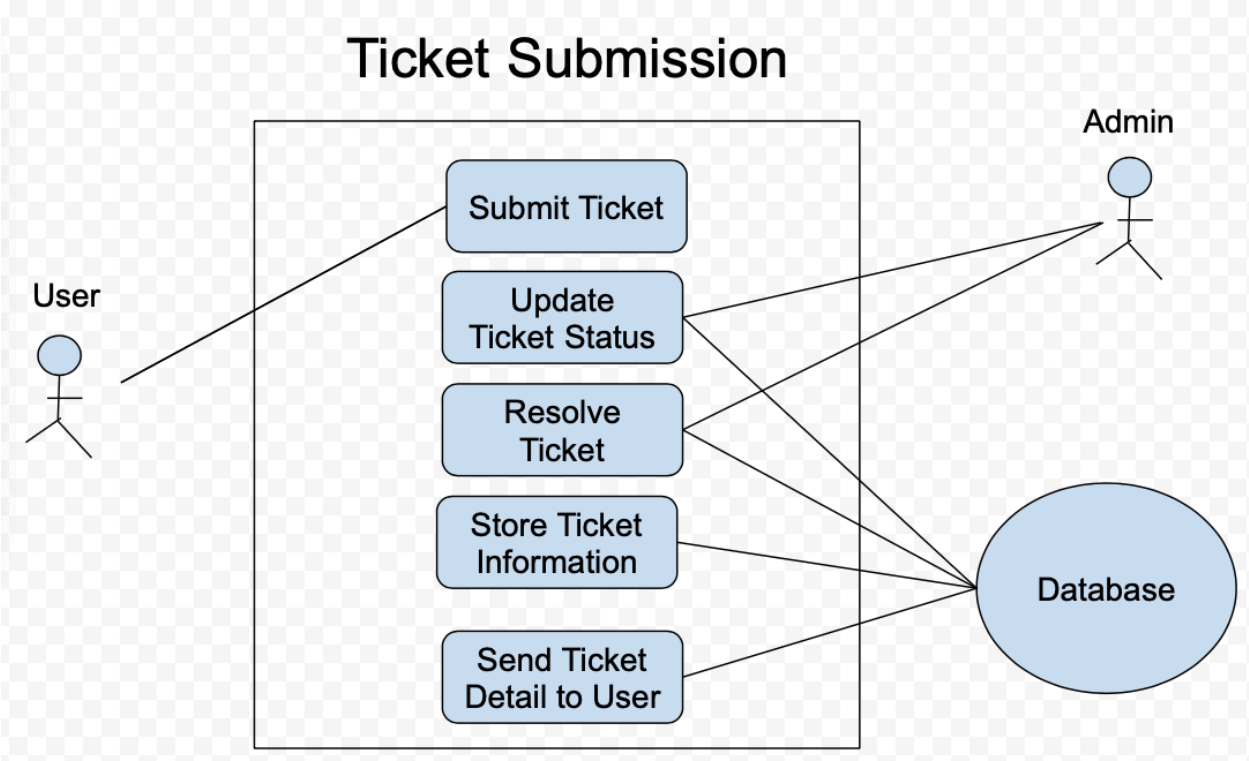
- Use Case Number: 15
- Introduction: Web Accessibility
- Inputs: User's font size choice
- Requirement Description:
    - The user should be able to change the font size of the entire site via a web accessibility button
    - The accessibility button is always hovering at the bottom corner of every page in the site
    - Clicking the accessibility button should open a pop up where the user can adjust font size.
    - Every page on the site should reflect the user's chosen font size.
    - User should not need to login to access the web accessibility button and its features.
- Outputs: Every text for every page on the site has been adjusted to the user's chosen font size.

- 5.3: Use Case Diagrams

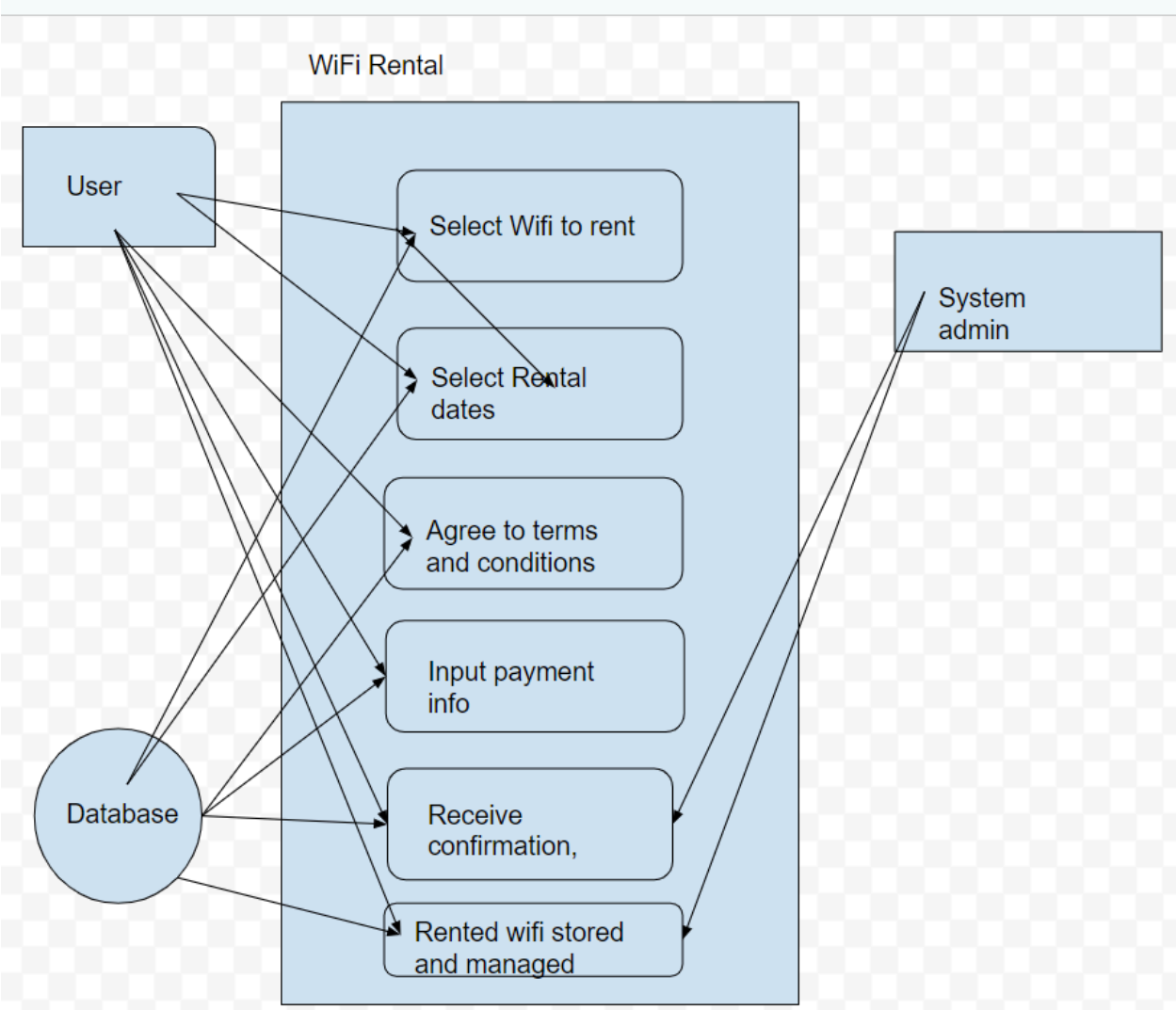
User Registration



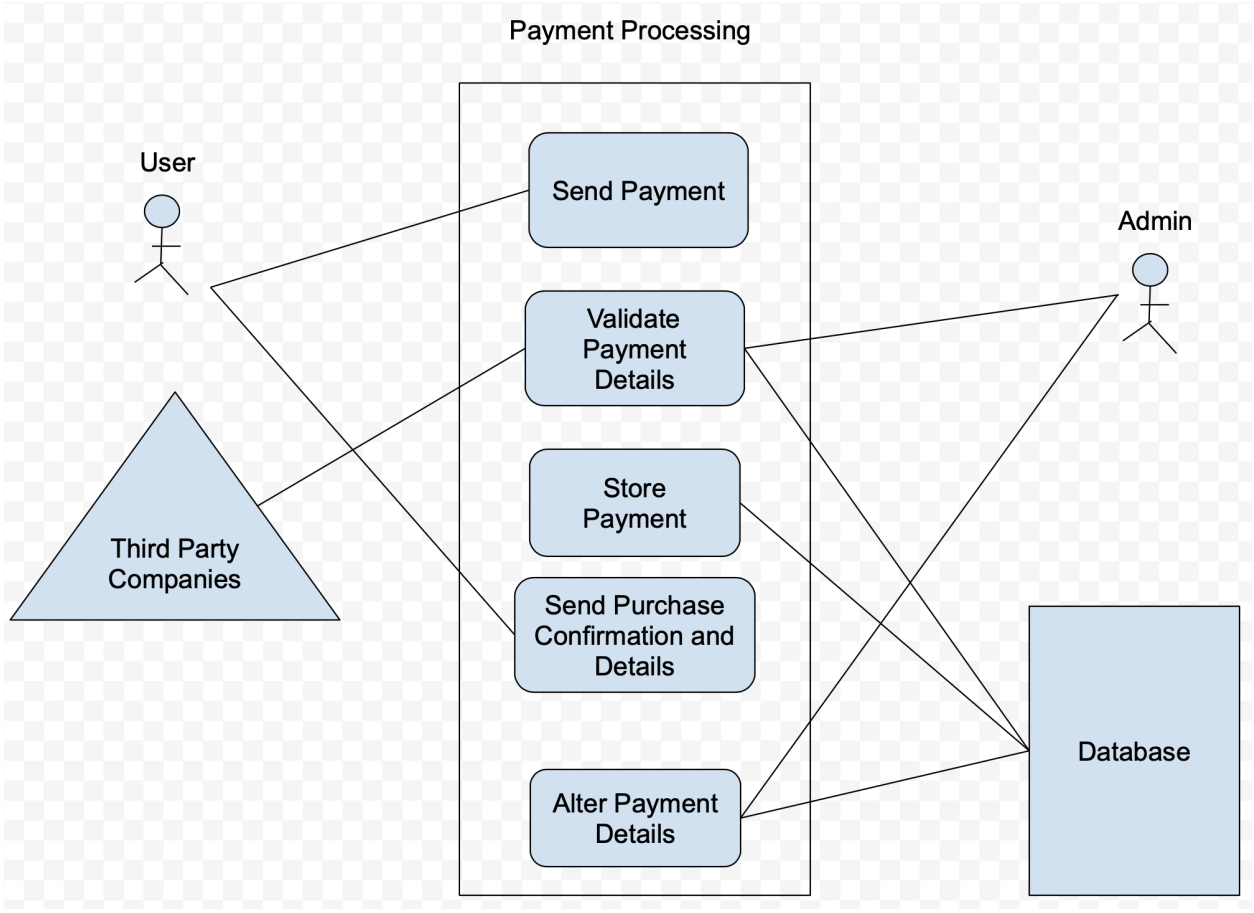
Ticket Submission



Wi-Fi Rental

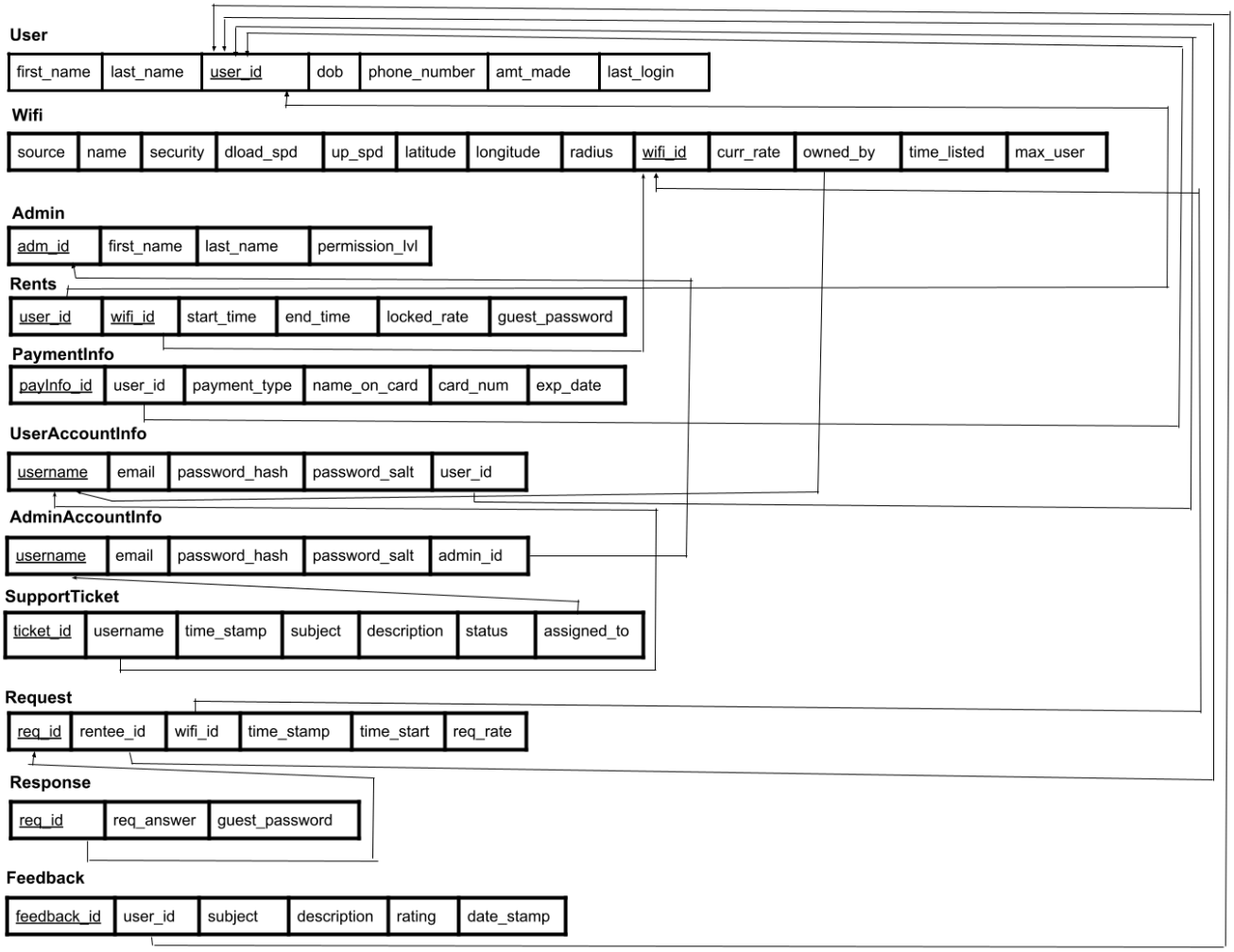

WiFi Rental
User
Select Wifi to rent
Select Rental dates
Agree to terms and conditions
Input payment info
Receive confirmation,
Rented wifi stored and managed
Database
System admin

Payment Processing



Section 6: Database Management

System Database Tables:



Database Management System:

MS SQLServer

Schema Statements:

User(user_id, first_name, last_name, dob, phone_num, amt_made, last_login)
User(user_id: string, first_name: string, last_name: string, dob: date, phone_num: string, amt_made: float, last_login: datetime)

Wifi(wifi_id, name, security, dload_spd, up_spd, source, latitude, longitude, radius, curr_rate, time_listed, max_user, owned_by)
Wifi(wifi_id: string, name: string, security: string, dload_spd: int, up_spd: int, source: string, latitude: float, longitude: float, radius: float, curr_rate: decimal, time_listed: datetime, max_user: int, owned_by: string)

Admin(admin_id, first_name, last_name, permission_level)
Admin(admin_id: string, first_name: string, last_name: string, permission_level: string)

Rents(user_id, wifi_id, start_time, end_time, locked_rate, guest_password)
Rents(user_id: string, wifi_id: string, start_time: datetime, end_time: date time, locked_rate: decimal, guest_password: string)

PaymentInfo(payInfo_id, user_id, type, name_on_card, card_num, exp_date)
PaymentInfo(payInfo_id: string, user_id: string, type: string, name_on_card: string, card_num: string, exp_date: date)

UserAccountInfo(username, email, password_hash, password_salt, user_id)
UserAccountInfo(username: string, email: string, password_hash: varbinary, password_salt: varbinary, user_id: string)

AdminAccountInfo(username, email, password_hash, password_salt, admin_id)
UserAccountInfo(username: string, email: string, password_hash: varbinary, password_salt: varbinary, admin_id: string)

SupportTicket(ticket_id, username, time_stamp, title, description, status, assigned_to)
SupportTicket(ticket_id: string, username: string, time_stamp: timestamp, title: string, description: string, status: string, assigned_to: string)

Request(req_id, rentee_id, wifi_id, time_stamp, time_start, req_rate)
Requst(req_id: string, rentee_id: string, wifi_id: string, time_stamp: timestamp, time_start: datetime, req_rate: decimal)

Response(request_id, permission_status, guest_password)
Response(request_id: string, permission_status: boolean, guest_password: string)

Feedback(feedback_id, user_id, rating, title, description)
Feedback(feedback_id: string, user_id: string, title: string, description: string)

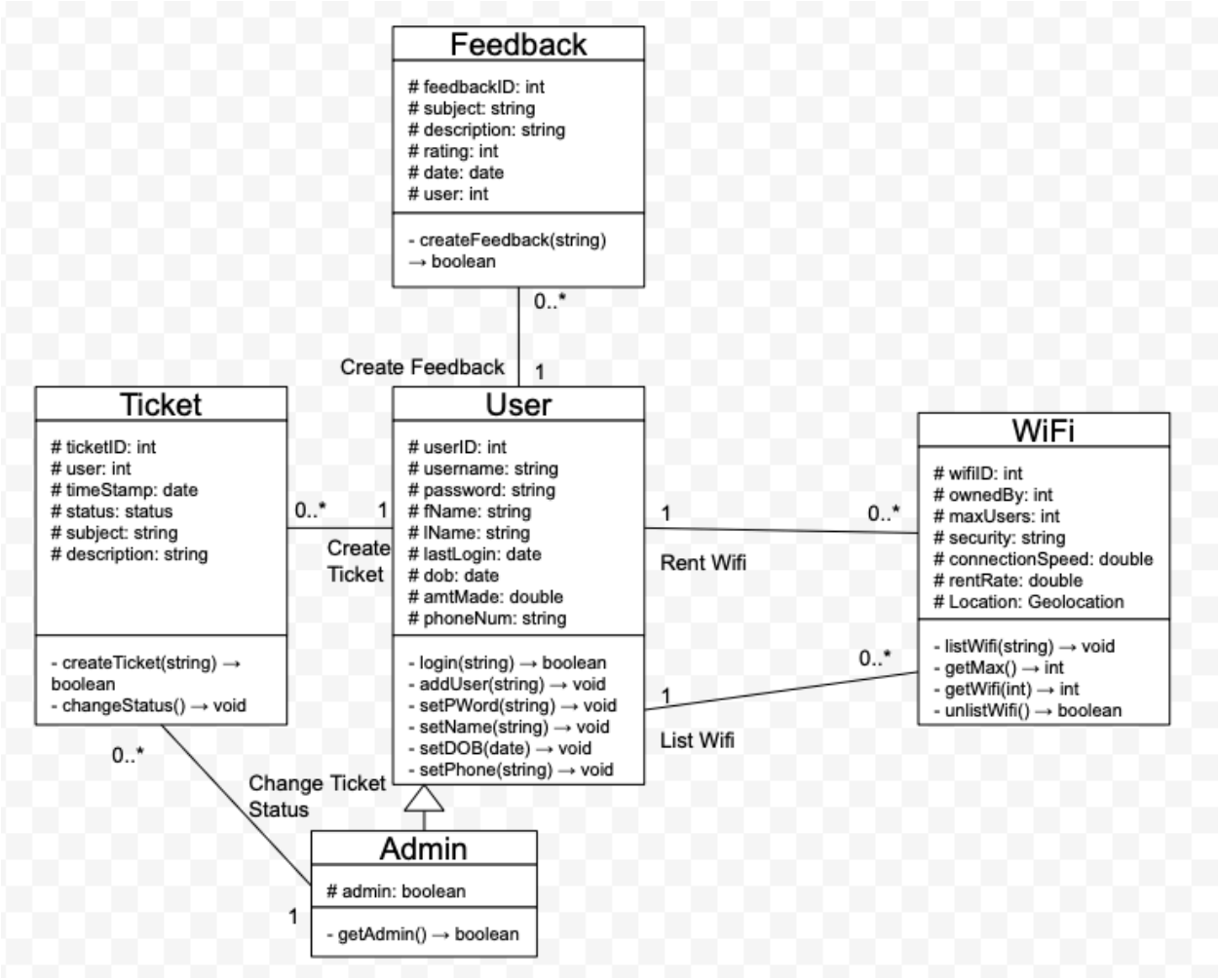
Relationships:

<u>Parent Tables</u>
User: user_id(PK), first_name, last_name, dob, phone_num, amt_made, last_login
Admin: admin_id(PK), first_name, last_name, permission_level

<u>Child Tables</u>
Wifi: wifi_id(PK), name, security, dload_spd, up_spd, source, latitude, longitude, radius, curr_rate, time_listed, max_user, owned_by(FK)
Rents(user_id(FK), wifi_id(FK), start_time, end_time, locked_rate, guest_password)
PaymentInfo(payInfo_id(PK), user_id(FK), type, name_on_card, card_num, exp_date)
UserAccountInfo(username(PK), email, password_hash, password_salt, user_id)
AdminAccountInfo(username(PK), email, password_hash, password_salt, admin_id)

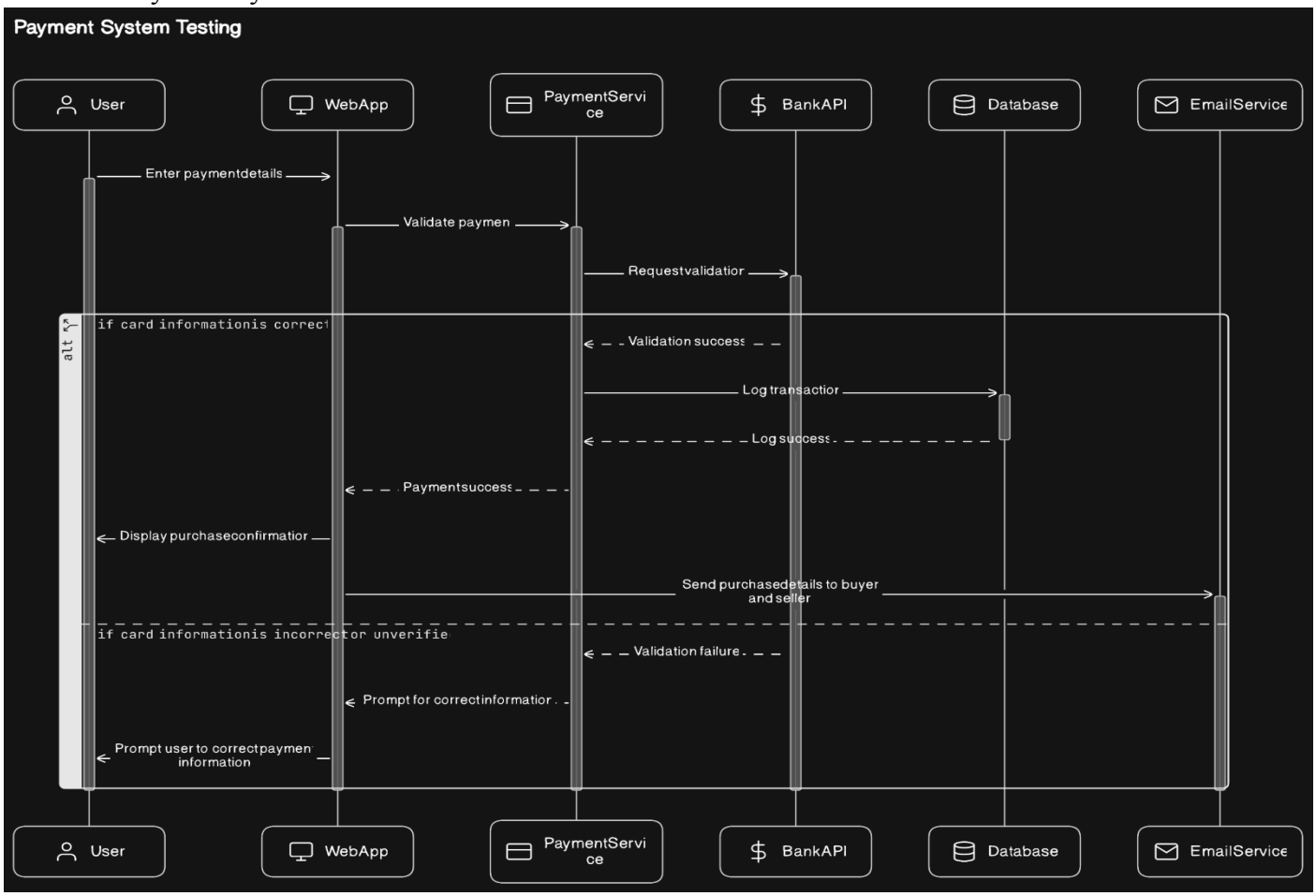
SupportTicket(ticket_id(PK), username(FK), time_stamp, title, description, status, assigned_to(FK)
Request(req_id(PK), rentee_id(FK), wifi_id(FK), time_stamp, time_start, req_rate)
Response(request_id(FK), permission_status, guest_password)
Feedback(feedback_id(PK), user_id(FK), rating, title, description)

Section 7: Class Diagram

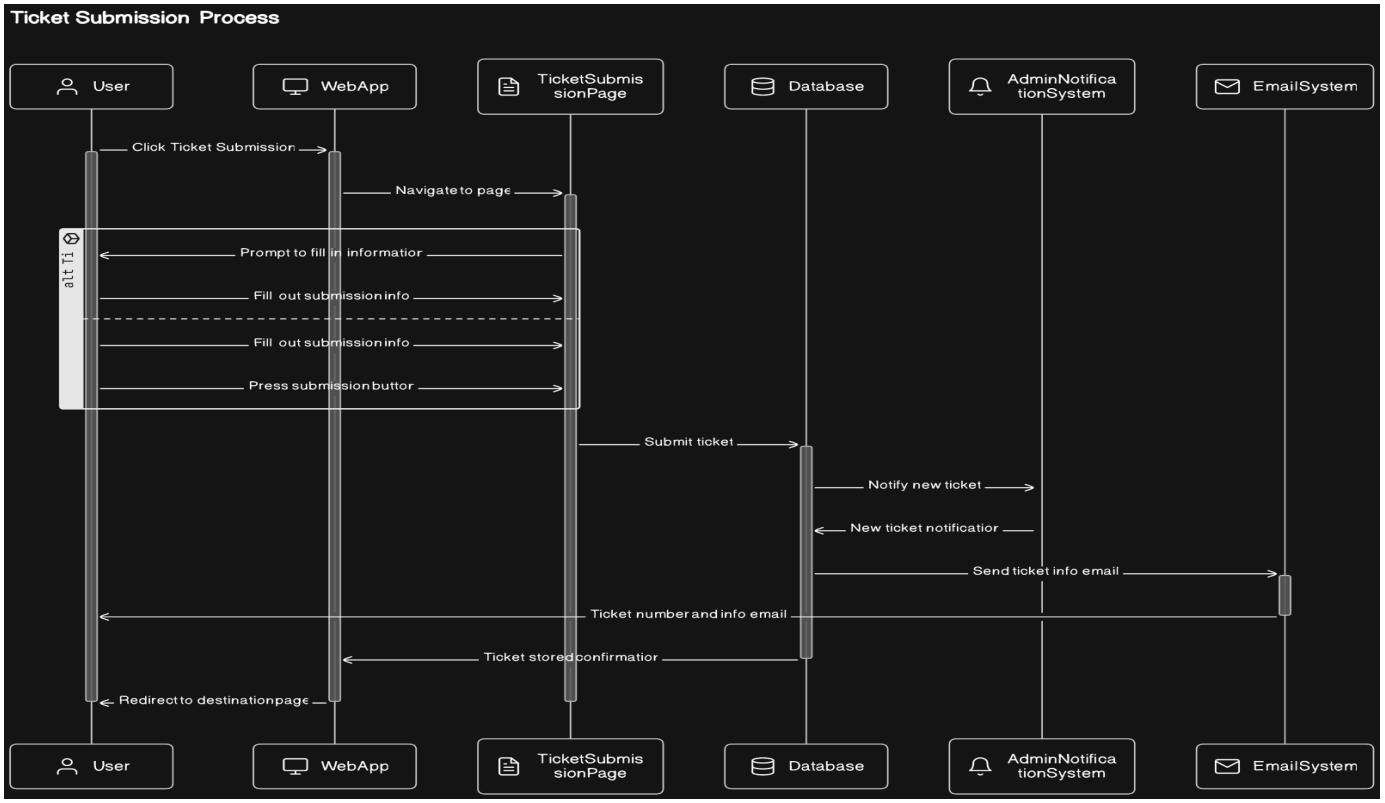


Section 8: Behavioral Modeling

Case #2 - Payment System:



Case #3 - Ticket Submission :



Section 10: Test Cases

Test Case ID: 1.1 – Login Testing, Correct Information

- Description: User enters correct login credentials that exist in the database into the log in portal
- Test Inputs: {Username (String), Password (String)}
- Expected Results: Hashed password fits the username's hashed password in the database. User is logged in and redirected to WiFind Homepage.
    - Given hashed password doesn't match hashed password in database → Error – incorrect password.
    - Username is not in the database → Error – User not registered.
    - Either field is blank → Error – Incomplete information.
- Dependencies: Database, Server
- Test Steps:
    - Give correct username and password
    - Verify user is logged in to correct account and redirected to WiFind Homepage.

- Post Conditions: User information is correctly verified and the user is logged into the correct account and redirected to the WiFind homepage.

Test Case ID: 1.2 – Login Testing, Incorrect Username

- Description: User enters incorrect username and a password that exist in the database into the log in portal
- Test Inputs: {Username (String), Password (String)}
- Expected Results: hashed password of given username does not match hashed password of username in the database. An error is thrown telling the user that the either the username or password is incorrect.
- Dependencies: Database, Server
- Test Steps:
    - Give incorrect username and a password
    - Check that information is correctly not verified by the database
    - Verify that error is shown to user telling them that either their username or password is incorrect
- Post Conditions: User information is correctly rejected and the user is shown a message that either the given username or password is wrong.

Test Case ID: 1.3 – Login Testing, Wrong Password

- Description: User enters correct username that exist in the database and the wrong password into the log in portal
- Test Inputs: {Username (String), Password (String)}
- Expected Results: Hashed password is not the same as the hashed password in the database. The user is shown a message that either the username or password is incorrect.
- Dependencies: Database, Server
- Test Steps:
    - Give correct username and incorrect password
    - Verify that the password is rejected by the database
    - Verify that the user is shown a message that either the username or password is incorrect
- Post Conditions: The password is rejected and the user is not logged in. The user is shown a message that the username or password is incorrect.

Test Case ID: 1.4 – Login Testing, Nonexistent username

- Description: User enters username that does not exist in the database and a password
- Test Inputs: {Username (String), Password (String)}
- Expected Results: Username is not found in the database and the user is given a message that the username
- Dependencies: Database, Server
- Test Steps:
    - Give correct username and incorrect password
    - Verify that the password is rejected by the database
    - Verify that the user is shown a message that the username does not exist in the database

- Post Conditions: The credentials are rejected and the user is directed to create an account.

Test Case ID: 1.5 – Login Testing, Not filled in username

- Description: User enters only a password
- Test Inputs: {Password (String)}
- Expected Results: Data is not submitted since username is null. User is given error message that information is missing.
- Dependencies: Database, Server
- Test Steps:
    - Fill in password
    - Submit information
    - Verify that data is not given to database
    - Verify that user is given error message saying that the information is incomplete.
- Post Conditions: The credentials are rejected and the user is given an error message that the information is incomplete.

Test Case ID: 1.6 – Login Testing, Not filled in password

- Description: User enters only a username
- Test Inputs: {Password (String)}
- Expected Results: Data is not submitted since password is null. User is given error message that information is missing.
- Dependencies: Database, Server
- Test Steps:
    - Fill in password
    - Submit information
    - Verify that data is not given to database
    - Verify that user is given error message saying that the information is incomplete.
- Post Conditions: The credentials are rejected and the user is given an error message that the information is incomplete.

Test Case ID: 2.1 – Payment System Testing, Correct Information

- Description: User inputs correct card information and the payment is processed.
- Test Inputs: {Name (String), Card_Number (String), CVC (Int), Expiration_Date (Date), Billing_Address (String), Billing_ZipCode (Int)}
- Expected Results: The card information is verified with a third party issuer, the payment is processed, and the user is given details about their purchase.
- Dependencies: Login Module, Listing Module, Database, Server
- Test Steps:
    - Input correct information
    - Verify that purchase is made
    - Verify that correct details are given for the purchase.
    - Verify that the purchase is added to the database.
- Post Conditions: The user is shown a confirmation message, and given details about their purchase.

Test Case ID: 2.2 – Payment System Testing, Incorrect Name

- Description: User enters incorrect name and the payment is rejected.
- Test Inputs: {Name (String), Card_Number (String), CVC (Int), Expiration_Date (Date), Billing_Address (String), Billing_ZipCode (Int)}
- Expected Results: The card information is not verified with a third party issuer, the payment is not processed, and the user is given a message saying that the card information is incorrect.
- Dependencies: Login Module, Listing Module, Database, Server
- Test Steps:
    - Input incorrect name and all other information is correct
    - Send card information to issuer
    - Verify that the card is rejected
    - Verify that the card is rejected by our payment module
    - Verify that our system rejects the card
    - Verify that the user is shown error message
- Post Conditions: The user is shown an error message telling them that the information given is incorrect.

Test Case ID: 2.3 – Payment System Testing, Incorrect Card Number

- Description: User enters incorrect card number and the payment is rejected.
- Test Inputs: {Name (String), Card_Number (String), CVC (Int), Expiration_Date (Date), Billing_Address (String), Billing_ZipCode (Int)}
- Expected Results: The card information is not verified with a third party issuer, the payment is not processed, and the user is given a message saying that the card information is incorrect.
- Dependencies: Login Module, Listing Module, Database, Server
- Test Steps:
    - Input incorrect name and all other information is correct
    - Send card information to issuer
    - Verify that the card is rejected
    - Verify that the card is rejected by our payment module
    - Verify that our system rejects the card
    - Verify that the user is shown error message
- Post Conditions: The user is shown an error message telling them that the information given is incorrect.

Test Case ID: 2.4 – Payment System Testing, Incorrect CVC

- Description: User enters incorrect CVC and the payment is rejected.
- Test Inputs: {Name (String), Card_Number (String), CVC (Int), Expiration_Date (Date), Billing_Address (String), Billing_ZipCode (Int)}

- Expected Results: The card information is not verified with a third party issuer, the payment is not processed, and the user is given a message saying that the card information is incorrect.
- Dependencies: Login Module, Listing Module, Database, Server
- Test Steps:
    - Input incorrect name and all other information is correct
    - Send card information to issuer
    - Verify that the card is rejected
    - Verify that the card is rejected by our payment module
    - Verify that our system rejects the card
    - Verify that the user is shown error message
- Post Conditions: The user is shown an error message telling them that the information given is incorrect.

Test Case ID: 2.5 – Payment System Testing, Incorrect Expiration Date

- Description: User enters incorrect expiration date and the payment is rejected.
- Test Inputs: {Name (String), Card_Number (String), CVC (Int), Expiration_Date (Date), Billing_Address (String), Billing_ZipCode (Int)}
- Expected Results: The card information is not verified with a third party issuer, the payment is not processed, and the user is given a message saying that the card information is incorrect.
- Dependencies: Login Module, Listing Module, Database, Server
- Test Steps:
    - Input incorrect name and all other information is correct
    - Send card information to issuer
    - Verify that the card is rejected
    - Verify that the card is rejected by our payment module
    - Verify that our system rejects the card
    - Verify that the user is shown error message
- Post Conditions: The user is shown an error message telling them that the information given is incorrect.

Test Case ID: 2.6 – Payment System Testing, Incorrect Billing Address

- Description: User enters incorrect billing address and the payment is rejected.
- Test Inputs: {Name (String), Card_Number (String), CVC (Int), Expiration_Date (Date), Billing_Address (String), Billing_ZipCode (Int)}
- Expected Results: The card information is not verified with a third party issuer, the payment is not processed, and the user is given a message saying that the card information is incorrect.
- Dependencies: Login Module, Listing Module, Database, Server
- Test Steps:
    - Input incorrect billing address and all other information is correct
    - Send card information to issuer
    - Verify that the card is rejected
    - Verify that the card is rejected by our payment module
    - Verify that our system rejects the card

    - Verify that the user is shown error message
- Post Conditions: The user is shown an error message telling them that the information given is incorrect.

Test Case ID: 2.7 – Payment System Testing, Incorrect Name

- Description: User enters incorrect billing zipcode and the payment is rejected.
- Test Inputs: {Name (String), Card_Number (String), CVC (Int), Expiration_Date (Date), Billing_Address (String), Billing_ZipCode (Int)}
- Expected Results: The card information is not verified with a third party issuer, the payment is not processed, and the user is given a message saying that the card information is incorrect.
- Dependencies: Login Module, Listing Module, Database, Server
- Test Steps:
    - Input incorrect billing zipcode and all other information is correct
    - Send card information to issuer
    - Verify that the card is rejected
    - Verify that the card is rejected by our payment module
    - Verify that our system rejects the card
    - Verify that the user is shown error message
- Post Conditions: The user is shown an error message telling them that the information given is incorrect.

Test Case ID: 2.8 – Payment System Testing, Incomplete Information

- Description: User enters incomplete information and the attempted payment is not submitted.
- Test Inputs: {Name (String), Card_Number (String), CVC (Int), Expiration_Date (Date), Billing_Address (String), Billing_ZipCode (Int)}
- Expected Results: The card information is not submitted for verification and the user is shown a message that the card information is incomplete.
- Dependencies: Login Module, Listing Module, Database, Server
- Test Steps:
    - Input incorrect name and all other information is correct
    - Send card information to issuer
    - Verify that the card is rejected
    - Verify that the card is rejected by our payment module
    - Verify that our system rejects the card
    - Verify that the user is shown error message
- Post Conditions: The user is shown an error message telling them that the information given is incorrect.

Test Case ID: 3.1 – Ticket Submission

- Description: User submits ticket for technical support with complete and correct information
- Test Inputs: {Title (string), Description (string)}

- Expected Results: User submits ticket with title and information and is given a confirmation and email detailing the status of their ticket.
- Dependencies: Login Module, Database, Email System
- Test Steps:
    - Submit correct information
    - Verify that the information is correctly stored in the database.
    - Verify that the user is shown a confirmation message.
    - Verify that the user is sent an email with the details and status of their ticket.
- Post Conditions: User shown confirmation message and redirected to the WiFind homepage.

Test Case ID: 3.2 – Ticket Submission, No Title
- Description: User submits ticket for technical support without title
- Test Inputs: {Description (string)}
- Expected Results: User shown message that ticket is incomplete
- Dependencies: Login Module, Database, Email System
- Test Steps:
    - Submit ticket without title
    - Verify that the ticket is rejected
    - Verify that the user is shown a message that the information is incomplete
- Post Conditions: User shown incomplete information message.

Test Case ID: 3.2 – Ticket Submission, No Description
- Description: User submits ticket for technical support without description
- Test Inputs: {Title (string)}
- Expected Results: User shown message that ticket is incomplete
- Dependencies: Login Module, Database, Email System
- Test Steps:
    - Submit ticket without description
    - Verify that the ticket is rejected
    - Verify that the user is shown a message that the information is incomplete
- Post Conditions: User shown incomplete information message.

Test Case ID: 4.1 – User Management Portal
- Description: Admin with an active admin token attempts to go into user management portal.
- Test Inputs: Unexpired admin bearer token
- Expected Results: Admin is redirected to user management portal page containing admin only actions.
- Dependencies: Admin login portal, database, server
- Initialization: Admin successfully logged in within the last few minutes and is on homepage
- Test Steps:
    - Click user management portal button
    - Verify redirection to user management portal by observing the link has “usermanagementportal” and the page does not say “Unauthorized”

- Post Conditions:
    - Admin is in the user management portal page.

Test Case ID: 4.2 – User Management Portal, Access

- Description: Admin with an expired admin token attempts to go into user management portal.
- Test Inputs: Expired admin bearer token
- Expected Results: Admin is redirected to admin login page
- Dependencies: Admin login portal, database, server
- Initialization: Admin successfully logged in and is on homepage
- Test Steps:
    - Idle past token's expiration time (3 hours)
    - Click user management portal button
    - Verify redirection to admin login page by observing the link has "adminlogin" and page is admin account login.
- Post Conditions:
    - Admin is in the admin login page.

Test Case ID: 5.1 – User Management Portal, Support Ticket Management

- Description: Admin with ticket permission enters user management portal
- Test Inputs: Active admin bearer token associated with an admin id is allowed tickets permission in it
- Expected Results: Ticket Management action card/button is on user management portal homepage
- Dependencies: Admin login portal, database, server
- Initialization: Admin successfully logged in and was successfully redirected to user management portal page
- Test Steps:
    - Wait for home page to load
    - Verify "Ticket Management" Card or Tab is on the page.
- Post Conditions: Admin can see ticket management on the user management portal page

Test Case ID: 5.2 – User Management Portal, Support Ticket Management

- Description: Admin with no ticket permission enters user management portal
- Test Inputs: Active admin bearer token associated with an admin id is does not have tickets permission in it
- Expected Results: Ticket Management action card/button is not on user management portal homepage
- Dependencies: Admin login portal, database, server
- Initialization: Admin successfully logged in and was successfully redirected to user management portal page
- Test Steps:
    - Wait for home page to load
    - Verify "Ticket Management" Card or Tab is not on the page.
- Post Conditions: Admin cannot view or interact the ticket management card/tab on the user management portal page

Test Case ID: 5.3 – User Management Portal, Support Ticket Management

- Description: Admin with ticket permission selects an unassigned ticket and attempts to assigns to self
- Test Inputs: Unassigned support ticket and admin token
- Expected Results: Support ticket is updated to have admin's username on it
- Dependencies: Admin login portal, database, server
- Initialization: Active admin bearer token associated with an admin id that has tickets permission in it, clicked on ticket management action card/tab, and can observe a list of support tickets on the page
- Test Steps:
    - Select a support ticket that has 'Unassigned' on it.
    - Click on the button 'Assign'.
    - Click on admin's username when drop down appears.
    - Click 'Ok' button.
    - Observe the ticket list refreshed and the chosen ticket has admin's username on it now.
- Post Conditions: Support ticket is no longer 'Unassigned' and instead has admin's username on it

Test Case ID: 5.4 – User Management Portal, Support Ticket Management

- Description: Admin with ticket permission selects an assigned ticket and attempts to assigns to self
- Test Inputs: Assigned support ticket and admin token
- Expected Results: Support ticket is updated to have admin's username on it
- Dependencies: Admin login portal, database, server
- Initialization: Active admin bearer token associated with an admin id that has tickets permission in it, clicked on ticket management action card/tab, and can observe a list of support tickets on the page
- Test Steps:
    - Select a support ticket that has another admin's username on it.
    - Click on the button 'Assign'.
    - Click on admin's username when drop down appears.
    - Click 'Ok' button.
    - Observe the ticket list refreshed and the chosen ticket has admin's username on it now.
- Post Conditions: Support ticket's previously "assigned to" value changed from another admin's username to the new admin's username

Test Case ID: 6.1 – User Management Portal, Inactive User Management

- Description: Admin with inactive user management permission enters user management portal
- Test Inputs: Active admin bearer token associated with an admin id is allowed user management permission in it
- Expected Results: Manage Inactive Users action card/button is on user management portal homepage

- Dependencies: Admin login portal, database, server
- Initialization: Admin successfully logged in and was successfully redirected to user management portal page
- Test Steps:
    - Wait for home page to load
    - Verify “Manage Inactive Users” Card or Tab is on the page.
- Post Conditions: Admin can see the inactive user management card/tab on the user management portal page

Test Case ID: 6.2 – User Management Portal, Inactive User Management

- Description: Admin with no inactive user management permission enters user management portal
- Test Inputs: Active admin bearer token associated with an admin id is does not have inactive user management permission in it
- Expected Results: Manage Inactive Users action card/button is not on user management portal homepage
- Dependencies: Admin login portal, database, server
- Initialization: Admin successfully logged in and was successfully redirected to user management portal page
- Test Steps:
    - Wait for home page to load
    - Verify “Manage Inactive Users” Card or Tab is not on the page.
- Post Conditions: Admin cannot view or interact the inactive user management card/tab on the user management portal page

Test Case ID: 6.3 – User Management Portal, Inactive User Management

- Description: Admin with inactive user permission selects a user from the inactive user table and attempts to delete user
- Test Inputs: Inactive user’s id and admin token
- Expected Results: Dialog appears on screen notifying the admin that the user was sent an email regarding inactivity and will be deleted if not logged in.
- Dependencies: Admin login portal, database, server
- Initialization: Active admin bearer token associated with an admin id that has inactive user management permission in it, clicked on manage inactive users action card/tab, and can observe a list of inactive users on the page
- Test Steps:
    - Select an inactive user.
    - Click on the button ‘Remove User’
    - Click on ‘Ok’ when dialog appears to confirm removal.
    - Observe the dialog popup stating the user has been notified and will be removed if last login does not change in the next 3 days.
- Post Conditions: Informed Inactive Users do not populate on list after refresh.

Test Case ID: 6.4 – User Management Portal, Inactive User Management

- Description: Admin with inactive user permission selects a user from the inactive user table and does not attempt to remove the user

- Test Inputs: Inactive user's id and admin token
- Expected Results: List still contains inactive user
- Dependencies: Admin login portal, database, server
- Initialization: Active admin bearer token associated with an admin id that has inactive user management permission in it, clicked on manage inactive users action card/tab, and can observe a list of inactive users on the page
- Test Steps:
    - Select an inactive user.
    - Click on the button 'Remove User'
    - Click on 'Cancel' when dialog appears to cancel removal.
    - Observe the user is not removed from the list.
- Post Conditions: The inactive user that was canceled for removal populates on list after refresh.

Test Case ID: 7.1 - Renting Wifi

- Description: User rents wifi from website
- Test Inputs:
    - User credentials
    - Wifi specification
    - Rental agreement
    - Payment info
- Expected Results: Upon confirmation of user's rental, they will receive access to the rented wifi service, enabling seamless connectivity for their devices during the specified rental period.
- Dependencies: User authentication system, Payment gateway, Wifi selection interface
- Test Steps:
    - User provides credentials
    - User selects wifi to rent
    - User selects specific rental term and agrees to conditions
    - Input payment info
    - Submit request to rental
- Post Conditions: User gets confirmation and access to rented wifi

Test Case ID: 7.2 - Error Handling: Incomplete Wifi Rental

- Description: user attempts to rent without complete info
- Test Inputs:
    - User credentials
    - Missing input (Such as missing payment info)
- Expected Results: Error message prompting user to fill out every info field
- Dependencies: User rental interface, User authentication system
- Test Steps:
    - Provide user credentials
    - Choose wifi rental
    - Skip or incomplete rental fields
    - Attempt to submit rental agreement
- Post Conditions: error message prompting user to complete the rental information

Test Case ID: 8.1 -Wifi listing

- Description: User lists their wifi on website to rent
- Test Inputs:
    - User login
    - Wifi device details
    - Rental terms
    - Agreement to listing terms and conditions
- Expected Results:
    - Confirmation of successful listing
    - Wifi listing published on website
- Dependencies: User authentication system, Listing interface
- Test Steps:
    - User logs in
    - User inputs wifi details
    - User agrees to listing terms and conditions
    - User submits listing
- Post Conditions: User receives confirmation message stating listing was successful, wifi listing published.

Test Case ID: 8.2 - Error Handling: User tries to list wifi with incomplete details

- Description: User attempts to list wifi with incomplete information.
- Test Inputs:
    - User login
    - incomplete/invalid wifi listing information
    - Listing term not agreed to
- Expected Results: Error message prompting user to complete listing info
- Dependencies: Listing interface
- Test Steps:
    - User logs in with credentials
    - User inputs incomplete or invalid wifi device info or rental terms
    - User attempts to submit listing
- Post Conditions: Error message prompting user to input valid and complete information.

Test Case ID: 9.1 - Managing wifi listings

- Description: User can manage their wifi listings
- Test Inputs:
    - User credentials
    - Access to listing management section, located in user dashboard
    - Selection of specific listing to edit
- Expected Results: Confirmation that users changes were made and saved. Changes and edits published to whatever user has changed
- Dependencies: User authentication, Listing management page
- Test Steps:
    - User logs in
    - User navigates to listing management in dashboard

    - Selection of specific listing to edit
    - Necessary changes are made
    - Changes are saved
- Post Conditions: User receives message of confirmation changes and listing is updated accordingly

Test Case ID: 9.2 - Listing removal confirmation
- Description: User removes listing, receives confirmation
- Test Inputs:
    - User credentials
    - Access to listings in users dashboard
    - Specific listing to remove
- Expected Results: Confirmation of listing being removed. Listing successfully removed from website
- Dependencies: Listing management interface, User authentication
- Test Steps:
    - User logs in
    - User navigates to listing management in dashboard
    - Selection of specific listing to remove
    - Confirm removal
- Post Conditions: User receives confirmation that listing is removed.

Test Case ID: 10.1-User Registration
- Description: User accesses the registration page, fills out the form with accurate details, and submits it. The system validates the information, storing the details in the database and confirms successful registration.
- Test Inputs: Valid inputs entered
- Expected Results: User's registration details are successfully stored in the database and system confirms successful registration.
- Dependencies: Database, server
- Test Steps:
    - Navigate to user registration page
    - Enter valid username, email, password, name, dob, and phone number
    - Click on the “Sign up” button
    - Verify registration is successful by logging in with the registered username and password
- Post Conditions: User account is created in and redirected to the login page

Test Case ID: 10.2-User Registration, Invalid Email Address
- Description: User accesses the registration page, fills out the form with invalid email address.
- Test Inputs: Valid inputs entered with the exception of an invalid email address
- Expected Results: System displays an error message indicating that the entered email address is invalid.
- Dependencies: Database, server
- Test Steps:

    - Navigate to user registration page
    - Enter valid username, password, name, dob, phone number and invalid email address
    - "Sign Up" button is disabled until a valid email address is entered
- Post Conditions: The user account is not created or stored in the system's database

Test Case ID: 11.1- Payment Confirmation

- Description: User receives a confirmation message containing transaction details, including ID, amount, date, time, and service summary. If needed, the user can contact customer support, and the transaction details are logged in the database for record-keeping.
- Test Inputs: Valid transaction id with the payment amount and summary of what was purchased
- Expected Results: Confirmation message with payment details is generated and the user receives it
- Dependencies: Login module, payment system, database, server
- Test Steps:
    - Initiate payment transaction and process it
    - System generates confirmation message and displays it
    - Review payment details
    - System logs transaction details
- Post Conditions: User is redirected to the home page with the system ready to process any further transactions

Test Case ID: 11.2- Payment Confirmation, Network Issues

- Description: User faces network issues accessing payment information
- Test Inputs: Valid transaction id with the payment amount for payment confirmation message to be generated
- Expected Results: Due to network issues the payment confirmation message fails to load. User may retry accessing the payment confirmation message after resolving network issues or contact customer support.
- Dependencies: Login module, payment system, database, server
- Test Steps:
    - Initiate payment transaction and the system attempts to process it
    - Due to network issues the system encounters difficulty retrieving payment confirmation details
    - Website displays an error message indicating the inability to fetch payment confirmation details due to network problems
- Post Conditions: User receives error message until network issues are resolved

Test Case ID: 12.1-User Feedback

- Description: User engages with the feedback interface, entering feedback which is submitted, processed, and stored in the database. Upon successful submission, a confirmation message is shown to the user.
- Test Inputs: User inputs feedback submission that fits word count limit

- Expected Results: The user feedback is successfully recorded and stored in the system's database. A confirmation message acknowledges the successful submission of the feedback.
- Dependencies: Login module, database, server
- Test Steps:
    - User is logged in and accesses the feedback submission form
    - User provides feedback regarding product and submits it
    - System records the feedback
    - Confirmation message indicates that it was successful
- Post Conditions: User is redirected to the homepage

Test Case ID: 12.2- User Feedback, Exceeds Word Count

- Description: User engages with the feedback interface
- Test Inputs: User inputs feedback submission that exceeds the word count limit
- Expected Results: Website displays a message informing the user that the entry exceeds the word count limit. The entry can not be submitted unless shortened.
- Dependencies: Login module, database, server
- Test Steps:
    - User is logged in and accesses the feedback submission form
    - User attempts to provide feedback but the entry exceeds the specified word count limit
    - A warning message appears informing the user that the feedback exceeds the word count limit
    - The "submit" button will not work unless the entry is shortened
- Post Conditions: The user feedback is not recorded

Test Case ID: 13.1–System Maintenance, Admin Maintenance Control

- Description: Admin with permission level that allows the admin to trigger site down for maintenance attempts to notify users of scheduled maintenance.
- Test Inputs: Admin token with admin id associated with site maintenance permission
- Expected Results: Banner appears in every page of the site with information of scheduled site maintenance
- Dependencies: Login module, DB, server
- Initialization: Given admin logged in, has an active valid token with admin id associated with site management and is on admin homepage version
- Test Steps:
    - Click "Start site management notification" button
    - Submit start datetime and estimated end datetime.
    - Submit a max 100 character sentence along the lines of "Site Down for Maintenance: <Reason Here>"
    - Click submit.
    - Observe that there is a banner at the top of every page.
- Post Conditions: There is a site down for maintenance banner at the top of every page.

Test Case ID: 13.2–System Maintenance, Admin Maintenance Control

- Description: Admin with permission level that allows the admin to trigger site down for maintenance attempts to notify users of scheduled maintenance.
- Test Inputs: Admin token with admin id associated with site maintenance permission
- Expected Results: Banner appears at the top of every page of the site with information of scheduled site maintenance
- Dependencies: Login module, DB, server
- Initialization: Given admin logged in, has an active valid token with admin id associated with site management and is on admin homepage version
- Test Steps:
    - Click “Start site management notification” button
    - Click End “Notification Banner”
    - Click Ok in confirmation dialog pop up.
    - Observe that there is no longer a banner at the top of every page.
- Post Conditions: There is no longer a site down for maintenance banner at the top of every page. top of every page.

Test Case ID: 14.1–User Account Deactivation, Remove Users

- Description: Admin with permission level of user management attempts to remove user and user’s account credentials after initial notification’s time window has past
- Test Inputs: Admin token with admin id associated with user management, notified inactive user that did not log in within the stipulated time window
- Expected Results: Querying for that user’s User info and associated user account return null
- Dependencies: Login module, DB, server
- Initialization: Given admin logged in, has an active valid token with admin id associated with user management and clicked on user management action card in User management portal
- Test Steps:
    - Filter table to find notified users that past time window
    - Click “Permanently remove user and user account”
    - Click confirm in Confirmation dialog pop up
    - Click “Test query in DB”
    - Enter user’s username
    - Observe no rows were found
- Post Conditions: Notified user that did not login within the stipulated time window and respective user account is no longer in the database.

Test Case ID: 14.3–User Account Deactivation, Remove Users

- Description: Admin with permission level of user management cancels attempt to remove user and user’s account credentials after initial notification’s time window has past
- Test Inputs: Admin token with admin id associated with user management, notified inactive user that did not log in within the stipulated time window
- Expected Results: Querying for that user’s User info and associated user account return null
- Dependencies: Login module, DB, server

- Initialization: Given admin logged in, has an active valid token with admin id associated with user management and clicked on user management action card in User management portal
- Test Steps:
    - Filter table to find notified users that past time window
    - Click “Permanently remove user and user account”
    - Click cancel in Confirmation dialog pop up
    - Click “Test query in DB”
    - Enter user’s username
    - Observe a row was found with the user’s username and user id
- Post Conditions: Notified user that did not login within the stipulated time window and respective user account is still in the database.

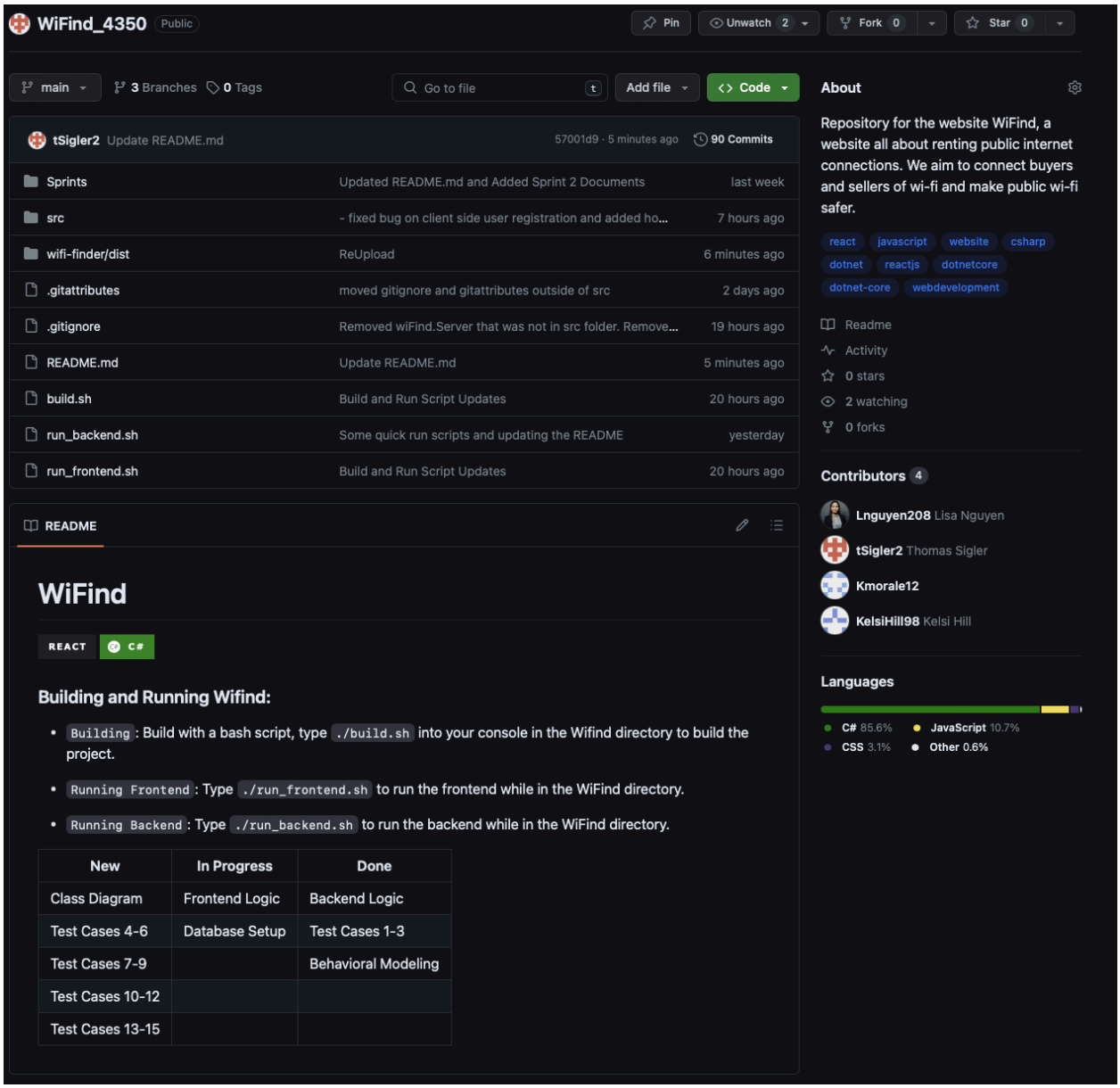
Test Case ID: 15.1–Web Accessibility, Font Size Adjustment (Increase)

- Description: User attempts to change font sizes to a larger size.
- Test Inputs: Desired text size
- Expected Results: All pages are updated to have the chosen font size.
- Dependencies: N/A
- Test Steps:
    - User clicks on Web Accessibility button
    - User uses incrementer to increase font size
    - User clicks “Confirm Changes” button
    - Observe the page font size was changed to the chosen increased font size
- Post Conditions:
    - Pages in the site uses the chosen increased font size for texts.

Test Case ID: 15.2–Web Accessibility, Font Size Adjustment (Decrease)

- Description: User attempts to change font sizes to a smaller size.
- Test Inputs: Desired text size
- Expected Results: All pages are updated to have the chosen font size.
- Dependencies: N/A
- Test Steps:
    - User clicks on Web Accessibility button
    - User uses incrementer to decrease font size
    - User clicks “Confirm Changes” button
    - Observe the text’s font size on the page was changed to the chosen decreased font size
- Post Conditions:
    - Pages in the site use the chosen decreased font size for texts.

Section 11: Github Link and Screenshot



https://github.com/tSigler2/WiFind_4350