# 4-Wheel Drive Used Car Dealership

Maintenance Manual

Fall 2020

Version 2.0

|  |  |
|---|---|
| CEO: | Dr. Jennifer Jin |
| Prepared By: | Alyssa Wilcox<br>Wen Zhang |

# 1. File Structures

The project's file structure is managed using Google Drive. All of our files are saved under the Project Documentation root directory. Source code is specifically saved under the Source Code directory, where each module has its own directory where relevant code and updates can be stored. The parent directory, and subsequently all subdirectories, are accessible to all team members, where each member can upload, download, and view files. Each newest update is saved in its own folder with the date and brief description.

**404.html**

This is a page provided by Firebase when initializing a project that serves as a default display page if another page is not able to be found and displayed on a web browser.

**index.js**

This actively serves as the Homepage for all users. It contains functions that direct users to other parts of the web application when users press on buttons on the homepage.

**aboutUs.html**

This actively serves as the About Us page for all users. It displays information about the 4-Wheel Drive Used Car Dealership company.

**auth.js**

This actively checks the user's credentials and has functions that display different links in the navigation bar depending on the credentials of the user. This function also handles the display of different buttons options in the vehicle details page based on the login credentials of the user. Additionally, this file contains the function that logs users out of their accounts.

**login.js**

This actively serves as the Login page for all users. It contains a login function that signs in a user with their email and password. Error handling is incorporated into this function to inform the user of a failed login due to bad input or a server error. Upon completion of this function, a success function is called that displays a success message below the login field. A redirect function is then called to redirect the user to the homepage after logging in. The Login page also includes links to the Request Password Change page and Create Account page.

**createAccount.js**

This actively serves as the Create Account page for all users. It contains a create account function that creates a new users account based on the input email and password. Error handling is incorporated into this function to inform the user of a failed create account attempt due to bad input or a server error. Upon successful creation of a user account, a success function is called to display a success message below the create account field, and the user is signed into their new account. A redirect function is then called to redirect the user to the homepage. The Create Account page also includes links to the Request Password Change page and the Login page.

**requestPSchange.js**

This actively serves as the Request Password Change page for all users. It contains a function that sends a password reset email to the email input by the user. Error handling is incorporated into this function to inform the user of a failed submission attempt due to bad input or a server error. Upon a successful submission, an email is sent to the input email and a success function is called. This function displays a success message below the input field. A redirect function is then called that redirects the user to the Login page. This file also includes links to the Create Account page and Login page.

**inventory.js**

This actively displays the full inventory on the Inventory page if a user has pressed the "See Our Full Inventory" button on the homepage. It contains a function that reads the data for each vehicle stored in Cloud Firestore, and it then dynamically creates HTML elements to display the vehicle's data on the Inventory page. After the data has been successfully read and displayed, a get images function is called to retrieve each car's image from Firebase Storage. Upon retrieval, the image is displayed on the Inventory page along with the relevant car data. This function handles a failed attempt to retrieve a car's image, meaning that an image was not uploaded for that car. Additionally, this file contains a function that redirects a user once they have clicked on a car and want to see more information.

**vehicleDetails.js**

This actively serves as the Vehicle Details page for all users. It contains a function that retrieves both the picture and data for the vehicle that was clicked on in the Inventory page, and then displays the picture and data on the page.

**myFavorites.js**

This actively serves as the My Favorites page for all logged in regular users. It contains a function that retrieves the picture and data for vehicles logged in users have saved to the My Favorites page, and then displays them on the page. A function also handles a logged in user deleting a vehicle from their My Favorites page.

**addVehicle.js**

This actively serves as the Add Vehicle page for all logged in employee users. It contains a function that retrieves all data and an image input by the employee user and then uploads that data and image to Cloud Firestore and Firebase Storage respectively. Error handling is incorporated into this function to inform the user of a failed submission due to bad input or a server error. Upon a successful submission, a success function is called that displays a success message on screen for the employee user. After three seconds, a delete message function is called that deletes this success message, removing it from the screen. Additionally, upon a successful submission, the input form is cleared, allowing the employee user to add another vehicle to the database.

**addToFavorites.js**
This actively allows logged in users to add vehicles to their My Favorites page through the use of a button on the vehicle details page. This file also allows users to remove a vehicle from their My Favorites page through the vehicle details page if they are viewing a vehicle already on their My Favorites page.

**deleteVehicle.js**
This actively allows logged in employee users to delete vehicles from the Cloud Firestore database through the vehicle details page.

**modifyVehicle.js**
This actively allows logged in employee users to modify the data and pictures of a vehicle in the database.

**specialDeals.js**
This actively allows logged in employee users to add and remove vehicles from a special deals list through the vehicle details page.

**displaySpecialDeals.js**
This actively displays vehicles marked as a special deal onto the homepage. It also redirects users to the vehicle details page upon a user clicking on a special deal vehicle.

**search.js**
This actively works as a search algorithm in the inventory page. Users are able to submit keywords to the search bar on both the homepage and the inventory page. The file then reads all vehicle information from the database and saves it into a two dimensional vector. A recursive search function is then used to search the vehicles with each keyword, removing any vehicles from the search results that do not match the keywords input. After the correct vehicles have been found, the results are displayed for the user on the inventory page.

**resetSearch.js**
This actively allows users to reset their search submission, by displaying the full inventory in the inventory page.

**filterFullInventory.js**
This actively allows users to filter the vehicles on the inventory page when the full inventory is being displayed. Specifically, users are able to filter by year ascending, year descending, price ascending, and price descending.

**filterSearchResults.js**
This actively allows users to filter the vehicles provided by the search results on the inventory page. Specifically, users are able to filter by year ascending, year descending, price ascending, and price descending.

## 2. Instructions

The application can be run using on of the following options listed below:

1. Using a URL generated and hosted by Firebase (recommended).
2. Emulating it on a Windows Computer.
3. Emulating it on Lynx or Mac.

To run the application through any of the above methods, one must:

1. Use a laptop or desktop computer that is connected to the internet.
2. Use Google Chrome to access or emulate the application.

To access functionalities for employee users, one can log in with this account:

- Email: Employee1@gmail.com
- Password: 123Emp1

**Method 1:** To run the application through a URL generated and hosted by Firebase, the following steps must be taken:

1. Use a laptop or desktop computer that is connected to the internet.
2. Actively run Google Chrome (other browsers will not work).
3. Type the URL: https://wheeldriveusedcardealership.web.app/index.html into the address bar.
4. Press enter.

**Method 2:** To run the application through emulating it on a Windows Computer, the following steps must be taken:

1. Install Node.js
    a. Go to https://nodejs.org/en/
    b. Click on "Recommended for most users"
    c. Follow the installation details provided by the installation wizard
2. Install Firebase CLI

    a. Open up your operating system’s command prompt
    b. Type “npm install -g firebase-tools” onto the command prompt
    c. Press enter
3. Navigate to the folder you want store the project in
    a. (optional) In the command prompt, change the harddrive you are currently accessing by inputting *hardrive_letter:* (Example, E: to access the E harddrive)
    b. In the command prompt, navigate to the folder you want to store the project in using “cd *folder_name*”, navigating folder by folder until you are in the desired folder
        i. You can create a folder using “cd *folder_name*”
4. Log into the Google Account associated with the project
    a. In the command prompt, input “firebase login”
    b. Make sure you are logged into the correct Google account
        i. If you are signed into a different account:
            1. Type “firebase logout”
            2. Type “firebase login”
            3. You should be brought to a screen allowing you to choose which Google account to login. Follow those directions and log into the shared account
            4. Type “firebase login” again to the command prompt to confirm that you are logged into the correct account
5. Deploy to Firebase Hosting
    a. Input “firebase init” and press enter
    b. On prompt “Are you ready to proceed?” type “Y” for yes
    c. On prompt “Which Firebase CLI features do you want to set up for this folder?” select the following options using space and the arrow keys and then press enter:
        i. Firestore
        ii. Hosting
        iii. Storage
        iv. Emulators
    d. Project Setup:
        i. On prompt “Please select an option:” use the arrow keys to select “Use an existing project”. Press enter
        ii. On prompt “Select a default Firebase project for this directory:” Select the project “wheeldriveusedcardealership” and press enter
    e. Firestore Setup:
        i. On prompt “What file should be used for Firestore Rules?” press enter
        ii. On prompt “What file should be used for Firestore indexes?” press enter
    f. Hosting Setup:

        i. On prompt "What do you want to use as your public directory?" press enter
        ii. On prompt "Configure as a single-page app (rewrite all urls to /index.html)?" type "N" and press enter
        iii. On prompt "Set up automatic builds and deploys with GitHub?" type "N" and press enter
    g. Storage Setup:
        i. On prompt "What file should be used for Storage Rules?" press enter
    h. Emulators Setup:
        i. On prompt "Which Firebase emulators do you want to set up?" select the following options with space and the arrow keys and then press enter
            1. Authentication Emulator
            2. Firestore Emulator
            3. Hosting Emulator
        ii. On prompt "Which port do you want to use for the auth emulator?" press Enter
        iii. On prompt "Which port do you want to use for the firestore emulator?" press Enter
        iv. On prompt "Which port do you want to use for the hosting emulator?" press Enter
        v. On prompt "Would you like to enable the Emulator UI?" type "Y" and press Enter
        vi. On prompt "Which port do you want to use for the Emulator UI?" press Enter
        vii. On prompt "Would you like to download the emulators now?" type "y" and press Enter

6. Download the appropriate files
    a. Use the link: to access the appropriate files
    b. Download the files to the "public" folder associated with your Firebase Project
7. Emulate the project.
    a. In the command prompt, navigate to the folder that contains your Firebase Project
    b. Type "firebase serve" and press enter
    c. Look for the URL generated that hosts the project (should be http://localhost:5000)
    d. Type in the hosting URL into Google Chrome's address bar and press enter
8. Close the emulation.
    a. When you are ready to close and stop the emulation, type "CTRL + C" into the command prompt
    b. On prompt "Terminate batch job?" type "Y" and press Enter
9. Run the emulation again.

    a. To run the emulation any time after the initial installation, simply repeat steps 7 and 8

**Method 3:** To run the application through emulating it on Lynx or Mac, the following steps must be taken:

1. Install Node.js
    a. Go to [https://nodejs.org/en/](https://nodejs.org/en/)
    b. Click on "Recommended for most users"
    c. Follow the installation details provided by the installation wizard
2. Install Firebase CLI
    a. Open up your operating system's command prompt
    b. Type "curl -sL https://firebase.tools | bash" onto the command prompt
    c. Press enter
3. Navigate to the folder you want store the project in
    a. (optional) In the command prompt, change the harddrive you are currently accessing by inputting *hardrive_letter:* (Example, E: to access the E harddrive)
    b. In the command prompt, navigate to the folder you want to store the project in using "cd *folder_name*", navigating folder by folder until you are in the desired folder
        i. You can create a folder using "cd *folder_name*"
4. Log into the Google Account associated with the project
    a. In the command prompt, input "firebase login"
    b. Make sure you are logged into the correct Google account
        i. If you are signed into a different account:
            1. Type "firebase logout"
            2. Type "firebase login"
            3. You should be brought to a screen allowing you to choose which Google account to login. Follow those directions and log into the shared account
            4. Type "firebase login" again to the command prompt to confirm that you are logged into the correct account
5. Deploy to Firebase Hosting
    a. Input "firebase init" and press enter
    b. On prompt "Are you ready to proceed?" type "Y" for yes
    c. On prompt "Which Firebase CLI features do you want to set up for this folder?" select the following options using space and the arrow keys and then press enter:
        i. Firestore
        ii. Hosting
        iii. Storage
        iv. Emulators
    d. Project Setup:

        i. On prompt “Please select an option:” use the arrow keys to select “Use an existing project”. Press enter
        ii. On prompt “Select a default Firebase project for this directory:” Select the project “wheeldriveusedcardealership” and press enter
    e. Firestore Setup:
        i. On prompt “What file should be used for Firestore Rules?” press enter
        ii. On prompt “What file should be used for Firestore indexes?” press enter
    f. Hosting Setup:
        i. On prompt “What do you want to use as your public directory?” press enter
        ii. On prompt “Configure as a single-page app (rewrite all urls to /index.html)?” type “N” and press enter
        iii. On prompt “Set up automatic builds and deploys with GitHub?” type “N” and press enter
    g. Storage Setup:
        i. On prompt “What file should be used for Storage Rules?” press enter
    h. Emulators Setup:
        i. On prompt “Which Firebase emulators do you want to set up?” select the following options with space and the arrow keys and then press enter
            1. Authentication Emulator
            2. Firestore Emulator
            3. Hosting Emulator
        ii. On prompt “Which port do you want to use for the auth emulator?” press Enter
        iii. On prompt “Which port do you want to use for the firestore emulator?” press Enter
        iv. On prompt “Which port do you want to use for the hosting emulator?” press Enter
        v. On prompt “Would you like to enable the Emulator UI?” type “Y” and press Enter
        vi. On prompt “Which port do you want to use for the Emulator UI?” press Enter
        vii. On prompt “Would you like to download the emulators now?” type “y” and press Enter
6. Download the appropriate files
    a. Use the link: to access the appropriate files
    b. Download the files to the “public” folder associated with your Firebase Project
7. Emulate the project.
    a. In the command prompt, navigate to the folder that contains your Firebase Project
    b. Type “firebase serve” and press enter

    c. Look for the URL generated that hosts the project (should be http://localhost:5000)
    d. Type in the hosting URL into Google Chrome's address bar and press enter
8. Close the emulation.
    a. When you are ready to close and stop the emulation, type "CTRL + C" into the command prompt
    b. On prompt "Terminate batch job?" type "Y" and press Enter
9. Run the emulation again.
    a. To run the emulation any time after the initial installation, simply repeat steps 7 and 8

## 3. Good Implementation

Positive implementations of the application:

- The application is pleasant in appearance.
- The app has a simple layout on each page of the application.
- The application contains error handling that should prevent unexpected crashes/undesired output.
- The application is interactive, with visual cues indicating what a user can interact with.

## 4. Needs Improvements

The following are weak points of the current implementation:

- The application only works on Google Chrome.
- The application is not mobile compatible, and it only works on a laptop or desktop computer.
- The application does not display properly on small screens.
- The application's database is limited due to Firebase's restrictions for free accounts.

## 5. Overall Recommendations

If a future development team were to work on our application, we would recommend:

- Enhancing the filter feature to include more options.
- Adding a business statement in the About Us page.
- Improving the source code by removing duplicate code.
- Making the application compatible on different browsers.