

# Technical Requirements Document

## Car Dealership Website

Team: Bassem Abou Daher, Omar Succar, Salim Hashisho, Maya Chami, Nancy Abdallah, Leah Sukkarieh

March 1, 2023

## Introduction

### Overview

Purchasing a new car is often a tedious task: visiting the car dealership and exploring different options of cars, having to inquire about the specs of each, and filling out the paperwork for a test drive. This process is time and energy consuming for people. They are likely to prefer being able to do all these tasks in minutes- using the click of a button. The idea is to introduce car dealerships to the world of e-commerce, a project that will no doubt attract dealership owners. A website would facilitate the car trading process for customers through several features that would focus on a seamless user experience and ease of navigation. The online process would also be easier for dealerships, since there is a time-consuming and costly effort for physical dealerships, as owners struggle with reaching their target audience, managing them, keeping track of their listings, hiring salespeople to advertise the cars and discuss their specs, as well as the costs of a showroom. Therefore, the stakeholders would be the car dealership, the car salespeople, the customers, and the website developers.

### Glossary

- An API is a set of protocols, tools and rules that act as an intermediary between two different software applications, allowing them to interact with each other and exchange information in a standardized way.
- Backend: The part of a website or application that is responsible for processing data and serving information to the frontend.
- Authentication: The process of verifying the identity of a user or system.
- Database: A structured collection of data that is stored on a computer system and can be accessed and managed by software.
- Frontend: The part of a website or application that users interact with directly, including the user interface and design.
- Web server: A computer system that hosts websites and serves web pages to clients who request them.
- Debugging: The process of finding and fixing errors or bugs in software.

- User Experience (UX) - the overall experience of a user while interacting with the website, including ease of use, functionality, and satisfaction.
- User Interface (UI) - the graphical layout and design of the website, including visual elements such as buttons, icons, and menus.

## Background

With the ever-growing importance of e-commerce, users are opting to shop online due to ease and accessibility. In the 21st century, it's indispensable for any business to have its own website. The traditional method of advertising and selling cars through print ads and in-person visits to the dealership is becoming less effective in the digital age. Car dealerships taking their business online offers them the opportunity to reach a wider range of their target audience in a more affordable manner and allows them to manage their listings and customers more efficiently. Customers would also benefit from being able to easily browse and research cars at their leisure before making a purchase decision.

This product aims to accomplish a simpler experience for both the user and the dealer, by allowing the customers to easily browse listings, specs, and details as well as request test drives, and enabling the dealer to efficiently manage their listings and schedule test drives. It will be designed for one car dealership with a set number of listings, but the roadmap envisions that it will be scaled to handle more. Thus, many considerations regarding scalability, performance, compliance, and completeness have been put in place to achieve the end goal of this project.

## Goals

### Product Requirements:

- As a customer, I want to be able to sign in or create an account.
- As a customer, I want to filter certain car characteristics to find the car that matches my needs.
- As a customer, I want to fill a test drive form and be able to schedule a suitable time, which I can also change later.
- As a customer, I want to have a smooth viewing experience on different devices and browsers.
- As a customer, I want the website to load quickly so I can find the information I need.
- As a dealer (admin), I want to be able to log in.
- As a dealer, I want to be able to edit listings, add listings, and delete listings.
- As a dealer, I want to view test drive request forms so I can schedule a time that fits both the customer and my schedule.
- As a dealer, I want to track customers preferences and previous interactions.
- As a dealer, I want to view the status of my business (sales, number of cars sold, scheduled test drives...)

## Technical Requirements:

- Frontend: Use React.js to develop a user-friendly interface, implementing simple and straightforward navigation.
- Backend: Use Node.js (JavaScript, HTML, CSS) for its compatibility with React.js. Use JSON to interact with the server.
- Database: Use MongoDB for its high availability, scalability, and flexibility which aligns with the product non-functional requirements.
- Deployment: Use Heroku to easily deploy the webapp into the cloud.
- Analytics: Use Google Analytics to continuously test performance and check compliance with the non-functional requirements.
- Integration with an authentication system to manage user access and permissions.
- Integration with a third-party service for email notifications and alerts.
- Integration with a third-party service for processing and managing test drive requests.

## Out of Scope

- No payment gateway to reserve a car (online purchasing not implemented).
- Media upload is restricted to images (no video support).
- Translation of the website into multiple languages.

## Future Goals

- Chat functionality- a chat bot or customer service representative can answer queries on the cars.
- Implement two-factor authentication for the dealer for added security.
- Implementation of social media integration for sharing car listings and promotions on social media platforms.

## Assumptions

- The car dealership will provide accurate and up-to-date information about the cars they want to list on the website.
- The dealership will have the necessary resources to manage and respond to requests for test drives.
- The website will be hosted on a reliable and secure server.
- Users will provide accurate and truthful information when requesting a test drive.

## Solution

### Current Solution

The current attempt at tackling this problem is using social media to showcase cars. Dealerships have opted to create accounts online where they post pictures of their cars, possibly including specs and details.

Pros	Cons
The admin can reach a wider range of their target audience than using traditional advertising.	Interested customers cannot express interest in cars except through direct messages on the platform, causing the admin to possibly be bombarded with messages, and unable to respond to them efficiently.
The users can browse cars online.	The admin must manage test drive requests manually; checking a calendar and making appointments and taking the time to respond.
The users can express interest by sending a direct message on the platform.	The admin must manually delete or update posts of cars that have been sold.
	The admin must delete posts if they wish to change car details or update images.
	The users are inconvenienced if the admin is overwhelmed with messages and might get a very late response if any at all.
	Management of customers and listings is made harder not easier.

## Suggested Solution

Pros	Cons
The user can easily browse listings without needing an account.	The product needs constant maintenance to remain up-to-date with the car information.
The car details, specs, and status are immediately available to the user.	The product reliability is in the hands of its dependencies.
The user is presented with options for available dates for a test drive.	API integration must be done with careful consideration of error handling else product might be unreliable.
The admin can manage listings more efficiently.	Extensive security considerations are required so that no attacker can access admin mode

Pros	Cons
The admin can easily update car information.	
The admin can respond to test drive requests with a click of a button.	
The user has his own account to view and manage his test-drive requests and purchases.	
The solution has low setup cost and few dependencies	
User can compare the specifications of two cars side by side.	
The proposed solution will provide a more user-friendly and efficient way for customers to browse and request test drives for cars.	
The car dealership will be able to update car listings and details in real-time, leading to more accurate and up-to-date information for customers.	
The integration with a test drive scheduling system will automate the process of scheduling test drives, reducing the workload on dealership staff.	

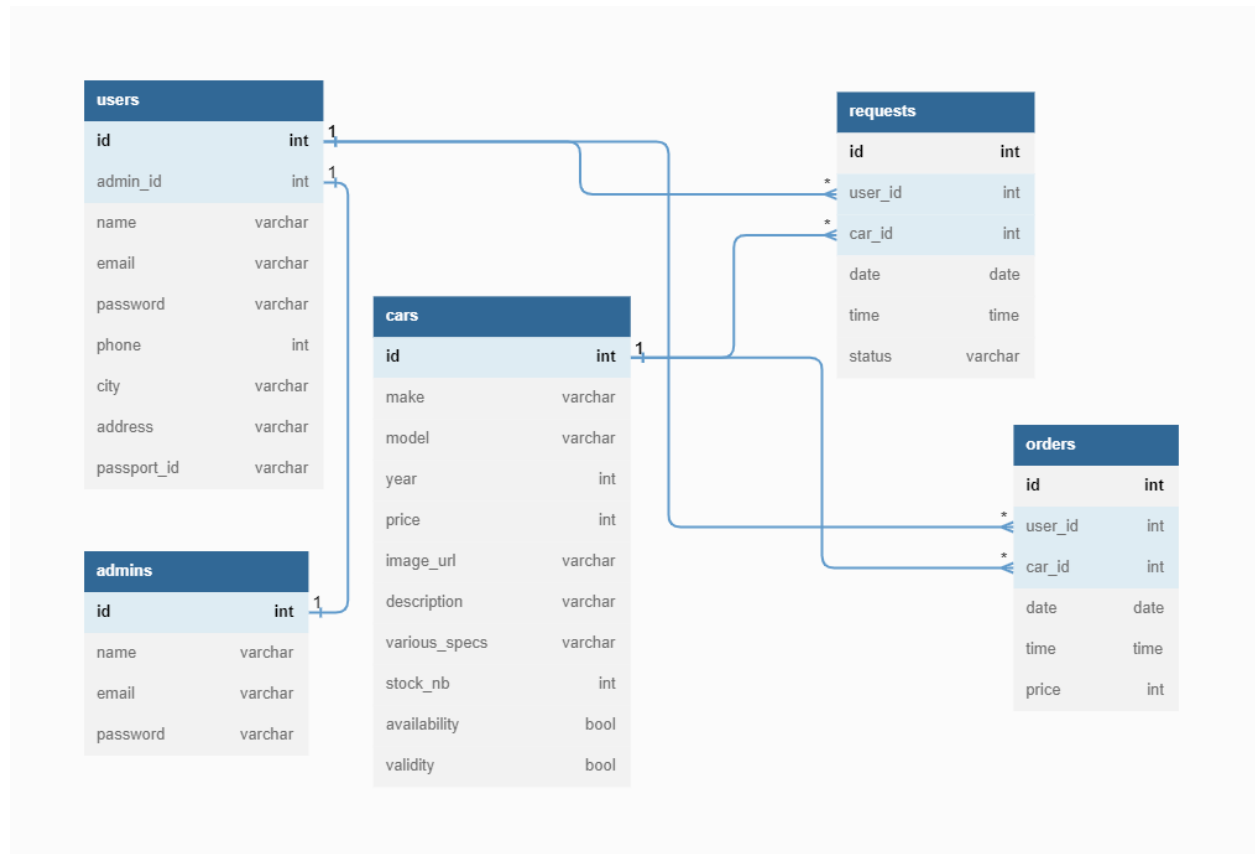
## Dependencies

The product is dependent on the consistency of the database and the proper functioning of the APIs. Any issue with these could render the website with incorrect or missing information.

- Database Management System (DBMS): MongoDB
- Front-end Library: React is used to provide client-side rendering and dynamic UI.

- Runtime environment: Node.js
- Web Framework: Express.js
- Hosting services: Heroku
- Browser compatibility
- APIs (mentioned later)

## Data Schema



The “admins” table is present despite having one admin for the purpose of scalability and future site updates.

## Data Validation Methods:

Front-end validation: Check for needed fields, data type, format, length, and consistency using client-side form validation.

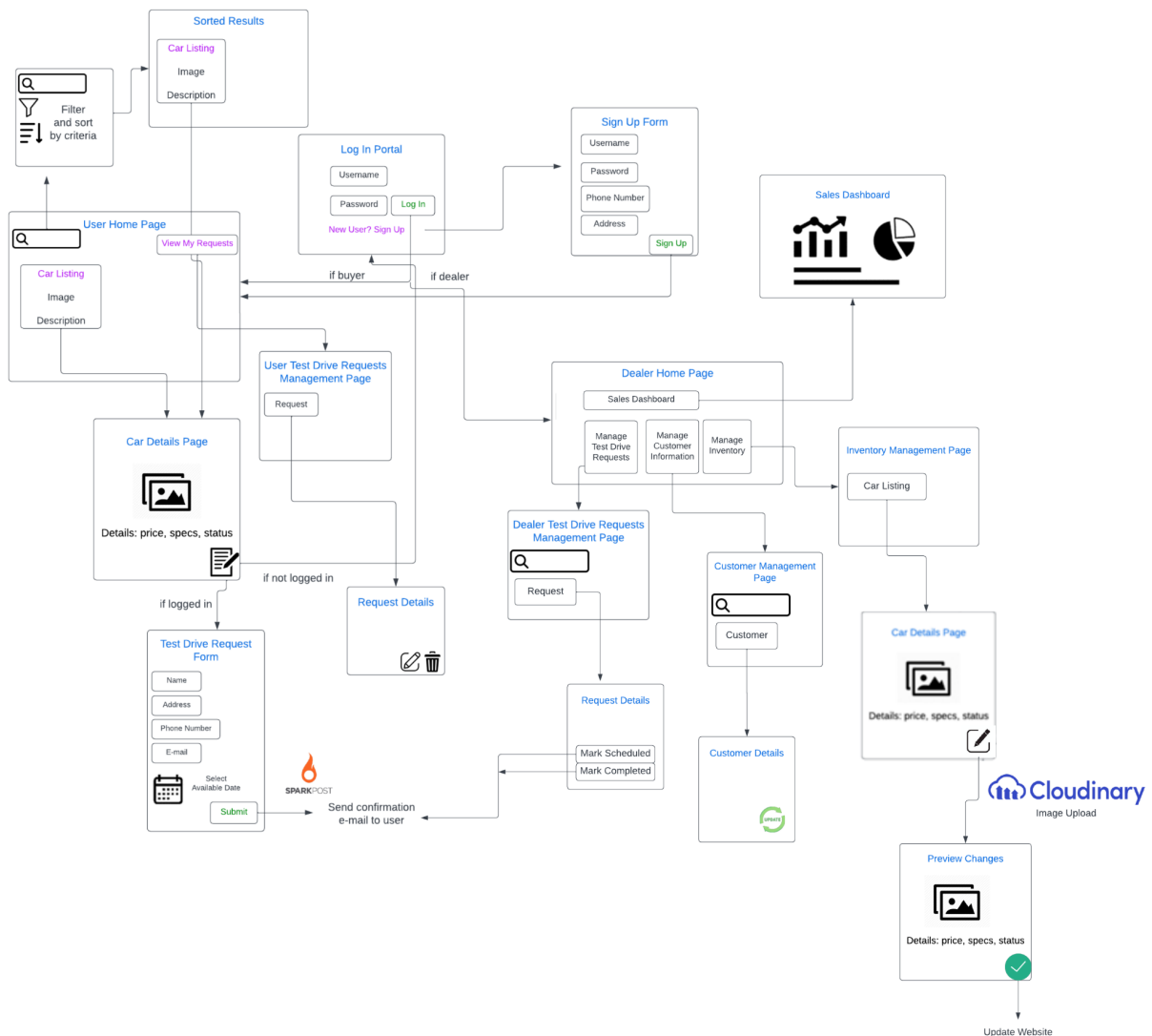
Back-end validation: Check for business logic rules, data integrity, and security threats using server-side validation.

Validation libraries: Since we're using express.js, we can use third-party libraries such as Express Validator for server-side validation.

Regular expressions: use regular expressions to define validation patterns such as email address, phone number, or dates.

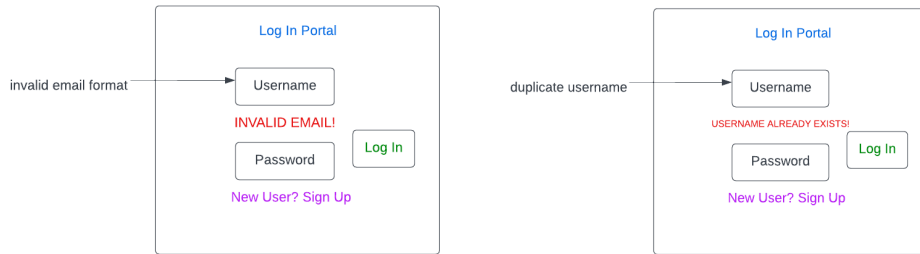
## Business Logic

## UI Diagram and APIs used

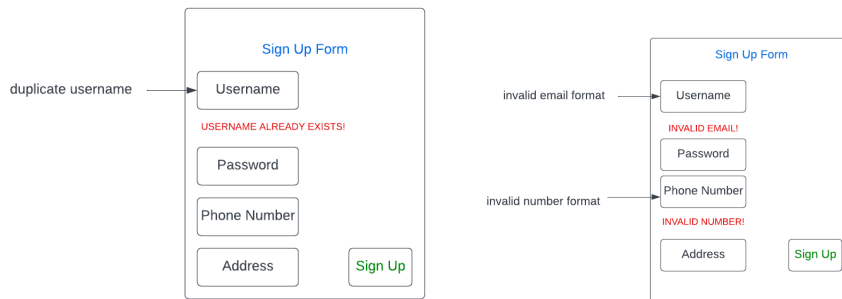


## UI Restrictions:

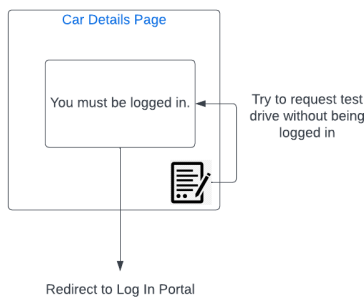
- **Log In:** A user cannot enter a duplicate or invalidly formatted e-mail.



- **Sign Up:** A username needs to be unique. E-mails and phone numbers must have valid format.



- **Test Drive Requests:** A user must be logged in to request a test drive.



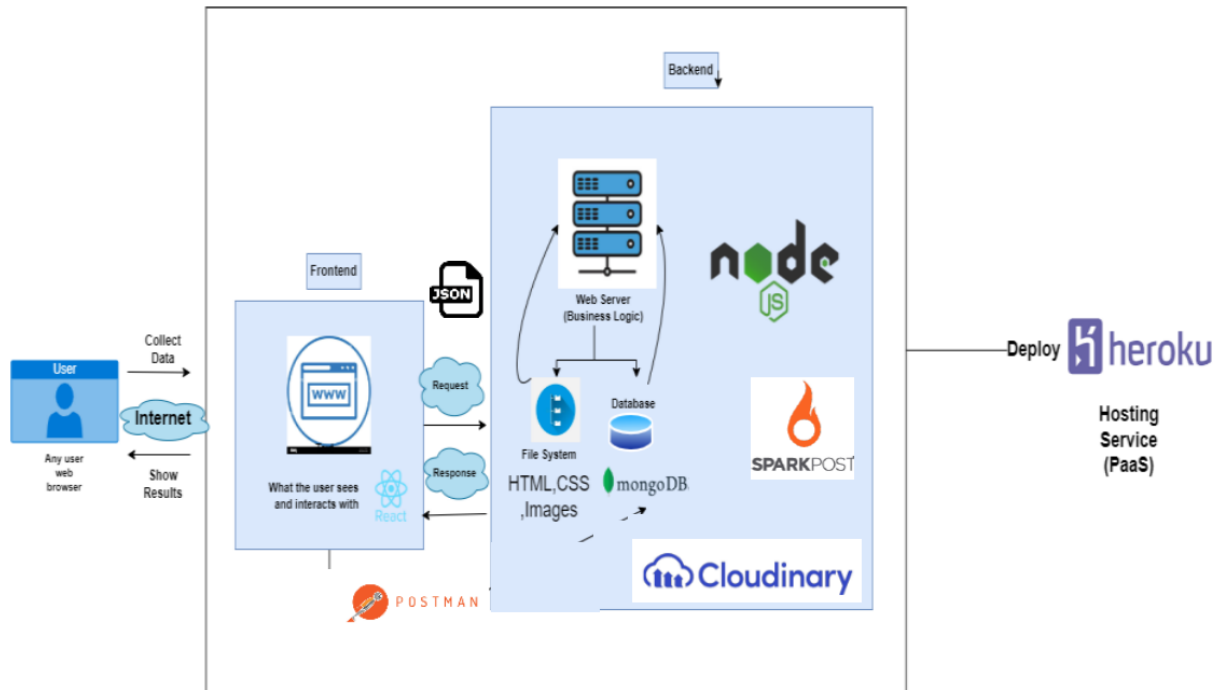
**Schedule Creation:** Schedule will be set by default to working hours of the dealership. Slots of 1 hour will be unavailable if taken and will be made available again when marked as completed.

**Same time scheduling:** To prevent double booking of the same time slot, the relevant rows in our database will be locked during the booking process.

1. When a user selects a time slot, the database is queried to check for availability.
2. If available, the relevant rows in the database will be locked to prevent other users from booking the same time slot while the booking process is in progress.
3. The application completes the transaction.
4. If the transaction is successful, the rows are unlocked and a message is shown to the user.
5. If the transaction fails due to locking of the rows by another user, a message is shown to the user to choose another time.



## UX and APIs Schema

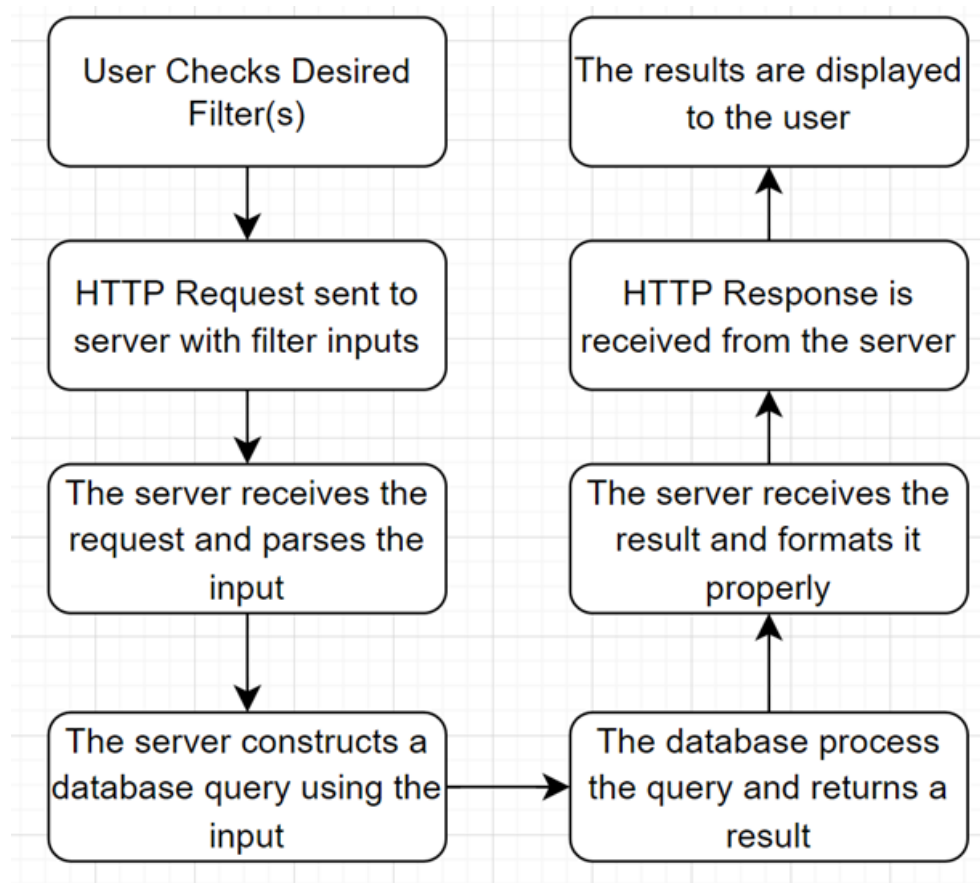


Any update or edit made in admin mode requires database access, as does car listing display. Integration within the APIs and database is necessary for test drive request forms, and image upload.

## APIs

- The Cloundinary API is used to enable image upload, because it can be integrated with MongoDB and Node.js.
- The SparkPost REST email API is used to implement the email responses sent to the users with the admin's test drive request decision.
- The Postman API is used to test frontend and backend integration. It enables checking whether the JSON requests to the server are working correctly.

## Flowchart



The above flow chart summarizes all typical operations, such as the filtering operation, done on the site. One difference would be the presence of an API. In this case, the flowchart would change. The server would send a request to the specific API which would return with a response that the server will handle.

## Errors and Failures

- The conditions that might lead to errors and failures are as follows:
- An error in accessing the database could cause the information displayed on the website to be inaccurate or incomplete. Car specs and details could be wrongly displayed.
- An error in accessing the database could restrict the admin from accessing admin mode.
- An error in updating database information could cause the database to lock and require intervention.
- An error within the Cloudinary API could render the images displayed incomplete, incorrect, or out-of-date.
- An error within the SparkPost API could mean that a user never receives a response to a test drive request, or even receives an incorrect approval or decline.
- Traffic above the estimated capacity could render the product non-functional.
- Database errors could render a page completely inaccessible and non-functional.

- i. An attacker could attempt to access admin mode.
- Certain errors could cascade and affect other functionalities:
  - An error in information update (case c) could result in inaccurate display (case a)
  - A particular scenario (in case g) could be the car details page being nonfunctional. This would inevitably make the test drive request form inaccessible as it can only be accessed through this page.

## Error Handling and Prevention

- Design for estimated capacity: handling a minimum of 20 car listings and 100 concurrent users is required to be acceptable.
- Design for performance according to the non-functional requirements.
- Implement isolation of components: an error in a component of the UI should not affect others, except for case (g) described above.
- Implement error handling within the code for all possible situations that might arise and affect the UI.
- Error messages: If a user tries to submit an incomplete form or invalid data, the website will display an error message with instructions to fix.
- Fallback options: If a user tries to access a page that doesn't exist, he'll get a blank page of the website with a "missing page" message and a button to go back to the main page.
- Network Errors: If the user's connection is too slow or there's a problem with the server, there will be a timeout and the website will display an error message.
- In the case of data loss or update error:
  - If there is an error in sending a test drive request, inform the user of the failure of the JSON POST request. It is for this reason that we have a success attribute in the test drive request table.
  - If there is an error in image upload, inform the admin, and enable them to retry.
  - If an uploaded image is corrupted, delete it, inform the admin, and prompt them to retry.
  - If there is an error in listing updates, display a message to the user that this information is not available at the time. This is related to the validity attribute in the listings table.

## Failure Recovery

The error handling is implemented in such a manner that UI and API errors will not result in a total failure. Only an inconsistent database could result in a crash. Thus, check the validity and success attributes and handle the error accordingly. Perform a cascaded delete of all the invalid information.

## Presentation Layer

Landing Page:


# Noureddine Auto

Email

Password

Sort by 

Sort by



NISSAN Qashqai 2018

12 200\$

Used


Test Drive

🕒 20 000 km.

📍 Beirut

⛽ Diesel 1.6

⚙️ mechanical



Volkswagen Passat 2020

14 200\$

Used


Test Drive

🕒 11 000 km.

📍 Beirut

⛽ Gas 1.6

⚙️ mechanical



Volkswagen Tiguan 2022


32 200\$

Used

Test Drive

🕒 11 000 km.

📍 Beirut



Nissan Juke

15 200\$

Used

Test Drive

🕒 20 000 km.

📍 Beirut

Filters 2

All Cars

Used

New

Type of vehicle

All types

Car brand

All brands

Car model

All models

Price from

Price to

Price from

Price to

Year from

Year to

Year from

Year to

Additional filters

Fuel

-

Mileage


-

Clear Filters

Apply Filters

Car Page:

## Noureddine Auto



NISSAN Qashqai 2018

12 200\$

Used

20 000 km.

Beirut

Diesel 1.6

mechanical

Buy

Test Drive

Other Specifications:


- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Aliquam tincidunt mauris eu risus.
- Vestibulum auctor dapibus neque.
- Nunc dignissim risus id metus.
- Cras ornare tristique elit.
- Vivamus vestibulum nulla nec ante.
- Praesent placerat risus quis eros.
- Fusce pellentesque suscipit.

Admin Page:

## Noureddine Auto

DashboardLog Out

Sort bySort byAdd Car



NISSAN Qashqai 2018

12 200\$

Used


20 000 km.

Beirut

Diesel 1.6

mechanical

Delete



Volkswagen Passat 2020

14 200\$

Used


11 000 km.

Beirut

Gas 1.6

mechanical

Delete



Volkswagen Tiguan 2022


32 200\$

Used

11 000 km.

Beirut

Delete



Nissan Juke

15 200\$

Used

20 000 km.

Beirut

Delete

Filters 2

All CarsUsedNew

Type of vehicle

All types

Car brand

All brands

Car model

All models

Price fromPrice to

Price fromPrice to

Year fromYear to

Year fromYear to

Additional filters

Fuel

-


Mileage

-

Clear FiltersApply Filters

Test Drive Form:

Select Test Drive Time

Select Date & Time 

Message

Submit

## Test Plan

The product will be subject to the following testing to ensure compliance with the product requirements, user needs, quality assessment and overall performance.

- Automated Load Tests: The product will be subject to a series of load tests of varying degrees to determine how user response time varies with different loads and if it fits the set performance standards. This will be done using JMeter.
- Alpha Tests: Members of the team will test the product to verify compliance with the requirements and completeness.
- Beta Tests: Real-time users will use the product and give feedback on possible defects or disturbances. Enhancements or optimization is to be done accordingly.
- Security Tests: Security tests will be performed using appropriate security test tools.
- Unit Tests: Each unit of the UI is to be tested independently to ensure its proper functioning.
  - Test the authentication page to ensure only authorized users can access it
  - Test the car listing page to ensure that all cars in the list are displayed correctly
  - Test the car details page to ensure that all the details are displayed accurately
  - Test the form submission process to ensure that it works correctly
  - Test the car update process to ensure that the updated information is displayed correctly

- **Integration Tests:** The Postman API will be used to test integration between the backend and the frontend.
  - Test the interaction between the authentication page and the car listing edit page to ensure that unauthorized users cannot view the page to edit the car listings
  - Test the interaction between the car listing page and the car details page to ensure that the correct details are displayed for each car
  - Test the interaction between the form submission process and the car update process to ensure that the updated information is correctly displayed after a user submits a form
- **Regression Tests:** After each sprint, Regression tests will be applied to look for any errors and avoid bug resurrection.
- Acunetix will be used to search for vulnerable spots and prevent SQL injections.
- BrowserStack will be used as a cross-browser testing app, to carry out automated tests

## Monitoring Plan

**Google Analytics:** This tool will be used to measure performance analytics and check compliance with the non-functional requirements, according to the metrics discussed later in this document.

The plan is to continuously monitor the performance and optimize according to the collected criteria. New versions will be designed according to the results of testing and analytics. This is further elaborated in the rollout plan.

**Logging plan:** Winston will be used to log errors in a specified database with regular checkups to delete older data as to avoid taking up useless space.

In case of website prolonged downtime, alert users via email.

## Rollout Plan

The product is to be available as a WebApp to be used on browsers such as Chrome, ensuring cross-browser compatibility but not mobile integration at this stage.

New versions are to be designed based on real-time user feedback and performance monitoring, after performing testing and monitoring.

After initial deployment, any new versions will be deployed in a ramped manner: instances of the old version are gradually replaced with the upgraded instances one at a time, and then the old version is deactivated. This enables performance monitoring of the new version. We will also use feature flags to selectively turn off the new version of the website without affecting other components or services.



Deployment will take place on Heroku which provides add-ons and automated scaling features.

We plan to complete all tests and communicate release plans to all stakeholders before full rollout. We will also provide release notes for each new version of the application, which will include a summary of new features, bug fixes, and any other changes. Users will be notified of any major changes through email or push notifications.

## Rollback Plan

The nature of the rollout plan facilitates the rollback plan. The deployment in a ramped manner enables us to gradually replace instances of the new version and test and monitor its performance. If the version is not functioning as intended, we could easily switch back to the former state, as it is deactivated but not deleted until we are certain the new version is fully operational as required. This gradual replacement allows us to easily manage, delete, and replace the files on the server. We will only maintain the latest version on the server prior to the new version that is being introduced, as it is updated and tested acceptably.

## Alternate Designs

- Include the test drive request form separately, outside the car listings page.
  - Pros: It is an isolated component and is still accessible if the car listings page goes down.
  - Cons: Users will have to manually browse through the car listings to select the exact model they want. It is simpler to link the request form in the car page to make the implementation more straightforward for the user.
- Implement communication within the app rather through the SparkPost email API.
  - Pros: Use in-house services rather than be dependent on a third-party API.
  - Cons: This greatly adds complexity to the app, and will probably force the implementation of socket programming to set up the communication, rather than have an API do the load of the work.
- Use an off-the-shelf e-commerce platform
  - Pros: Quick deployment, established platform, low cost
  - Cons: Limited customization, potential compatibility issues with existing systems, vendor lock-in
  - Reasons why it couldn't work: Existing systems may not integrate well with the platform, platform may not meet all business requirements
  - Inferior to proposed solution: Limited customization and potential compatibility issues may hinder future growth and scalability
  - Migration plan: Evaluate other off-the-shelf platforms or move to a custom solution if the selected platform cannot meet business requirements
- Develop a custom e-commerce solution in-house



- Pros: Complete control over functionality and customization, seamless integration with existing systems, potential for future scalability
- Cons: Longer development time, higher cost, potential for technical issues during development and implementation
- Reasons why it couldn't work: Lack of development resources, inability to meet project timeline and budget constraints
- Inferior to proposed solution: Higher cost and longer development time may delay time-to-market and hinder business goals
- Migration plan: Evaluate other custom development solutions or move to an off-the-shelf platform if in-house development is not feasible
- Use a hybrid solution combining off-the-shelf and custom components
  - Pros: Faster time-to-market than a completely custom solution, ability to customize specific components, potential for future scalability
  - Cons: Potential compatibility issues between off-the-shelf and custom components, higher cost than a completely off-the-shelf solution
  - Reasons why it couldn't work: Compatibility issues may prevent seamless integration of components, inability to find the right combination of off-the-shelf and custom components to meet business requirements
  - Inferior to proposed solution: Potential compatibility issues may hinder future scalability and integration of additional components
  - Migration plan: Evaluate other hybrid solutions or move to a completely custom or off-the-shelf solution if the selected hybrid solution cannot meet business requirements.

The proposed solution was selected as the best solution for the project because it balances the need for customization, scalability, and time-to-market. It allows for the necessary customization and seamless integration with existing systems, while also leveraging an established e-commerce platform to reduce development time and cost. In the event that the proposed solution falls through, the migration plan would be to evaluate the next best alternative solution and determine the best path forward.

## Further Considerations

### Impact

- This product will provide users with ease and accessibility. It will bring the full experience of a car dealership to the users' homes.
- For the admin, it will allow them to easily manage their listings and customers. They can update the car information, view test drive requests without having to manually manage the date as it will be maintained within the database and respond with a simple click of a button.

- On a broader scope, it will introduce car dealerships to the world of e-commerce, provide significant advertising, and move a step forward in the market. This will no doubt be competition for other dealerships which still stick to traditional methods, as statistics show that users opt to favor online shopping over tedious physical shopping.

### Third-party services and platforms considerations

- The third-party services used are the APIs, and the platform as a service (PaaS).
- All the third-party services used are trusted, reliable, secure and ease the process for us. The services are mostly open-source and free to use. In case of the need of scaling, we might need to upgrade some of the services to a premium version.
- Cloudinary is indispensable for image upload, as it rids of the complexity of using in-house services.
- PostMan, the API to be used for integration, is very important, as it allows us to conduct all the tests we need on the communication with the backend server. We can send specific JSON requests and view the responses.

### Cost analysis

- The cost to run the solution per day should be taken into account, including hosting fees, domain fees, and any third-party service fees.
- Given that the product is a web-app that utilizes APIs but is not highly complex, the deployment on Heroku as a product costs \$25 a month. In this first stage, it is being designed for a single car dealership selling 20 cars, and it is estimated that they will be sold within a year at most due to their small number. This brings the total cost up to \$300 in the first version.
- Were the product to be scaled to meet the future goals, it will no doubt increase in complexity due to the added features and functionalities, so the cost of deployment is expected to increase over a larger period of time.
- The usage of MongoDB and the other software is free and incur no additional financial costs. However, the number of users determines the size of the database and hence the storage costs.

### Security considerations

Threats	Mitigations
Malware and viruses	Keep hardware and software up-to-date, regular scans for malware, use of security plugins.

Denial of Service attacks	Requiring request verification and auto- scaling implementation.
SQL injections	Parametrized queries, Acunetix and checking for vulnerable spots.
Password attacks	Requiring strong passwords, two-factor authentication
Phishing attacks	Spam filters, monitor website for suspicious URLs, SSL encryption.
XSS (Cross-site scripting attacks)	Input validation, content security policy, HTTP-only cookies.
Data breaches	Secure coding practices, encryption, access control.
User generated content	Ban certain expressions and specify types of user inputs
Attacker's admin access	MongoDB protects data with preconfigured security features for authentication, authorization, encryption, and more

## Privacy considerations

- The product directly complies with standard regulations for data privacy, as the users are never required to input any personal information except for their email address, and this poses no risk. Furthermore, online purchasing is not an option so users will never be asked to give payment information.

## Regional considerations

- The solution will not be internationalized, due to different legal regulations between different countries, and due to the assumption that those who wish to use the product are local to the car dealership, as they will need to request test drives. Thus, we can assume that latency and availability are expected to be fairly high, especially considering the

small number of listings. Thus, we can say that data transfer will be simple across the local servers.

- From a legal perspective, there are no issues as it is simply a form of advertising, so there are no commerce laws that interfere with the product.

## Accessibility Considerations

- To cater to those with potential sight disabilities, implement a voiceover button to read the text in every page in the UI.
- This implementation will be evaluated using Beta tests to get feedback from real-life users and improve on the solution.

## Risks

- Security risks: The solution may be vulnerable to cyber attacks or data breaches, which could compromise user data and lead to reputational damage or legal liabilities.
- Compliance risks: There may be legal or regulatory requirements that need to be met in order to launch the solution, such as data privacy or accessibility standards.
- Resource risks: There may be concerns around the amount of resources (time, money, personnel) required to develop, launch, and maintain the solution.
- Technical risks: There may be technical challenges in implementing the solution, such as compatibility issues with existing systems or unforeseen bugs that could cause downtime or errors.

## Operational considerations

- The solution should be designed to minimize adverse aftereffects, such as downtime, data loss, or errors.
- The solution should be designed to recover in case of a failure and minimize downtime. This plan should include procedures for diagnosing and resolving issues, as well as contingency plans in case of severe system failures.
- Operational costs should be kept low by optimizing the use of cloud resources and minimizing unnecessary spending.

## Support considerations

- Customer support is not a feature in this version.
- The solution is designed to provide sufficient information to answer the user's potential inquiries with minimal need for support.

- We can provide a help page that includes a walkthrough tutorial for all the website's features, especially for non-tech-savvy people.
- The admin will present his contact information in case of important inquiries.

## Success Evaluation

### Impact

Security is of the utmost importance and is estimated to have significant impact in the deployment of this product. It is the main risk with the development of this application. The success of this product depends primarily on the security of the database and protection of admin functionalities.

Performance impact is vital if this product is intended to scale to larger businesses and attract a larger number of users by reducing load and response times. The aim is to provide users with a smooth experience, and this will be evaluated using Beta tests.

Cost impact is not expected to be significant, given that the deployment is measured to be the only aspect that incurs cost at this stage. The cost is expected to increase as the product is scaled, but this comes with the attraction of more businesses, and thus will be leveled out if not profitable.

The evaluation of the impact on other components and services that the dealership may be using is crucial as to guarantee that the project does not affect their functionalities.

### Metrics

This product will be evaluated according to the following metrics:

- Latency: User requests must be serviced within the acceptable range of [50-100 milliseconds], with the optimal range being < 50 milliseconds.
- Scalability: This product is meant for use for one car dealership listing exactly 20 cars; it must be able to service 100 concurrent users to be acceptable. However, this number must increase proportionally if the number of car listings or users were to increase.
- Availability: The software must maintain a high level of availability under normal circumstances, exceeding 95% for baseline traffic and 90% for high traffic.
- Reliability: The product must maintain a low failure rate.
- Fault tolerance: The product must not crash should one of its functionalities stop operating as intended; there must be isolation between the different components.
- Compliance: All functional requirements must meet the acceptance criteria.
- Security: Security tests with appropriate tools are to be performed to defend against potential threats and attacks.

- Cost: Development, maintenance, and hosting.