REPORT TWO
FINAL

GROUP 3

Brett J. Davis
Thomas Kerley
Jasmine Irvin
Joshua Puetz

CONTRIBUTIONS

Brett J. Davis 25%
Thomas Kerley 25%
Jasmine Irvin 25%
Joshua Puetz: 25%

Table of Contents

1 PART 1

1.1 Analysis and Domain Modeling

1.1.1 Conceptual Model

1.1.1.1 Concept Definitions

Responsibility descriptions

Use Case	Responsibility Description	Type	Concept name
UC-1	coordinate actions between concepts	D	controller
UC-1	request for lot status	K	statusRequest
UC-1	Web page shows current context, actions, and history of prior actions	K	interfacePage
UC-1	render the retrieved records into an HTML	D	pageMaker
UC-1	Coordinate form specifying the new vehicle and location and returns	K	updateController
UC-1	queries and forwards database requests and data	D	databaseConnection
UC-2	coordinate actions between concepts	D	controller
UC-2	form specifying the search parameters	K	searchRequest
UC-2	render the retrieved records into an HTML	D	pageMaker
UC-2	Web page shows current contexts	K	interfacePage
UC-2	notify technician of vehicle request	D	notifier
UC-2	Wait 5 min	K	timer
UC-2	Sensor check for space vacancy	D	sensorChecker
UC-2	queries and forwards database requests and data	D	databaseConnection
UC-3	coordinate actions between concepts	D	controller
UC-3	form specifying the search parameters	K	searchRequest
UC-3	render the retrieved records into an HTML	D	pageMaker
UC-3	HTML document that shows the vehicles location	K	interfacePage
UC-3	Wait 5 min	K	timer
UC-3	Sensor space vacancy check	D	sensorChecker
UC-3	queries and forwards database requests and data	D	databaseConnection
UC-4	coordinate actions between concepts	D	controller
UC-4	form specifying the search parameters	K	searchRequest
UC-4	render the retrieved records into an HTML	D	pageMaker
UC-4	HTML shows vehicles listed	K	interfacePage
UC-4	form specifying the reservation, update database with reservation	K	reservationController
UC-4	notify salesperson of reservation	D	notifier
UC-4	queries and forwards database requests and data	D	databaseConnection
UC-5	coordinate actions between concepts	D	controller
UC-5	Sensor pushes notification of change	D	sensorChecker
UC-5	Conveys database updates time sensor notification, lot info, and vehicle info, who was notified, etc.	D	updateController
UC-5	queries and forwards database requests and data	D	databaseConnection
UC-5	database notifies manager	D	notifier

1.1.1.2 Association Definitions

Associations

Use Case	Concept Pair	Association Description	Association name
UC-1	Controller and pageMaker	Controller passes requests to pageMaker and receives back pages prepared for display	conveys requests
	pagemaker and interfacePage	pageMaker prepares the data for the web page	prepares
	controller and interfacePage	Controller posts page	posts
	controller and statusRequest	Controller receives and sanitizes requests for lot status	receives
	updateController and databaseConnection	Controller passes sanitized requests to the database	conveys requests
	controller and updateController	Controller receives and forwards update instruction & data	receives
UC-2	Controller and pageMaker	Controller passes requests to pageMaker and receives back pages prepared for display	conveys requests
	pagemaker and interfacePage	pageMaker prepares the data for the web page	prepares
	controller and interfacePage	Controller posts page	posts
	controller and searchRequest	Controller receives and sanitizes requests for vehicle location	receives
	controller and databaseConnection	sends request for vehicle location or sensor update	conveys requests
	Controller and notifier	Controller sends a notice to a technician	notifies
	controller and timer	controller activates a timer	activates
	sensorChecker and timer	timer triggers a sensor check, if still occupied, restart timer	activates
	sensorChecker and controller	sends an update to be forwarded to the database	updates
UC-3	Controller and pageMaker	Controller passes requests to pageMaker and receives back pages prepared for display	conveys requests
	pagemaker and interfacePage	pageMaker prepares the data for the web page	prepares
	controller and interfacePage	Controller posts page	posts
	controller and searchRequest	Controller receives and sanitizes requests for vehicle location	receives
	controller and databaseConnection	sends request for vehicle location or sensor update	conveys requests
	controller and timer	controller activates a timer	starts timer
	sensorChecker and timer	timer triggers a sensor check, if still occupied, restart timer	updates
	sensorChecker and controller	sends an update to be forwarded to the database	updates

Use Case	Concept Pair	Association Description	Association name
UC-4	Controller and pageMaker	Controller passes requests to pageMaker and receives back pages prepared for display	conveys requests
	pagemaker and interfacePage	pageMaker prepares the data for the web page	prepares
	controller and interfacePage	Controller posts page	posts
	controller and searchRequest	Controller receives and sanitizes requests for vehicle location	receives
	controller and databaseConnection	sends request for vehicle location or sensor update	conveys requests
	Controller and notifier	Controller sends a notice to a salesperson	notifies
	Controller and reservationController	passes reservation data from the web page via the controller to the database	conveys requests
UC-5	Controller and notifier	Sends notification to tech/manager of non-requested change	notifies
	controller and updateController	Controller receives update instruction & data	conveys requests
	sensorChecker and controller	sends an update to be forwarded to the database	updates
	updateController and databaseConnection	the controller checks and sends an update for existing status and updates. It should also check against false positives	receives
	databaseConnection and Controller	Sends notification confirmation	confirms

1.1.1.3 Attribute Definitions

Attributes			
Use Case	**Concept**	**Attributes**	**Attribute Description**
UC-1	statusRequest	Users's Id	who is requesting data
		lotSearchParameters	if multiple lots, which lots are in question
	updateController	Users's Id	who is requesting data
		updated parameters	What information about a vehicle is changing
	databaseConnection	queryData	a formatted data package containing search terms, ids, etc. that needs to be updated/requested
		returnData	a formatted data package to be returned
UC-2	searchRequest	Users's Id	who is requesting data
		search Parameters	if multiple lots, which lots are in question
	notifier	contactInfo	Contact information of party to be notified
	timer	duration	how long the timer is
		snoozeCount	how many times the timer reset for a single request
	sensorChecker	sensorStatus	data on if a space is detected as empty or not.
	databaseConnection	queryData	a formatted data package containing search terms, ids, etc. that needs to be updated/requested
		returnData	a formatted data package to be returned
UC-3	searchRequest	Users's Id	who is requesting data
		search Parameters	if multiple lots, which lots are in question
	timer	duration	how long the timer is set for/remaining
		snoozeCount	how many times the timer reset for a single request
	sensorChecker	sensorStatus	data on if a space is detected as empty or not.
	databaseConnection	queryData	a formatted data package containing search terms, ids, etc. that needs to be updated/requested
		returnData	a formatted data package to be returned
UC-4	searchRequest	Users's Id	who is requesting data
		search Parameters	if multiple lots, which lots are in question
	Reservation Controller	Users's Id	who is requesting data
		vehicle	which vehicle is reserved
		dateTime	time of reservation
	notifier	contactInfo	Contact information of party to be notified
	databaseConnection	queryData	a formatted data package containing search terms, ids, etc. that needs to be updated/requested
		returnData	a formatted data package to be returned
UC-5	sensorChecker	sensorStatus	data on if a space is detected as empty or not.
	updateController	dateTime	information on when alert occurred
		sensorInfo	Which space is the triggered sensor reporting
		vehicle	supposed missing vehicle
	databaseConnection	updateData	data to be updated
	notifier	contactInfo	Contact information of party to be notified

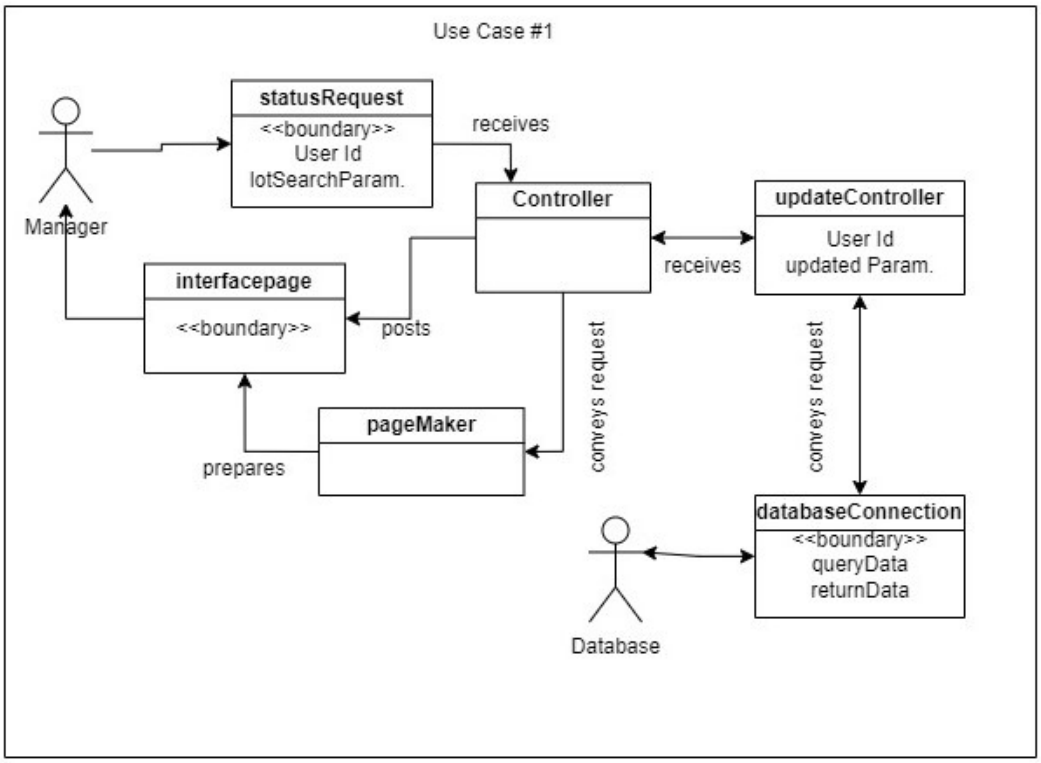
1.1.1.4 Traceability Matrix

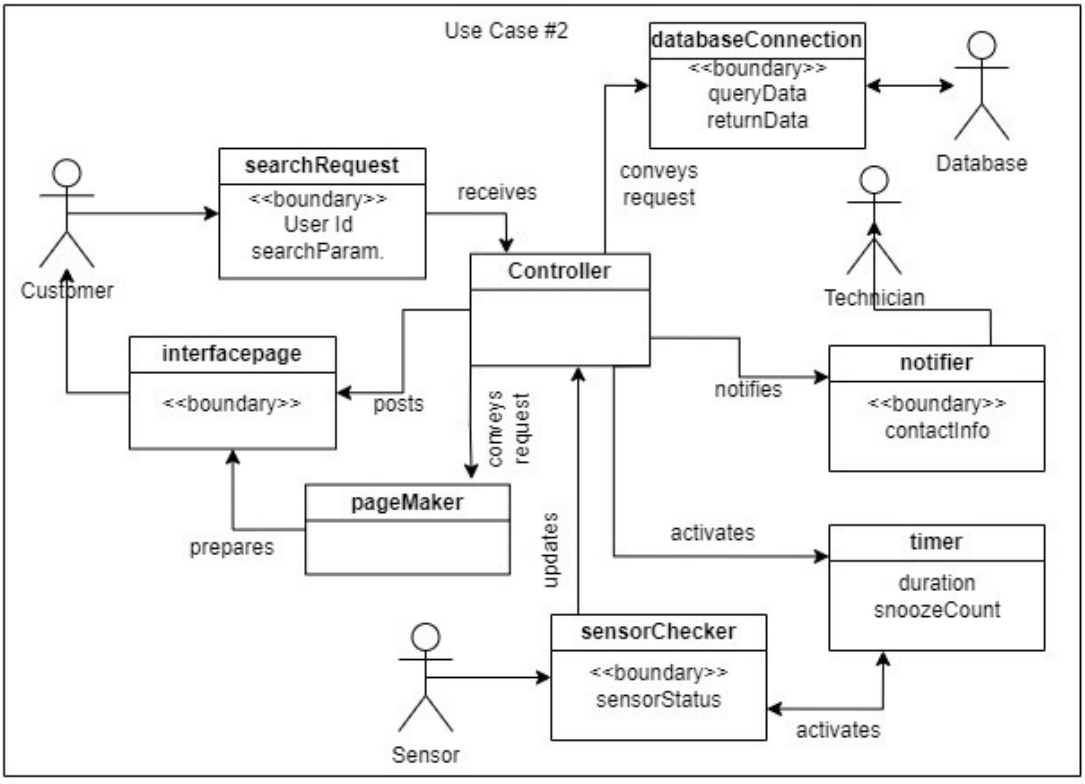
Domain Traceability Matrix												
		Domain Concepts										
Use Case	PW	controller	statusRequest	interfacePage	pageMaker	updateController	databaseConnection	searchRequest	notifier	timer	sensorChecker	ReservationController
UC1	18	X	X	X	X	X	X					
UC2	10	X			X		X	X	X	X	X	
UC3	15	X		X	X		X	X		X	X	
UC4	5	X		X	X		X	X	X			X
UC5	20	X				X	X		X		X	

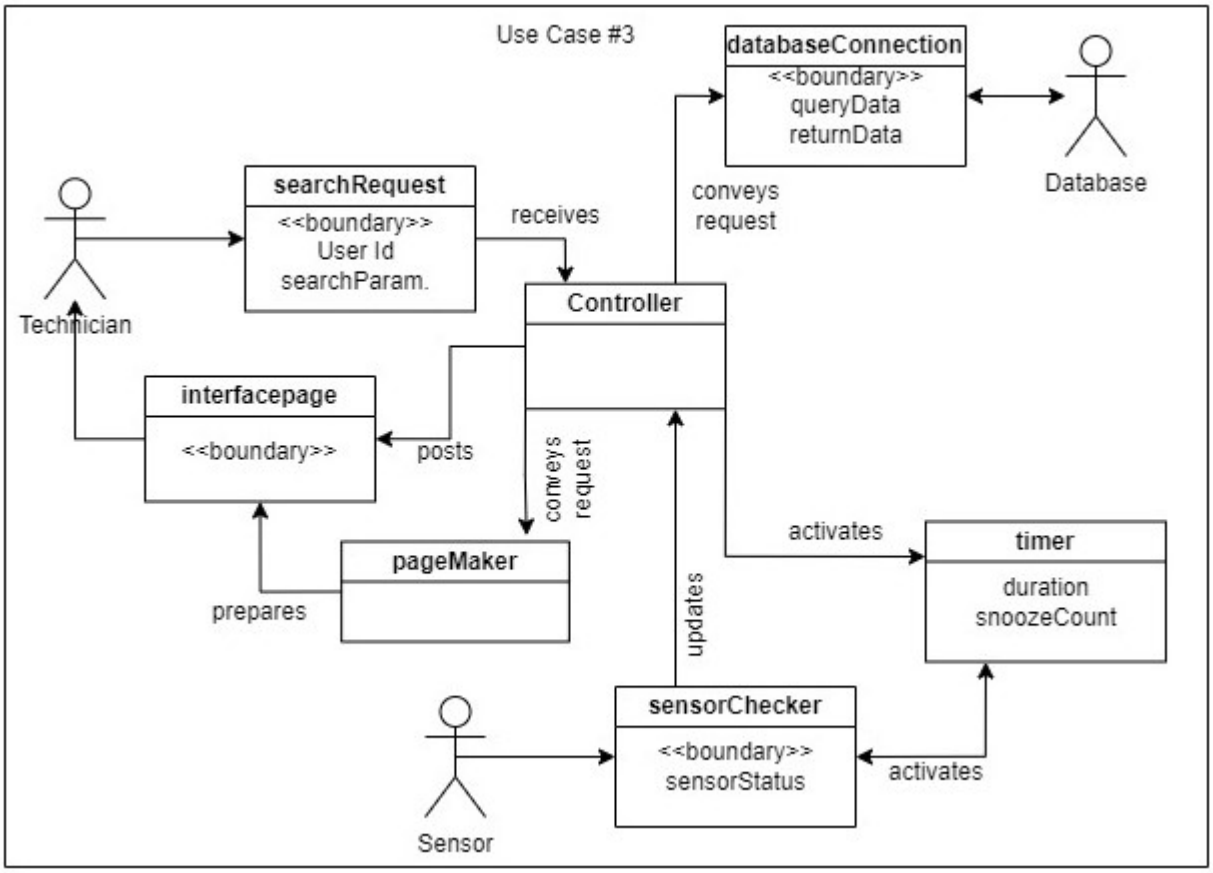
1.1.1.5 Domain Model

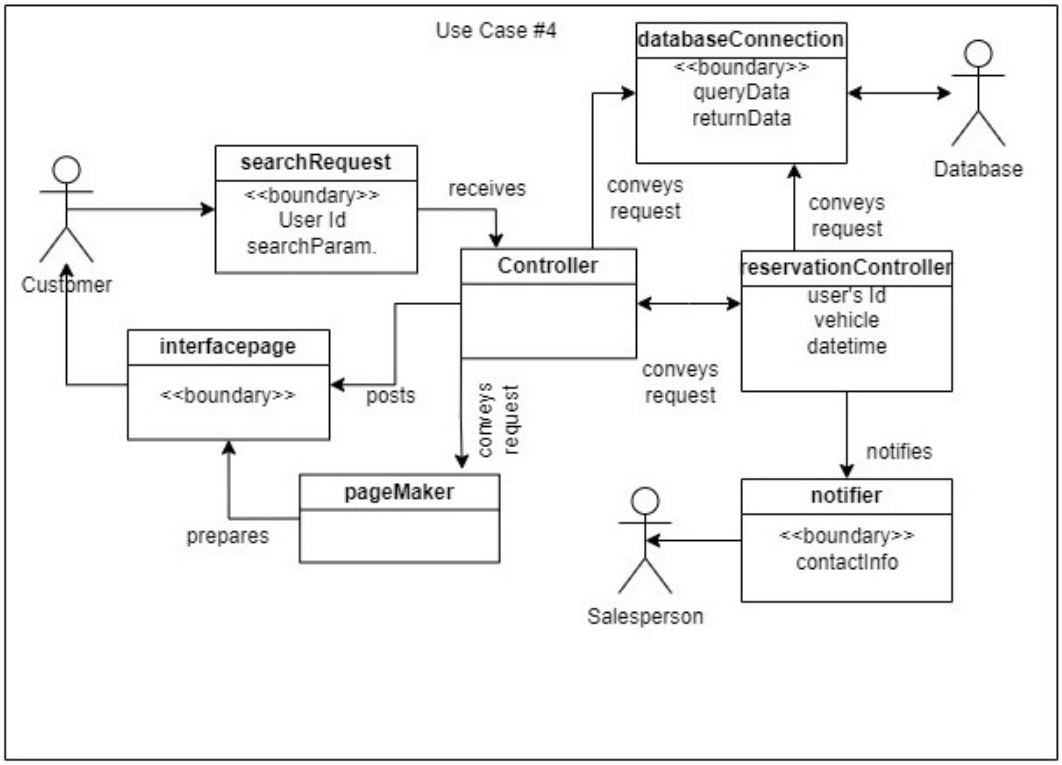
Each use case has a form of user interface that allows a customer, technician, salesperson, or manager to perform creation, reading, updating, and deletion of vehicles within the inventory system; with the exception of use case number five, which we will discuss at the end. Each use case involves the database to track what changed, who changed what. This will allow for future additions of adding in sales data collection. The use cases one thru four initiate via a user; a manager, technician, customer, or salesperson. They will interact with the system via a web page that will take in their requests, such as a search request, and pass it to the controller to convey to the appropriate service. In most cases, this will involve communicating with a database for the required data and then a page maker to translate and return the data to the website. A website versus a specific software gui allows for easy deployment to most computers and existing systems along. In use cases two and three, they alert a different actor, namely a technician, to bring the car to the requested spot. This can be set-up to a specific IP address or device identifier at the customer's site. Use cases two and three have timers to check if the car has been moved per a request and uses a timer to recheck the sensor's status to know if there needs to be a re-alert to the technician or salesperson. Use case four has a unique element of a reservation. The reservation is initiated by the customer via the dealership's website.

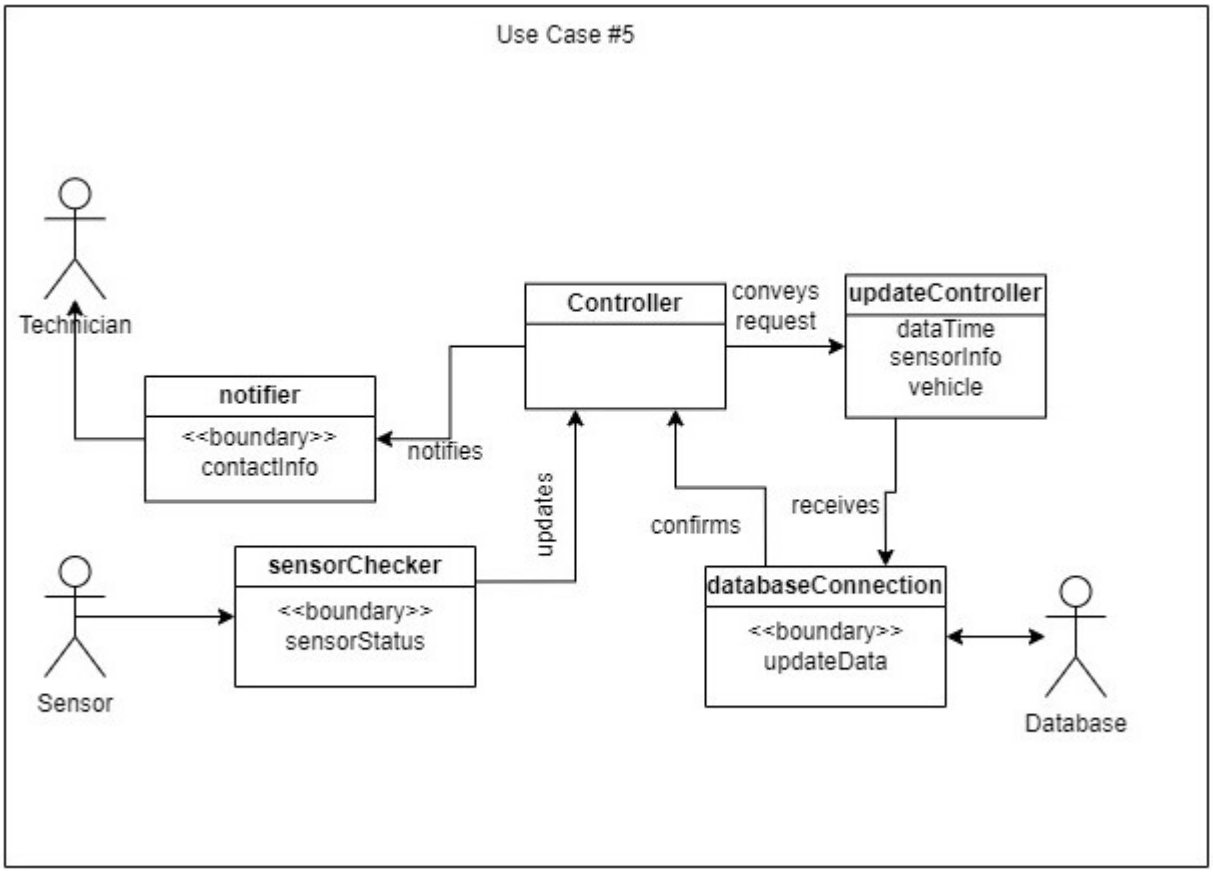
Use case number five is the most unique of them by being the one that the initiator is the sensor. If a sensor is tripped, it will begin a check to the data base to see if the car was cleared to be relocated. Once confirmed via a database check, the controller will send a notice to a technician or salesperson that an unauthorized move has occurred. The technician can do a visual check per the dealership's policy to confirm the car has gone missing. The database will be updated of the missing car, a sensor's ID, and the time of the sensor alert.


Use Case #1
statusRequest
<<boundary>>
User Id
lotSearchParam.
receives
Manager
Controller
updateController
User Id
updated Param.
receives
interfacepage
<<boundary>>
posts
conveys request
conveys request
pageMaker
prepares
databaseConnection
<<boundary>>
queryData
returnData
Database


Use Case #2
databaseConnection
<<boundary>>
queryData
returnData
Database
conveys request
searchRequest
<<boundary>>
User Id
searchParam.
receives
Customer
Technician
Controller
interfacepage
<<boundary>>
posts
notifier
<<boundary>>
contactInfo
notifies
conveys request
pageMaker
prepares
updates
activates
timer
duration
snoozeCount
sensorChecker
<<boundary>>
sensorStatus
activates
Sensor


Use Case #3
databaseConnection
<<boundary>>
queryData
returnData
Database
searchRequest
<<boundary>>
User Id
searchParam.
receives
conveys
request
Technician
Controller
interfacepage
<<boundary>>
posts
conveys
request
activates
timer
duration
snoozeCount
pageMaker
prepares
updates
sensorChecker
<<boundary>>
sensorStatus
activates
Sensor


Use Case #4
databaseConnection
<<boundary>>
queryData
returnData
Database
searchRequest
<<boundary>>
User Id
searchParam.
receives
conveys
request
conveys
request
Customer
Controller
reservationController
user's Id
vehicle
datetime
interfacepage
<<boundary>>
posts
conveys
request
conveys
request
notifies
pageMaker
prepares
notifier
<<boundary>>
contactInfo
Salesperson


Use Case #5
Technician
Controller
conveys
request
updateController
dataTime
sensorInfo
vehicle
notifier
<<boundary>>
contactInfo
notifies
updates
confirms
receives
sensorChecker
<<boundary>>
sensorStatus
databaseConnection
<<boundary>>
updateData
Sensor
Database

1.1.2 System Operation Contracts

Attribute Contracts			
Attribute	**Use Cases**	**Precondition**	**Post condition**
Users's Id	1,2,3,4	User id is an alphanumeric string smaller than 64 characters	
LotSearch Parameters	1	lot numbers searched for existing in DB and are integers	
updating parameters	1	fields are not empty. And match variable types, e.g. dates are valid dates	
queryData	1,2,3,4,5	fields are not empty. And match variable types, e.g. dates are valid dates	
returnData	1,2,3,4,5	fields are not empty. And match variable types, e.g. dates are valid dates	
search Parameters	2,3,4	fields entered match variable types. Lengths are reasonable. No special characters	
contactInfo	2,4,5	valid string name, string phone, string email	
duration	2,3	start time is non-zero or a non-zero default	Duration = 0, add to snoozeCount, renotify/check, then reset
Snooze Count	2,3	SnoozeCount = 0, variable is of type Int.	SnoozeCount >= 0; if sensor clears, snooze count resets to zero, else snoozeCount++
Senosr Status	2,3,5	boolean value	
dateTime	5	dateTime variable or equivalent	
sensorInfo	5	unique id of sensor, this UID denotes which parking space is which.	
vehicle	5	fields are not empty and match types	

1.1.3 Data Model and Persistent Storage

The first step is to determine if the system requires data storage beyond a single execution. Based on the information, it can be inferred that the system does require persistent data storage. This means that data needs to be saved and accessed across multiple system executions. Next, we need to identify the persistent objects that need to be stored in the system. From the system architecture description, the following objects can be considered as potential candidates for persistent storage:

- Cars: Information about the cars in the dealership, including attributes such as make, model, year, and availability.
- Spaces: Details about the parking spaces in the dealership, including their status (occupied or vacant) and any associated attributes.
- Users: Data related to authenticated users, including their roles and credentials.

- Sales Data: Information about sales made, including details about the sold cars, customers, and transaction history.

These objects are subject to further analysis and refinement based on the specific requirements of the system.

Considering the requirements for persistent data storage and the nature of the data objects, a relational database is a suitable storage management strategy. A relational database provides a structured and efficient way to store and retrieve data, ensuring data integrity and enabling complex queries and relationships between different entities. To design the database schema, further analysis is required to determine the specific attributes and relationships for each persistent object. Based on this analysis, a detailed database schema can be created using SQL (Structured Query Language).

The database schema will define the tables, columns, data types, constraints, and relationships between different tables. It will serve as the blueprint for creating and managing the database. In conclusion, the system requires persistent data storage, and a relational database is a suitable storage management strategy. The identified persistent objects, such as cars, spaces, users, and sales data, can be stored and managed effectively using a well-designed database schema implemented with SQL.

A basic schema/database example is linked below:

1. Cars Table:
    - car_id (Primary Key)
    - make
    - model
    - year
    - availability
    - (additional attributes specific to cars)

2. Spaces Table:
    - space_id (Primary Key)
    - status
    - (additional attributes specific to spaces)

3. Users Table:
    - user_id (Primary Key)
    - username
    - password
    - role
    - (additional attributes specific to users)

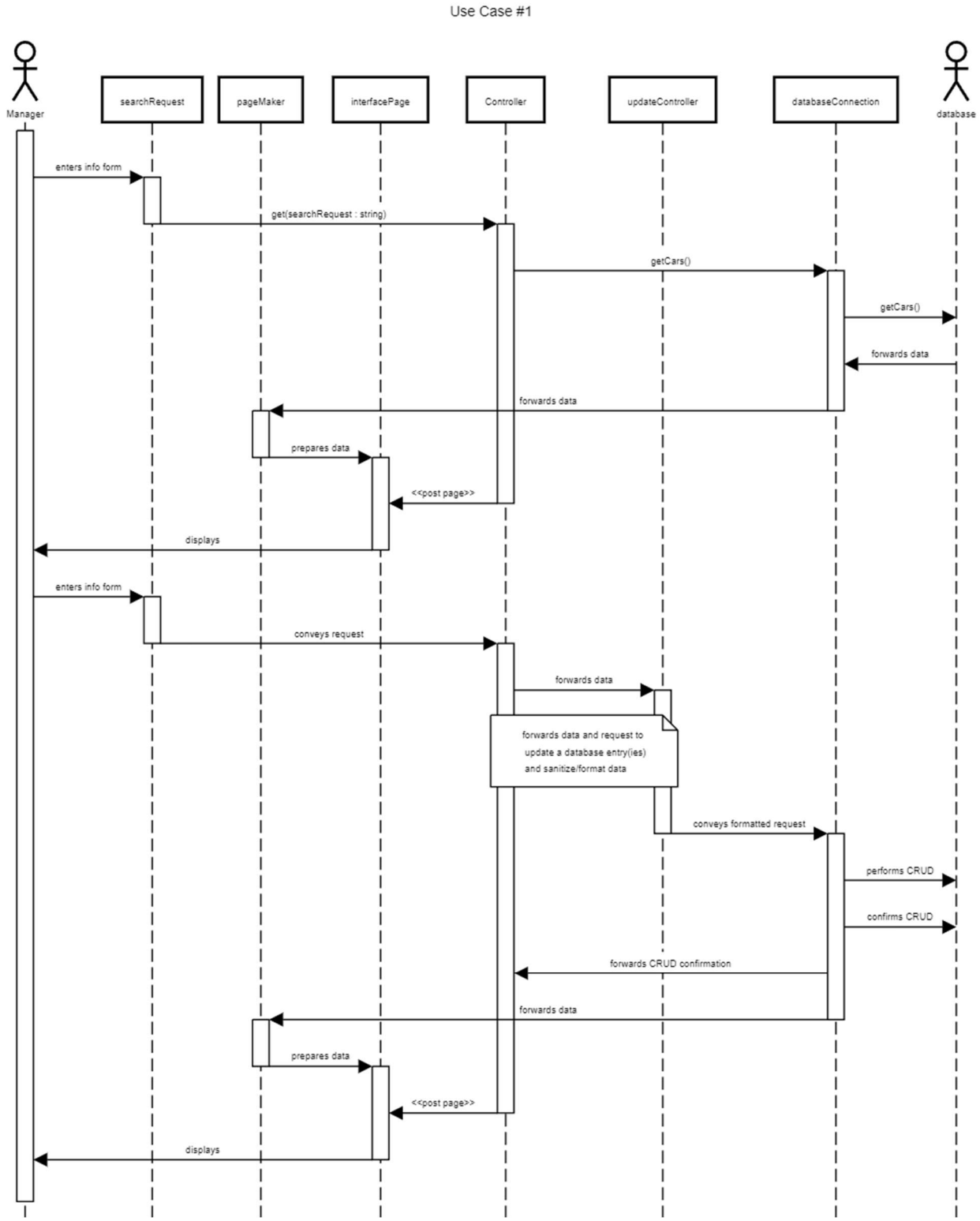
4. Sales Table:
    - sale_id (Primary Key)
    - car_id (Foreign Key referencing the Cars Table)
    - customer_id (Foreign Key referencing the Users Table)
    - transaction_date
    - (additional attributes specific to sales)

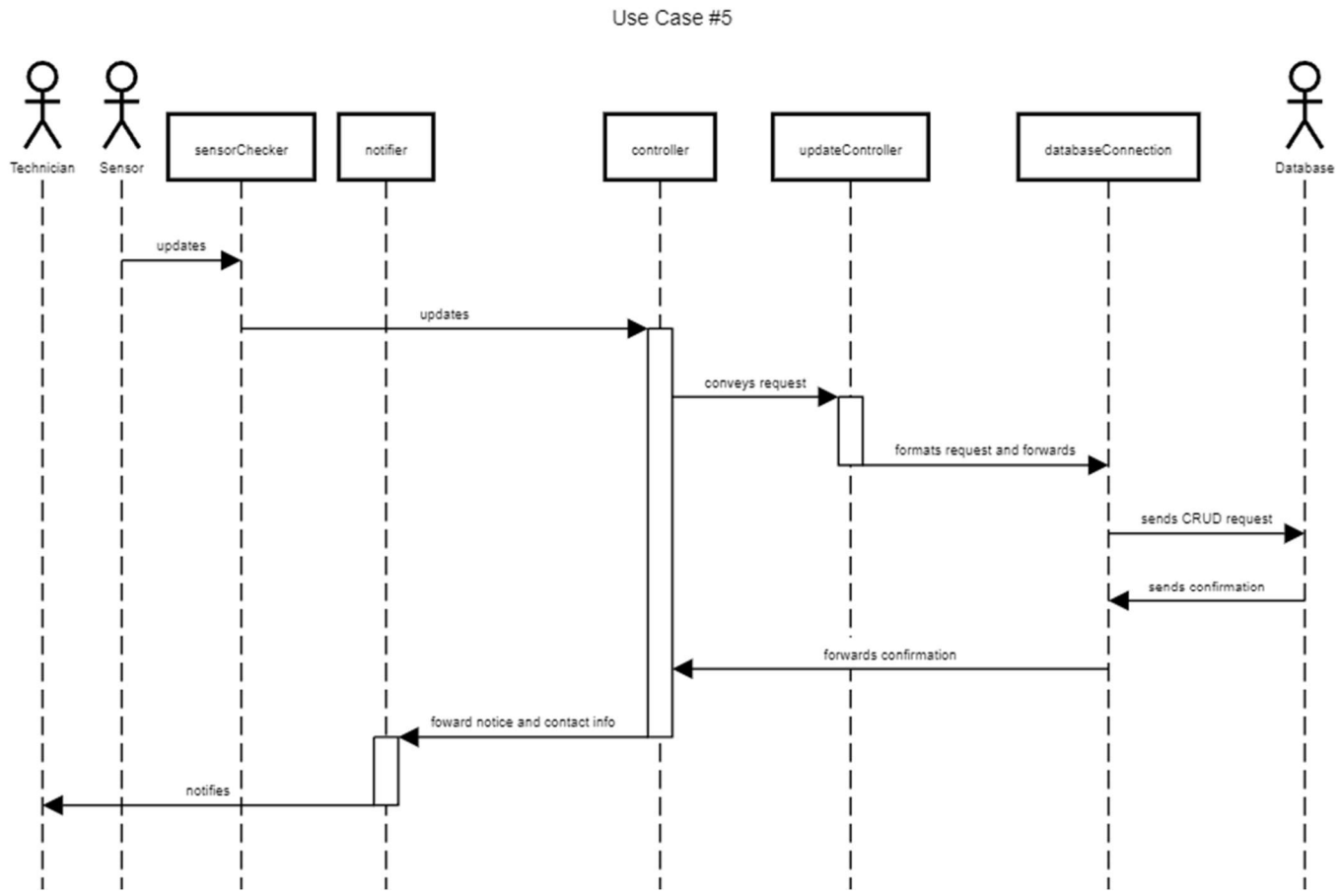
1.1.4 Mathematical Model

None.

2 PART 2

2.1 Interaction Diagrams

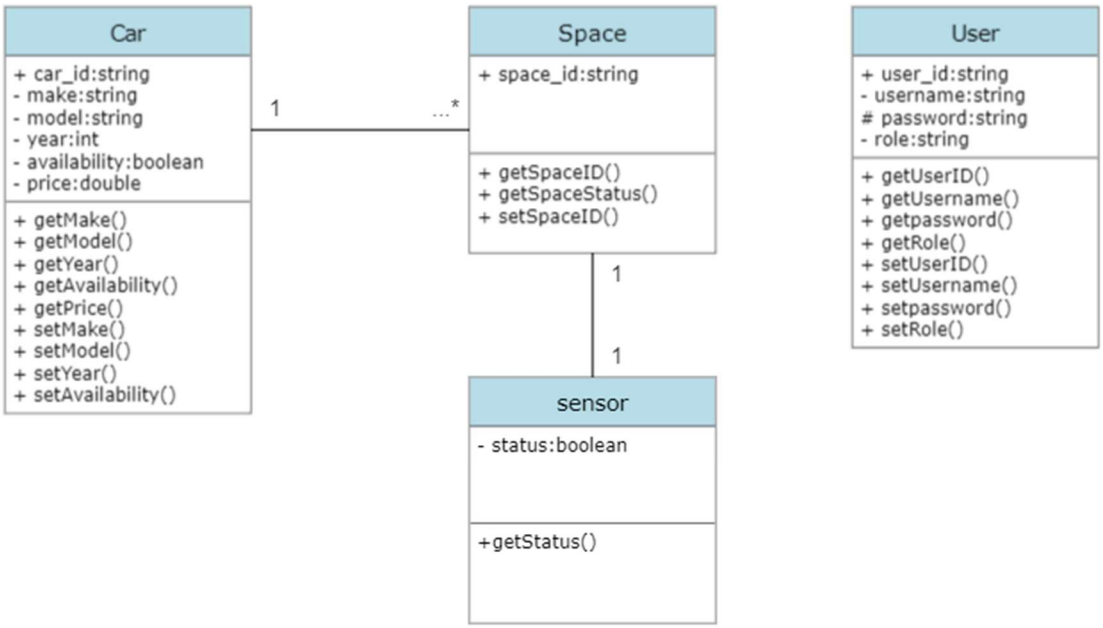




Starting with the Controller, we agreed it should follow the expert doer principle to interact between the different components. This might cause low cohesion or high coupling as the controller passes all the requests. However, by using pageMaker and interfacePage, which meet the low coupling and high cohesion principles, we can make the page feel responsive, improving the user's experience and reduce waiting times after polling the system to find a car. The updateController, searchRequest, and databaseConnection have both high cohesion and low coupling and are primarily to reduce the controller responsibilities regarding these principles.

2.2 Class Diagram and Interface Specification

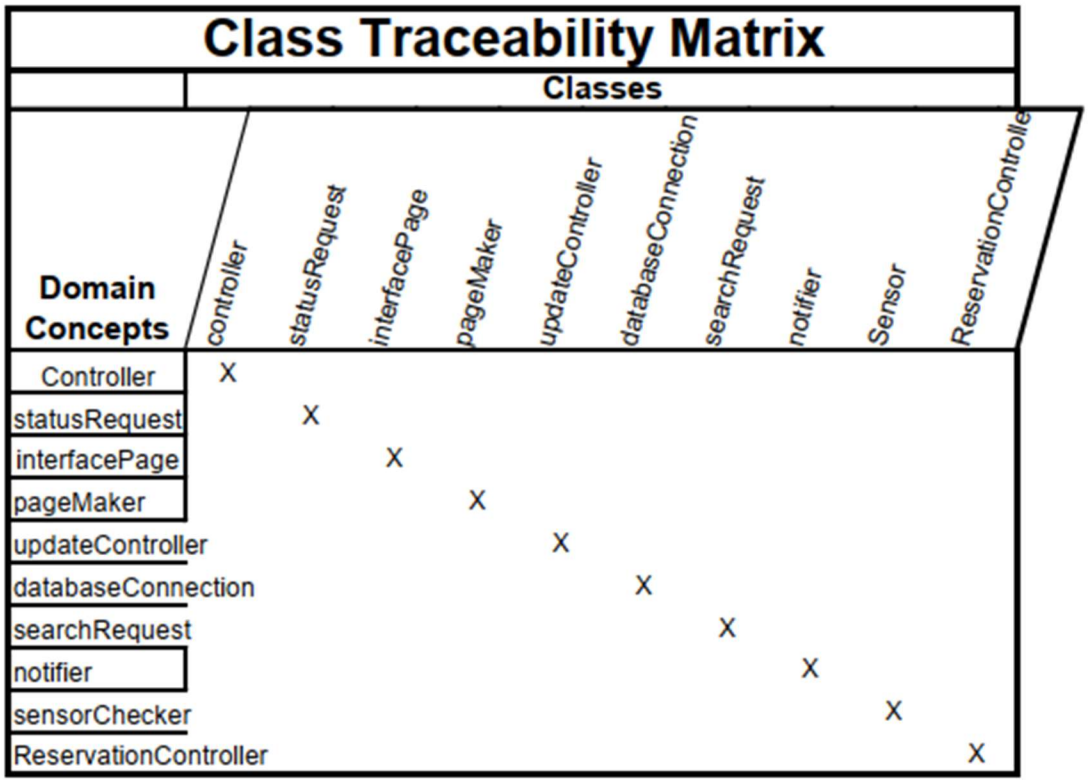
2.2.1 Class Diagram



2.2.2 Data Types and Operation Signatures

We have three classes, Cars, Users, and Space. The system can later incorporate sales data upon stage one completion. The cars class contains a unique identifier, car_id, and stores various information about the car such as make and model. This class can be expanded to export data to a future sales class. The space class is the second simplest, it requires a unique space_id attribute and is populated by a sensor class. Such sensor class only reports the status of a sensor, a Boolean value, and a method to forward the status upon change or request. The user class is the most complicated and the one that will expand as more elements of our system come online. It contains a unique id attribute, requires an encrypted password attribute, a username, and the appropriate getter and setter methods. The key thing of this class is the role attribute, which will line up with user permissions on our app. For example, we will need to distinguish between a technician and a customer. As we develop more of this system, specifically with the sales integration, we will need to add more user information such as email, contact information, etc.

2.2.3 Traceability Matrix



3 PART 3

3.1 Algorithms and Data Structures

Not applicable.

3.2 User Interface Design and Implementation

We have not made significant changes to the initial screen mock-ups developed for Report #1. But, we are currently investigating whether it is possible to have an interactive image that depicts a map of the parking lot. When a user queries the location of a vehicle, the interactive image will highlight the location of the vehicle. Suitable choices appear to include HTML image mapping. We believe that this will increase the "ease-of-use" of the user interface by allowing the user to instantly "see" the location of the vehicle.

3.3 Design of Tests

The testing of our software will be comprehensive and cover various aspects of the system. For unit testing, we will design and implement specific test cases to verify the correctness and functionality of individual units, such as functions and methods. These test cases will cover critical functionalities, edge cases, and possible scenarios to ensure robustness.

We will also focus on achieving high test coverage, aiming to cover statements, branches, and paths within the codebase. Integration testing will be conducted to validate the interaction and integration between different components or modules of the software. We will identify integration points and define test scenarios that cover the communication and data flow between these components.

The Laravel framework provides built-in testing capabilities, which we will leverage to facilitate both unit and integration testing. Additionally, we will conduct extensive testing of the user interface to ensure usability, responsiveness, and adherence to user interface requirements. Throughout the testing process, we will closely monitor and measure test coverage to ensure that critical parts of the software are thoroughly tested.

4 PROJECT MANAGEMENT and PLAN OF WORK

4.1 Merging the Contributions from Individual Team Members

Due to the short time frame, issues encountered while compiling everyone's work included dividing up the work evenly and ensuring that everyone was aware of their assigned work. To overcome this issue, the Group holds weekly telephone conferences at a set time. In addition, the Group uses texting, emailing, and video conferencing to communicate.

4.2 Project Coordination and Progress Report

So far, no use case has been fully implemented. Use case #1 is currently being tackled. The major project activities include establishing the framework of the website, setting up the database, and working on the classes. See following Gnatt charts for more detailed information.

4.3 Plan of Work

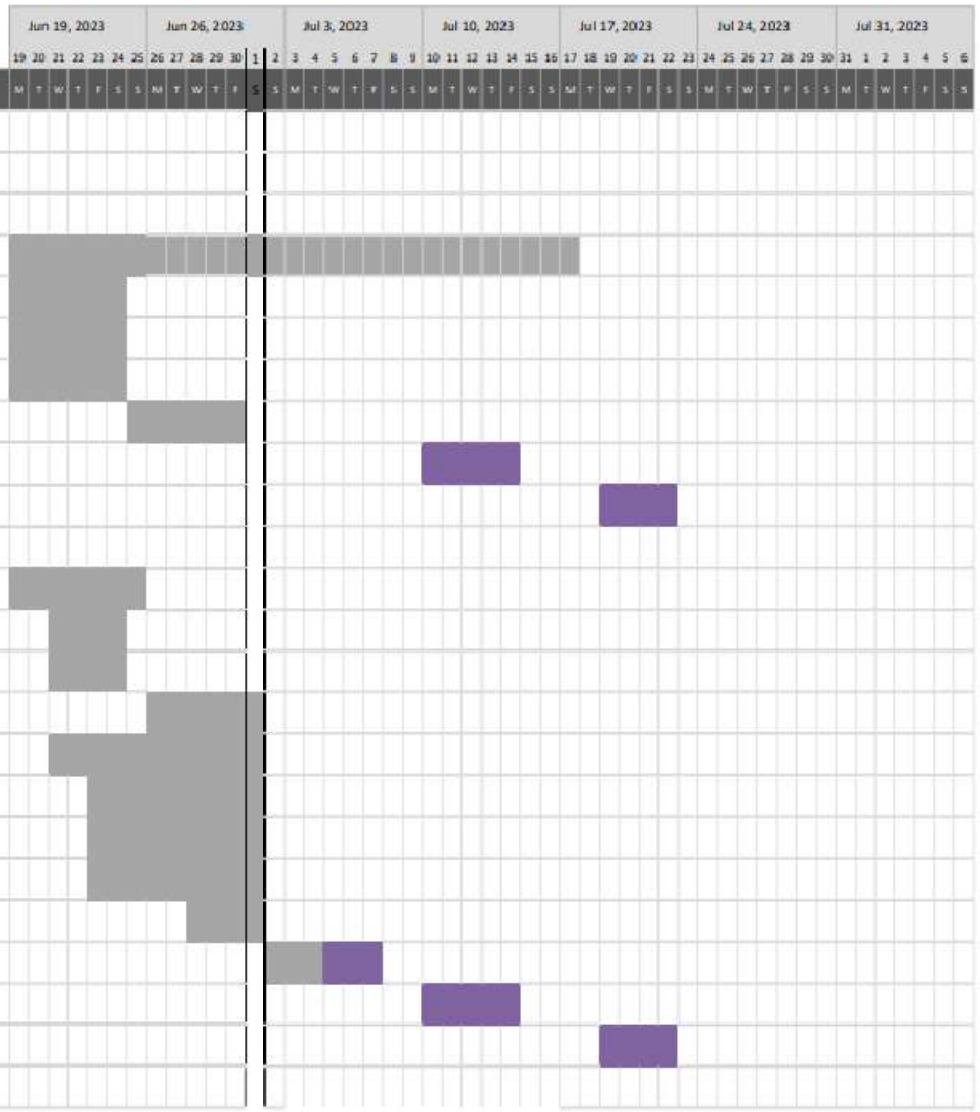
(see following Gnatt charts)

Lot Management

Group 3
Summer 2023

CSCI 441 Software Eng.

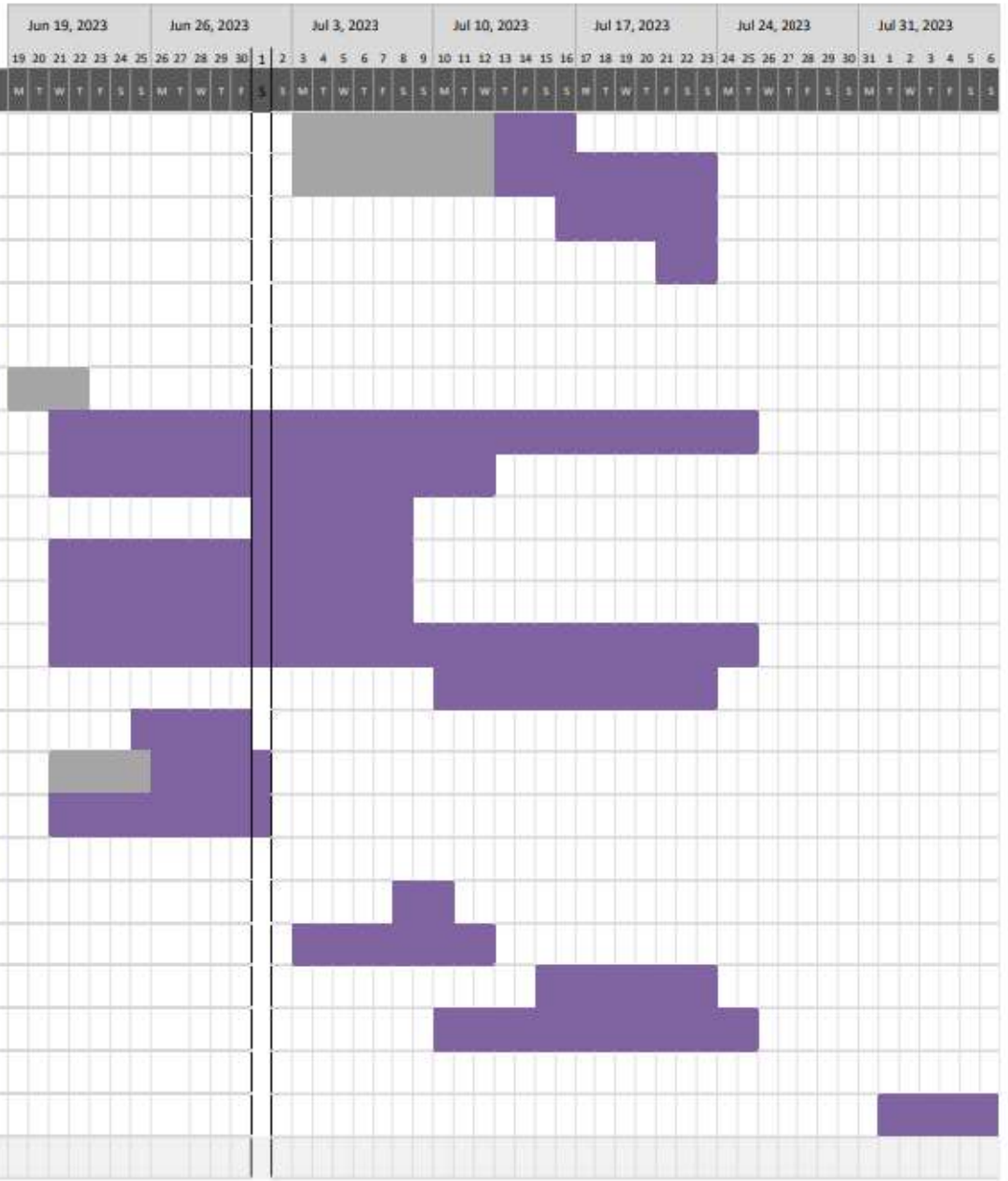
Project Start: Mon, 6/12/2023
Display Week: 2



TASK	ASSIGNED TO	PROGRESS	START	END
Report One				
Part One	Jasmine & Thomas	100%	6/12/23	6/18/23
Part Two	Brett & Thomas	100%	6/15/23	6/18/23
Part Three	Team	300%	6/15/23	6/25/23
P3 A-D	Joshua	100%	6/19/23	6/24/23
P3 E, f	Brett	100%	6/19/23	6/24/23
P3 #6, 7	Thomas	100%	6/19/23	6/24/23
Use Case Points review	Team	100%	6/25/23	6/30/23
Update Check (Appendix Reference)		0%	7/10/23	7/14/23
Update Check		0%	7/19/23	7/22/23
Report Two				
Part One	Team	100%	6/19/23	6/25/23
P1 C	Jasmine	100%	6/21/23	6/24/23
P1 A,B	Thomas	100%	6/21/23	6/24/23
Part two	Thomas	100%	6/26/23	7/1/23
Part Three Sec. 4	Joshua	100%	6/21/23	7/1/23
Part Three Sec. 5	Brett	100%	6/23/23	7/1/23
Part Three Sec. 6	Jasmine	100%	6/23/23	7/1/23
Part Three Sec. 7	Thomas	100%	6/23/23	7/1/23
Part Three Sec. 8	Team	100%	6/28/23	7/1/23
Use Case Points review	Thomas	50%	7/2/23	7/7/23
Update Check		0%	7/10/23	7/14/23
Update Check		0%	7/19/23	7/22/23
Report Three				

CSCI 441 Software Eng. Display Week: 2

TASK	ASSIGNED TO	PROGRESS	START	END
Part One	Team	75%	7/3/23	7/16/23
Part Two	Team	50%	7/3/23	7/23/23
Reflective Essay	Individual	0%	7/16/23	7/23/23
Update Check		0%	7/21/23	7/23/23
Unused		0%	date	date
Code				
Set up Github	Thomas	100%	6/15/23	6/22/23
Front End		0%	6/21/23	7/25/23
Site Map		0%	6/21/23	7/12/23
Sales web pages		0%	7/1/23	7/8/23
Technician Web pages		0%	6/21/23	7/8/23
Parking lot UI display page		0%	6/21/23	7/8/23
Back End		0%	6/21/23	7/25/23
Parking location Optimizer		0%	7/10/23	7/23/23
Parking Lot data structure		0%	6/25/23	6/30/23
Database initialize	Joshua & Jasmine	50%	6/21/23	7/1/23
Database Tests cre	Jasmine	0%	6/21/23	7/1/23
Demo work				
Demo 1 Final Checks		0%	7/8/23	7/10/23
First Demo work		0%	7/3/23	7/12/23
Demo 2 Final Checks		0%	7/15/23	7/23/23
Second Demo Work		0%	7/10/23	7/25/23
Sales integration	Team	0%	8/1/23	9/1/23
Insert new rows ABOVE this one				



4.4 Breakdown of Responsibilities

4.4.1 Developing, Coding, and Testing

Group Member	**Classes/Modules**
Thomas Kerley	Sensor
Joshua Irvin	User
Brett Davis	Car
Jasmine Irvine	Space

4.4.2 Integration

Joshua Irvin will coordinate integration.

4.4.3 Integration

Testing will likely be done for each unit by the group member who developed that unit.

5 Reference list

Brenckle, J. and Stroisch, C. (2022). *Chevrolet and Ford Full Size Pick-Ups Most Stolen Vehicles For Second Year in a Row | National Insurance Crime Bureau*. [online] www.nicb.org. Available at: https://www.nicb.org/news/news-releases/chevrolet-and-ford-full-size-pick-ups-most-stolen-vehicles-second-year-row [Accessed 15 Jun. 2023].

National Insurance Crime Bureau (2023). *Archived Tables*. [online] Insurance Information Institute. Available at: https://www.iii.org/table-archive/21263 [Accessed 15 Jun. 2023].

Netwatch (2023). *Crime Affecting Car Dealerships Is on the Rise*. [online] Netwatch North America. Available at: https://netwatchusa.com/industry-solutions/automotive/ [Accessed 16 Jun. 2023].

Olsen, P. (2022). *Catalytic Converter Theft: 10 Most Targeted Vehicles*. [online] CARFAX. Available at: https://www.carfax.com/blog/catalytic-converter-theft [Accessed 15 Jun. 2023].

Russo, D. (2023). *Cost to Replace a Catalytic Converter (2023) | ConsumerAffairs*. [online] www.consumeraffairs.com. Available at: https://www.consumeraffairs.com/automotive/cost-to-replace-a-catalytic-converter.html [Accessed 15 Jun. 2023].