Introduction

Welcome to Car Aggregate Reform (C.A.R)! Have you ever needed a car on a vacation or while yours was in the shop? Did you want a sweet ride to impress your classmates at your high school reunion or pick up your girl for prom? We here at Ascension Technologies have you covered! Our application is an intuitive, simple way to view and rent a vehicle. Simply creating a personalized account puts our extensive catalog of cars at your fingertips!

Abstract

Car Aggregate Reform, or C.A.R., is a rental car application designed by Ascension Technologies. It allows users to create an account, then easily login and filter and reserve vehicles. Representatives and Administrators can also reserve vehicles, while having the added capability of editing other user accounts. Administrators are able to update the car catalog. The designers used UML diagrams such as class and sequence diagrams. These designs were implemented by seven distinct design patterns and was analyzed using FindBugs for potential errors. In total, the team spent approximately 270hrs building the project.

C.A.R. Vision

Revision History

Version	Date	Description	Author
Inception Draft	Jan 22, 2019	First draft. To be refined during elaboration prior to iteration 1.	Andrew Case
Iteration 1 Draft	Feb 9, 2019	Draft to be presented in iteration 1.	Andrew Case Maggie Burton

Vision Statement

This system provides a simple, efficient, robust application for the provision and acquisition of a wide range of rental cars, promoting the reuse of vehicles.

Business Opportunity

Currently, the vehicle industry is riddled with problems pertaining to recalls and maintenance. This process promotes more complexity and involvement of a driver's busy lifestyle, not to mention the price associated with each repair. Throughout the life of a vehicle, general repairs can be easily avoided through the rental of vehicles. Renting a vehicle also allows drivers to operate exotic or newer model cars they would not otherwise get to use.

Summary of System Features

- Book a reservation for a specific vehicle and cancel a reservation, with real-time transactions and updates to the system
- Payment Authorization
- Option to purchase cars, with the sale being conducted by a representative of the company
- View catalog of vehicles, and allow administrators to update
- Option to purchase insurance from rental company or use personal insurance when renting cars
- Efficient billing and payroll system for company
- Customer assistance email service

Login
Username:
Password:
Login

View Local Listings
C.A.R.
Search
History
Active Rentals
Account Details
Search Filters
Make
Model
Year
Type
Transmission
Mileage
MPG Highway
Interior Color
Exterior Color
Results

Iteration 1 Timecards

Name	Time Spent
Andrew Case	8h 30m
Matthew Darby	8h 15m
Mark Du	8h 1m
Maggie Burton	8h 30m
Stevie Damrel	8h 17m
Weston Straw	8h 14m

Iteration 1 Suggested Points Distribution

All six team members participated and shared the workload evenly. Therefore, it is recommended that each team member be given equal points of the calculated grade.

Iteration 1

Feb 11, 2019

http://

Project manager	
Project dates	Jan 17, 2019 - Feb 13, 2019
Completion	0%
Tasks	10
Resources	6

Tasks

Name	Begin date	End date
Iteration 1	1/17/19	2/12/19
Timecards	1/17/19	2/11/19
Teamwork Plan	1/17/19	2/12/19
Project Vision	1/17/19	1/22/19
Requirements	1/23/19	2/9/19
Use Cases	1/24/19	2/9/19
SSDs	1/27/19	2/9/19
UI Wireframes	2/7/19	2/9/19
Domain Model	2/9/19	2/9/19
Actors	2/9/19	2/10/19

Resources

Name	Default role
Andrew Case	Project Lead
Matthew Darby	Coder
Mark Du	Designer
Maggie Burton	Quality Control
Stevie Damrel	Coder
Weston Straw	Designer

Gantt Chart

GANTT project
January 2019
February 2019
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11
Name Begin ... End d...
Iteration 1 1/17... 2/12...
Timecar... 1/17... 2/11...
Teamw... 1/17... 2/12...
Project ... 1/17... 1/22...
Require... 1/23... 2/9/19
Use Cas... 1/24... 2/9/19
SSDs 1/27... 2/9/19
UI Wiref... 2/7/19 2/9/19
Domain... 2/9/19 2/9/19
Actors 2/9/19 2/10...

Resources Chart

C.A.R. Traceability Matrix

		Use Case ID										Use Case ID									
		Registration	Login	View Catalog	Search Car	Sort Car	"Need Help?"	Select Car	Cancel Reservation	Payment	Bill for damages	Book	Buy Insurance	Personal Insurance	Pickup	Return	Buy	Turnover	Payroll	Maintain Info	Manage personel
	No of Use	Number of uses per Use Case per Req																			
Req ID	Cases per Req	1	1	2	1	1	1	1	1	2	2	2	2	2	1	1	1	2	3	3	4
REQ 1.1	2	x	x																		
REQ 1.2	3			x	x	x															
REQ 1.3	1							x				x									
REQ 1.4	2									x	x										
REQ 1.5	1								x												
REQ 1.6	0												x	x							
REQ 1.7	1						x														
REQ 2.1	2			x																x	
REQ 2.2	2									x	x										
REQ 2.3	1																				x
REQ 2.4	1																		x		
REQ 2.5	1																				x
REQ 3.1	1														x						
REQ 3.2	2															x		x			
REQ 3.3	2										x							x			
REQ 3.4	1																x				
REQ 3.5	5							x		x		x	x	x							
REQ 3.6	1						x														
REQ 3.7	3																		x	x	x
REQ 4.1	3																		x	x	x

Scope: C.A.R.

Actors:

- **Company:** rental car agency
- **Customer:** one who wishes to obtain service from company
- **Sales Representative:** Operates front desk, logs all in-person interaction, handles sales and walk in rentals, customer service
- **System Administrator:** Maintains catalog, handles all updates to system, manages secondary billing processes
- **Subcontractors:** Drive shuttle busses, wash, inspect and repark cars

System: C.A.R.
Requirements List

REQ 1.1: CREATE AND LOGIN
Customer must be able to create and log into account in system.
REQ 1.2: CATALOG
Customer must be able to view car catalog. The customer must be able to search for cars within the catalog and view any specific car.
REQ 1.3: RENT
Customer must be able to request rental car. This car must then be reserved for the customer on the dates of their choosing and at the specified location at that time.
REQ 1.4: PAYMENT
Customer must be able to pay for rental through payment system.
REQ 1.5: CANCELLATION
Customer must be able to cancel reservation. Customer should receive reimbursement for canceled service.
REQ 1.6: INSURANCE
Customer must be able to obtain insurance for rental.
REQ 1.7: ASSISTANCE
Customer must be able to submit questions or requests for assistance through email.
REQ 2.1: UPDATE CATALOG
System administrator must be able to update catalog. This includes adding and deleting cars and editing car specs.
REQ 2.2: BILLING
System administrator must be able to manage billing. This includes payment for rental cars, and additional fees and payment for damages to car. This also includes any reimbursement customers receive.
REQ 2.3: PERSONNEL
System administrator must be able to manage personnel – representatives, subcontractors and system administrator(s) – and customers stored in database.
REQ 2.4: PAYROLL
System administrator must be able to manage payroll for all employees of the company.
REQ 2.5: SUBCONTRACTING
System administrator must be able to manage subcontractors and subcontracting companies.
REQ 3.1: PICKUP
Representative must be able to log rental pickup when customer obtains car.
REQ 3.2: RETURN
Representative must be able to log rental return. This begins the turnover process.
REQ 3.3: INSPECTION
Representative must be able to log inspection results.
REQ 3.4: SALE
Representative must be able to conduct sale of car.
REQ 3.5: WALK-IN
Representative must be able to handle walk-in rental request. This includes booking a request, and entering license and insurance into the system.

REQ 3.6: CUSTOMER ASSISTANCE

Representative must be able to answer “Need Help?” emails. This includes viewing the email and responding as a customer service representative.

REQ 3.7: REPRESENTATIVE HOURS

Representative must be able to clock in/out to record hours worked. This is used in the payroll process.

REQ 4.1: SUBCONTRACTOR HOURS

Subcontractors must be able to clock in/out to record hours worked. This data is sent to the subcontracted company.

Use Case UC1: Create Customer Profile

Scope: Customer Database

Level: Customer Goal

Primary Actor: Customers

Stakeholders and Interests:

- Customer: Wants a profile to be able to rent/return/buy vehicles with.
- Company: Wants a customer profile that accommodates all uses of system.

Preconditions: The customer is ready to create an account.

Main Success Scenario (Basic Flow)

1. Customer selects the create profile button.
2. System displays new customer page.
3. Customer inputs necessary, requested information.
4. System displays all entered information to clarify correctness.
5. Customer selects to submit their information and create account.
6. System adds new customer to the database.

Extensions:

*System Fails.

1. System restarts.
2. Customer logs in.
3. System reconstructs prior state.

1a. Profile link does not work.

1. System displays error to screen with redirection link to home page.
2. Customer can use get help link.

3a. Customer's provided username is already taken.

1. System requests a new username until a non-used one is entered.

3b. Customer's provided email is already associated with an account.

1. System alerts customer.

5a. Customer finds incorrect data in final display of information.

1. System provides customer with edit link to edit the info that is incorrect.

6a. Customer selects submit button, but it fails.

1. System reports error to customer and asks them to wait and try again.

7a. System fails to connect the customer with the database.

1. System reports error to customer and system admin.
2. System requests customer to wait and try again.

Use Case UC2: Login

Scope: Customer Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders and Interests:

- Customer: Wants fast and intuitive login process to allow for easy access to the system.
- Company: Wants efficient login process that is secure and robust.

Preconditions: Customer has an account already that exists within the database and is not flagged.

Success Guarantee (or Postconditions): Customer is granted access to the system.

Main Success Scenario (or Basic Flow):

1. Customer reaches the hosted site and lands on the landing page.
2. Customer enters a login name and its associated password.
3. Customer uses 'Login' button.
4. System initiates authentication process (database verification).
5. Customer is permitted to access the system.
6. Customer's login is recorded within the system.

Extensions:

*System Fails.

1. System restarts.
2. Customer logs in.
3. System reconstructs prior state.

2a. Unauthorized customer attempts to login.

1. System denies access to customer.
2. Customer is prompted to login again.

2b. Customer doesn't have an account.

1. System prompts customer for create account.
2. Customer enters valid credentials.

3a. Customer is a flagged customer.

1. System responds with error message.

4a. Customer enters incorrect information.

1. System denies access.
2. Customer must reattempt to login.

6a. Incorrect login is used.

1. System will not log the login information.

Use Case UC3: View Catalog

Scope: Vehicle Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders and Interests:

- Customer: Wants to have easy access to view the list of rental vehicles.
- Company: Wants to provide customers with ability to view list of rental vehicles.

Preconditions: Customer has logged in.

Success Guarantee (Postconditions): The customer gains access to the list of vehicles.

Main Success Scenario:

1. Customer selects view catalog button.
2. System displays the available rental vehicles, based on popularity.
3. Customer selects desired vehicle.
4. System shows specific vehicle details and information.
5. Customer logs out.

Extensions (Alternative Flows):

*System Fails.

1. System restarts.
2. Customer logs in.
3. System reconstructs prior state.

Use Case UC4: Search Vehicle

Scope: Vehicle Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders and Interests:

- Customers: Wants quick and intuitive search process to find vehicles within provided criteria.
- Company: Wants effective search methods that are unique and simple.

Preconditions: Customer has an associated account, has been identified, and authenticated.

Success Guarantee (or Postconditions): Customer can search the system for registered vehicles with matching criteria.

Main Success Scenario (or Basic Flow):

1. Customer selects search catalog button.
2. System allows the customer to access the vehicle database.
3. Customer then enters search criteria they wish their rental vehicle to have.
4. System updates the customer's search results.
5. Customer selects a vehicle to view.
6. System provides customer with updated information about that vehicle.

Extensions:

*System Fails.

1. System restarts.
2. Customer logs in.
3. System reconstructs prior state.

2a. Customer changes search criteria.

1. Customer deselects the given the option.
2. System updates the provided search results.

Use Case UC5: Sort Vehicle

Scope: Vehicle Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders and Interests:

- Customer(s): Wants a easy-to-use website with straightforward navigation and quick search results to find desired vehicles.
- Company: Wants a unique, simplistic website, making the customers' experience as best as possible.
- Vehicle Company: Wants their vehicle to be accurately displayed and advertised.

Preconditions: The customer has an account and the vehicle catalog exists.

Success Guarantee (or Postconditions): Results of sorted vehicles are viewable by the customer.

Main Success Scenario (or Basic Flow):

1. Actor reaches the main search page.
2. Actor is presented with the catalog of vehicles.
3. Actor enter valid information into the search bar.
4. System returns all the matching vehicles that fit the criteria provided by the customer.
5. Actor can continue to view the provided results or change the search criteria.

Extensions:

*System Fails.

1. System restarts.
2. Customer logs in.
3. System reconstructs prior state.

Use Case UC6: Select Vehicle

Scope: Vehicle Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders & Interests:

- Customers: Selects a given vehicle, providing all the details associated with that vehicle.
- Company: Ensure all essential information associated with a vehicle is in the hands of the customer.

Preconditions: The vehicle is registered within the system and meets the customer's specified search criteria.

Postconditions: The vehicle view count is updated.

Main Success Flow:

1. Customer selects a vehicle they wish to rent.
2. System displays all information pertaining to that vehicle for the customer.
3. System updates view count associated with the vehicle.
4. Customer chooses the "rent" option provided with selected vehicle.
5. System prompts the customer for the amount of time they will rent the vehicle for and where to pick it up from.
6. Customer selects the amount of time they would like to rent for and pickup location.
7. System displays a success dialogue box and puts the selected vehicle in the rented section of the customer.

Extensions:

*System Fails.

1. System restarts.
2. Customer logs in.
3. System reconstructs prior state.

4a. Customer wants to buy the vehicle, rather than rent it.

1. System confirms the customer's choice of buying, rather than renting
    1. Customer proceeds with the buying process.
        i. System initiates the buying process of a vehicle.
    2. Customer signifies that they do not want to buy the vehicle, rather rent it instead.
        i. System returns the customer back to the vehicle being viewed.

7a. Customer wants to cancel a rental request.

1. Customer proceeds to view rented vehicle on their profile.
2. System returns the vehicles currently on request OR actively rented.
    a. Customer selects return vehicle.
    b. System initiates the customer's return request.

Use Case UC7: Select Reservation

Scope: Vehicle Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders & Interests:

- Customers: Selects to rent a particular vehicle or cancel a previously set reservation.
- Company: Provide the customer with a simplistic rental process, including confirmation and cancellation screens.

Preconditions: The vehicle is selected by the customer. The customer wishes to rent the selected vehicle.

Postconditions: The customer has set a rental period and pickup location. The vehicle is updated as “in use.”

Main Success Flow:

1. Customer selects a vehicle they wish to rent.
2. Customer chooses the “rent” option provided with selected vehicle.
3. System prompts the customer for the amount of time they will rent the vehicle for and where to pick it up from.
4. Customer selects the amount of time they would like to rent for and pickup location.
5. System displays a success dialogue box and puts the selected vehicle in the rented section of the customer.

Extensions:

*System Fails.

4. System restarts.
5. Customer logs in.
6. System reconstructs prior state.

1a. Customer wants to cancel a rental request.

1. Customer proceeds to view rented vehicle on their profile.
2. System returns the vehicles currently on request OR actively rented.
    a. Customer selects return vehicle.
    b. System initiates the customer’s return request.

Use Case UC8: Payment/Billing System

Scope: Vehicle Rental Payment System

Level: Customer Goal

Primary Actor: Employee

Stakeholders and Interests:

- Employee: Wants accurate and quick when processing payment.
- Customer: Wants a pleasant purchase experience, fast service with minimal effort, and a well-organized display of purchases and costs.
- Company: Wants to accurately record transactions and satisfy customer interests.

Preconditions: Employee is identified and authenticated. Customer has paid for a service.

Success Guarantee (or Postconditions): Sale is saved, tax is correctly calculated, inventory is adjusted, and receipt is generated.

Main Success Scenario:

1. Customer arrives at checkout.
2. Employee uses POS and records the rental purchase.
3. System displays the total cost of the purchase along with the items purchased.
4. System requests customer for a type of payment (cash, debit, credit).
5. Customer pays using their preferred method of payment.
6. System validates the transaction, makes a record, and updates the inventory.
7. Customer receives a receipt from the System.

Extensions (Alternative flows):

*System Fails.

7. System restarts.
8. Customer logs in.
9. System reconstructs prior state.

*b. Invalid item ID

1. System signals error and rejects entry
2. Employee responds to error, manually enters in ID
3. Employee continues with sale

Use Case UC9: Bill for Damages

Scope: Payment System

Level: System Goal

Primary Actor: System

Stakeholders and Interests:

- Company: Wants to receive compensation for any damages to its vehicles as efficiently as possible.
- Customer: Wants to be billed in a fair and timely manner.
- Insurance Company: Wants to pay the minimum in repairs.

Preconditions: Vehicle is returned, and the system reflects that.

Success Guarantee (or Postconditions): The vehicle is inspected, washed and rentable again. The system is updated to reflect this.

Main Success Scenario (or Basic Flow):

1. Company reports damages during inspection of vehicle.
2. System logs damages from vehicle inspection.
3. System logs costs of damages, upon vehicle being fixed.
4. System bills the customer's insurance or the customer for the cost to fix any damages.
5. Insurance company or the customer pays for the damage in full.

Extensions (or Alternative Flows):

*System Fails.

1. System restarts.
2. Customer logs in.
3. System reconstructs prior state.

4a. The Insurance Company denies claim:

1. Bill is sent to customer instead.
2. Customer appeals to insurance company.

a) Insurance company accepts claim and pays for repairs.

b) Insurance again denies claim and customer is liable.

3. Customer pays for damages independent from insurance company.

4b. Customer neglects to pay for damages:

4c. No payment is logged within given time frame.

4d. Customer is sent more billing statements.

1. Customer pays for damages.
2. Customer refuses to provide compensation.

4e. Company files charges against customer for breaking contract.

1. Customer is convicted on criminal charges.
2. Customer is found innocent of breaking contract.

4f. Company files civil suit for damages and legal fees.

1. Customer is held liable for damages and fees and ordered to pay.
2. Customer is found not to be liable for damages or fees and company must pay.

Special Requirements:

- Robust recovery of sensitive data such as billing or insurance information
- Alert System for overdue bills

Frequency of Occurrence: As often as vehicles are returned having accrued damages.

Open Issues:

- What were the circumstances of the damage occurring?

Use Case UC10: Purchase Insurance Policy

Scope: Customer Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders and Interests:

- Customers: Wants to have the option of purchasing a temporary insurance plan through the rental company.
- Company: Wants to provide the customer with an insurance purchasing option in case the customer does not have insurance or does not want to use their own policy.

Preconditions: Customer has an account within the database.

Success Guarantee (or Postconditions): Customer is granted a company insurance policy.

Main Success Scenario (or Basic Flow):

1. Customer reaches the 'Purchase Policy Through C.A.R.' part of the application.
2. Customer selects 'Yes'.
3. System adds insurance fee to the Actor's total purchase.
4. Customer is granted an insurance policy.

Extensions:

*System Fails.

10. System restarts.
11. Customer logs in.
12. System reconstructs prior state.

1a. Customer can purchase a policy later even if they declined it initially.

2a. Customer selects 'No'

2a1. Customer will not be permitted to rent/buy a vehicle

Use Case UC11: Use Personal Insurance Policy

Scope: Customer Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders and Interests:

- Customers: Wants to have the option of using their own auto insurance policy.
- Company: Wants to allow customers to provide their own auto insurance policy.

Preconditions: Customer has an account within the database.

Success Guarantee (or Postconditions): Customer is listed in the database as using their personal policy.

Main Success Scenario (or Basic Flow):

1. Customer reaches the 'Purchase Policy Through C.A.R.' part of the application.
2. Customer selects that they would like to use their personal auto insurance policy.
3. System prompts customer for their auto insurance info.
4. Customer's auto info is entered into the database and the application continues.

Extensions:

*System Fails.

13. System restarts.
14. Customer logs in.
15. System reconstructs prior state.

1a. Customer can opt to use their personal auto insurance policy later if they decide to do so

Use Case UC12: Vehicle Pickup

Scope: Vehicle Database

Level: Customer Goal

Primary Actor: Customer

Stakeholders and Interests:

- Customer: Wants to have easy access to rental vehicle pickup without any issues.
- Company: Wants to provide customers with easy process for rental vehicle pickup.

Preconditions: Customer has a valid driver's license and is capable of operating a motor vehicle.

Success Guarantee (Postconditions): The customer leaves with their rental vehicle.

Main Success Scenario:

1. Customer arrives at location of vehicle pickup.
2. Customer finds a parking spot and parks.
3. Customer meets with the employee and views the vehicle.
4. Customer accepts the vehicle and receives the keys.
5. System updates the inventory.
6. Customer leaves the place with their rental vehicle.

Extensions (Alternative Flows):

*System Fails.

16. System restarts.
17. Customer logs in.
18. System reconstructs prior state.

1a. If the chosen vehicle is unavailable

1. Another vehicle may be chosen instead.

2a. The customer declines their rental vehicle

1. The employee moves the rental vehicle back in inventory.

Use Case UC13: Return Vehicle

Scope: System Database

Level: System Goal

Primary Actor: On Sight Attendant

Stakeholders and Interests:

- On Sight Attendant: Wants to input information to return vehicle successfully so database knows where and when it is available.
- Customer: Wants to return vehicle and be charged for rental.

Preconditions: Customer must have a vehicle to return

Main Success Scenarios (Basic Flow):

1. Customer arrives at physical location of rental service and parks in an empty spot.
2. On sight attendant approaches and begins return process.
3. Attendant logs in to system and navigates to return vehicle section
4. Attendant asks customer for username or email to look up rental information
5. Attendant selects rental from customer profile that they are returning
6. System prompts for location of return, information on vehicle appearance (scratches, etc.), and for mileage. Also provides information about previous damages to ensure customer is not charged for damages they did not cause.
7. Attendant submits required information.
8. System calculates cost of rental based on days rented, damages, and mileage
9. Attendant shows customer total cost of rental and gets their confirmation
10. System charges the card on file for the cost of the rental.
11. Ensure the card goes through, if not, alert attendant who will request a different one from the customer at that moment.
12. Display a thank you for using our business message for attendant to say to the customer

Extensions (Alternate Flow):

*System Fails.

19. System restarts.
20. Customer logs in.
21. System reconstructs prior state.

3.a Username and email are not recognized in system

2.a.1 Attendant asks for customer's actual name and looks it up via their name

4.a Rental vehicle being returned does not show on system

3.a.1 Contact location where the customer says the vehicle was rented from for information needed to return vehicle

8.a Customer argues over price or requests more information about why they are being charged more than expected

8.a.1 Attendant shows the customer the cost calculation, pointing out any damages they marked and the total mileage and time

10.a New card fails as well

10.a.1 Attendant asks for new card until one is accepted or receives cash payment as last resort

Use Case UC14: Buy Vehicle

Scope: Vehicle Database

Level: Customer Goal

Primary Actor: Customer

Stakeholders and Interests:

- Company: Wants to sell the vehicle at the highest possible price and accurately record sale and wants payment in full of customer or third party loan.
- Customer: Wants to purchase vehicle in good condition for lowest offered price as fast as possible.
- Bank: Wants to ensure customer will repay loan through credit history or collateral, and profit from interest on loan.
- Government Tax Agencies: Wants to collect tax from the sale.
- Salesman: Wants to quickly get customer approved for loan and sell the vehicle.

Preconditions: Customer desires to purchase vehicle.

Success Guarantee (or Postconditions): Customer owns vehicle. Sale is saved. Tax is correctly calculated. System is updated to reflect sale.

Main Success Scenario (or Basic Flow):

1. Customer decides to purchase used vehicle from rental agency.
2. Salesman starts a new sale.
3. Salesman/Company and customer negotiate a final price.
4. Customer provides proof of existing insurance and valid driver's license.
5. Customer applies for loan from personal bank or bank associated with company.
6. Customer is approved for loan and payment plan is negotiated and approved.
7. Customer pays down payment (check, debit, or cash).
8. Vehicle registration is transferred to customer.
9. System is updated to reflect sale.

10. Customer leaves with vehicle.

11. Customer continues to make loan payments until vehicle is paid in full or resells vehicle.

Extensions (or Alternative Flows):

*System Fails.

22. System restarts.

23. Customer logs in.

24. System reconstructs prior state.

2a. Customer and company cannot agree on price:

1. Customer decides to pay company offered price.
2. Company discounts vehicle to meet customer desires.
3. Customer decides not to purchase vehicle.
    a. Sale is voided.

2b. The loan application is denied:

1. Customer reapplies for loan.
2. Loan is approved.
    a. Sale continues.
3. Loan is again denied.
    a. Customer proceeds with other options.

2c. Customer pays in full independent of third party.

1. Customer does not purchase vehicle.
    a. Sale is voided.

2d. Customer does not have valid vehicle insurance:

1. Customer purchases insurance after sale completion.
2. Bank and company are sent proof of new insurance.
3. Customer leaves with vehicle.

2e. Company holds vehicle while customer obtains insurance.

1. Company holds vehicle for 3 business days.
2. Customer returns with proof of insurance.
    a. Sale continues.
3. Customer does not obtain insurance.
    a. Sale is voided.

2f. Customer cannot obtain insurance.

1. Customer cannot take vehicle until he/she or other driver proves vehicle is insured.

2g. Customer does not purchase vehicle.

1. Sale is voided.

2h. Customer does not have valid license:

2i. Customer obtains valid license after sale.

1. Customer returns provides proof of valid license.
2. Customer leaves with vehicle.

2j. Company holds vehicle while customer obtains license.

1. Company holds vehicle for 3 business days.
2. Customer returns with proof of license.
    a. Sale continues.
3. Customer brings another driver with valid license.
    a. Sale continues.
4. Customer does not return.
    a. Sale is voided.

2k. Customer brings another licensed driver to take vehicle.

1. Sale continues.

2l. Customer does not purchase vehicle.

1. Sale is voided.

2m. Customer cannot pay down payment:

1. Company holds vehicle for 3 business days.
2. Customer returns with down payment.
    a. Sale continues.
3. Customer does not return with valid payment.
    a. Sale is voided.

2n. Customer decided not to purchase vehicle:

1. Sale is voided.

Special Requirements:

- Copy machine or scanner required to save license and proof of insurance.

Frequency of Occurrence: When customer wishes to purchase vehicle.

Open Issues:

What are requirements for obtaining license plate and registration in state?

Use Case UC15: Vehicle Turnover

Scope: Vehicle Database

Level: Customer Goal

Primary Actor: On Sight Attendant

Stakeholders and Interests:

- Company: Wants to make returned vehicles rentable again as fast as possible to maximize profit.
- On Sight Attendant: Wants to efficiently log vehicle turnover and deal with issues.

Preconditions: A vehicle is returned by a customer.

Success Guarantee (or Postconditions): The vehicle is inspected, washed and rentable again. The system is updated to reflect this.

Main Success Scenario (or Basic Flow):

1. Customer returns rental vehicle.
2. On Sight Attendant gets vehicle washed.
3. On Sight Attendant inspects vehicle and any damage is logged or it is recorded there was no damage.
4. On Sight Attendant Gas fills tank and amount put in is recorded.
5. On Sight Attendant parks vehicle in the rental lot.
6. System is updated to reflect the care is available again.

Extensions (or Alternative Flows):

*System Fails.

25. System restarts.
26. Customer logs in.
27. System reconstructs prior state.

3a. Vehicle requires repairs.

1. Vehicle is sent to mechanic and repaired.

2. If vehicle is deemed totaled, it is decommissioned.
3. On Sight Attendant begins process to bill customer who caused damages.
4. Vehicle is returned and placed back in rental lot.

3b. Vehicle is decommissioned.

3c. Vehicle is sold to customer.

3d. Vehicle is resold to manufacturer.

3e. Vehicle is sold at wholesale at auction.

3f. Vehicle is sold for parts or to a salvage yard.

Frequency of Occurrence: Every time a vehicle is returned; could be nearly continuous during hours of operation.

Use Case UC16: Manage Personnel

Scope: Customer Database

Level: System Goal

Primary Actor: System Admin

Stakeholders and Interests:

- System Admin: Wants efficiently edit personnel profiles.
- Personnel: Wants to perform quality service and retain employment.
- Company: Wants encouraging, manageable personnel.

Preconditions: Personnel have manageable accounts and information pertaining to their employment.

Success Guarantee (or Postconditions): All personnel are accounted for within the system, allowing for easy contacting.

Main Success Scenario (or Basic Flow):

1. Personnel are hired for a specific task.
2. System confirms that the information is correct for their profile.
3. System Admin uses the personnel profile to assign necessary tasks.
4. Personnel perform the tasks, logging hours associated with their profile.

Extensions:

*System Fails.

28. System restarts.
29. Customer logs in.
30. System reconstructs prior state.

Use Case UC17: Maintain Payroll

Scope: System Database

Primary Actor: System Admin

Stakeholders and Interests:

- System Admin: Wants to update customers in the database, including removing inactive customers, old representatives, and adding new representatives.

Preconditions: Administrator has access to the system and an account.

Main Success Scenario (Basic Flow):

1. Admin logs on to the system.
2. Admin navigates to representative page.
3. System displays list of representatives.
4. Admin navigates to payroll for representatives.
5. System displays the payroll for all representatives.
6. Admin selects representatives that need their payroll updated.
7. System displays the selected representative's payroll information.
8. Admin changes the amount a representative is paid and submits the changes.
9. System updates the payroll information in the database for the representative.

Extensions (Alternative Flow):

1.a Admin login or account is invalid.

1. System asks customer for different account or to create a new one

2.a Link navigates to somewhere other than representative page

4.a Link navigates to somewhere other than payroll page

6.a Link navigates to somewhere other than the selected representative's payroll page

8.a Admin inputs data that is not accepted in the payroll field

1. Display error and request correct data

Use Case UC18: Request Help

Scope: Customer Database

Level: Customer Goal

Primary Actor: All Customers

Stakeholders and Interests:

- Customer(s): Wants fast responses, great UI, and various preset topics.
- Company: Wants feedback for constant improvement.

Preconditions: The customer has an account already (in the database) and is not flagged.

Success Guarantee (or Postconditions): Customer's feedback (in the form of an email) is sent.

Main Success Scenario (or Basic Flow):

1. Customer wants help with an issue.
2. Customer selects the need help button to request assistance.
3. Customer composes an email.
4. Customer submits the email to the support page.
5. System sends the email to the correct help email.

Extensions:

*System Fails.

1. System restarts.
2. Customer logs in.
3. System reconstructs prior state.

:Customer
: System
1: SelectPersonalPolicy()
1.1: PromptForInsurance()
2: EnterInsuranceInfo()
2.1: LogInsuranceInfo()
2.2: ConfirmInsurance()
Powered By Visual Paradigm Community Edition

:Customer
: System
sd WhileNotRentalConfirmed
1: SelectDesiredVehicle()
1.1: DisplayVehicleInfo()
2: SelectRent()
2.1: PromptForRentalDates()
3: EnterDates()
3.1: PromptForLocation()
4: EnterLocation()
alt
4.2: DisplayConfirmation()
4.1: PlaceCarOnHold()
4.3: DisplayCarUnavailable()
Powered By Visual Paradigm Community Edition

:Customer
: System
1: SelectPurchasePolicy()
1.1: AddInsuranceFee()
1.2: LogPolicyForUser()
1.3: ConfirmPolicyNewPrice()
Powered By Visual Paradigm Community Edition

:Customer
: System
sd loop
1: Login(username, password)
1.1: Validate(User)
1.2: PermitUser()
1.3: RecordAccess(User)
Powered By Visual Paradigm Community Edition

:Customer
: System
1: createProfile()
1.1: DisplayCreateCustomer()
sd loop
2: UserInput(User)
2.1: Validation(User)
3: Submit()
3.1: AddUser(User)
Powered By Visual Paradigm Community Edition

:Company
: System
1: ReportDamages()
1.1: LogDamages()
1.2: LogRepairCost()
1.3: ReportRepairCost()
1.4: SendBill()
2: ReportPaymentReceived()
2.1: LogPaymentReceived()
2.2: ConfirmPaymentReceived()
Powered By Visual Paradigm Community Edition

:Customer
: System
1: ViewCatalog()
1.1: DisplayCatalog(PopularCars)
2: CarSelected()
2.1: DisplayCar(Car)
Powered By Visual Paradigm Community Edition

:Employee
: System
1: washVehicle()
2: inspectVehicle()
2.1: recordVehicleStatus()
3: fillGasTank()
4: parkVehicle()
4.1: makeVehicleAvailable
Powered By Visual Paradigm Community Edition

:Customer
: System
1: SearchPage()
1.1: DisplayCatalog(PopularCars)
loop
2: SearchCriteria()
2.1: DisplayCatalog(CarCriteria)
Powered By Visual Paradigm Community Edition

:Customer
: System
1: SelectVehicle()
1.1: DisplayCar(Car)
2: RentCar()
2.1: RentalInformation()
2.2: RentalInformation(startDate, endDate, pickUpLocation)
2.3: Success()
2.3: CarToRentedSection()
Powered By Visual Paradigm Community Edition

:Customer
: System
1: SearchCatalog()
1.1: DisplayCatalog(PopularCars)
2: SearchCriteria()
2.1: DisplayCatalog(CarResults)
3: CarSelected()
3.1: DisplayCar(Car)
Powered By Visual Paradigm Community Edition

:Customer
: System
1: requestHelp()
2: composeEmail(statement)
2.1: replyEmail(statement)
Powered By Visual Paradigm Community Edition

:Employee
: System
1: RecordRental(Car)
1.1: ...
1.2: RequestPayment()
1.3: ReceivePayment(PaymentType)
1.4: ...
1.5: RecordRental(Car)
1.6: UpdateInventory(Car)
1.7: PrintReceipt()
Powered By Visual Paradigm Community Edition

:Administrator
: System
1: hirePersonnel(Employee)
2: assignTask(task)
2.1: updatePersonnelInfo()
3: logHours(hours)
3.1: updateProfileInfo()
Powered By Visual Paradigm Community Edition

:Administrator
: System
loop
1: login(username, password)
1.1: Validate(User)
2: showRepresentatives(Employee)
2.1: DisplayRepresentatives
3: showPayroll(payroll)
3.1: displayPayroll(payroll)
4: selectRepresentative(Employee)
4.1: displayRepPayrollInfo(info)
5: changeRepPayroll()
5.1: updateRepPayroll()
Powered By Visual Paradigm Community Edition

:Customer
: System
1: startSale(sale)
2: enterInsuranceInfo(info)
2.1: updateInsuranceInfo()
3: makePayment()
3.1: updateStatus
Powered By Visual Paradigm Community Edition

Administrator
System
1: makeCarAvailable()
1.1: carAvailable
2: setVehiclePickup()
2.1: vehiclePickupReady
Powered By Visual Paradigm Community Edition

Customer
System
loop
1: login(username, password)
1.1: homePage()
2: searchCar(carName)
2.1: displayCar()
3: viewCatalog()
3.1: show Car List
4: sortCars()
4.1: displayCar()
5: bookReservation(name)
6: cancelReservation()
6.1: reservationCancelled
7: addCar(carType)
7.1: new Car added
8: makePayment(amount)
8.1: print receipt
9: PurchaseInsurance()
9.1: receiveInsurance

Employee
System
1: makeNewSale()
2: enterCar(carID, quantity)
2.1: print description and total
3: makeCarAvailable()
3.1: updateCarAvailability
4: billForDamages()
4.1: print total
Powered By Visual Paradigm Community Edition

Domain Model
Sends Information to
WindowView
-jpanel : JPanel
Calls
1
1
AdminController
Updates
User
-username : String
-password : String
-email : String
-fullName : String
-creditcard : int
1..*
Receipt
-id : int
-date : date
-amount : double
-username : String
Receives>
0..*
1..*
Calls
Edits
1
UserController
Prints
1
1
1..*
1
DatabaseAdaptor
1
1
Sends Information to
Calls
Representative
1..*
1..*
Creates
Returns
Generates
1..*
Rental
-rentalStart : date
-rentalTime : int
-lateFee : double
-tax : double
-cost : double
-username : String
1..*
Calls
Administrator
Update
1
RepresentativeController
+addCar() : Car
+removeCar(Car car) : void
+updateCar() : Car
0..*
1..*
Updates
Has
1
Vehicle
-vin : string
-make : String
-model : String
-rented : boolean
-costPerDay : double
-mileage : double
-type : String
-transmission : String
-cylinders : int
-colorInt : String
-colorExt : String
1
Sends Information to
Powered By Visual Paradigm Community Edition

C.A.R. Vision

Revision History

Version	Date	Description	Author
Iteration 2 Draft	Mar 27, 2019	Draft to be presented in Iteration 2.	Andrew Case
Inception Draft	Jan 22, 2019	First draft. To be refined during elaboration prior to iteration 1.	Andrew Case
Iteration 1 Draft	Feb 9, 2019	Draft to be presented in iteration 1.	Andrew Case Maggie Burton

Vision Statement

This system provides a simple, efficient, robust application for the provision and acquisition of a wide range of rental cars, promoting the reuse of vehicles.

Business Opportunity

Currently, the vehicle industry is riddled with problems pertaining to recalls and maintenance. This process promotes more complexity and involvement of a driver's busy lifestyle, not to mention the price associated with each repair. Throughout the life of a vehicle, general repairs can be easily avoided through the rental of vehicles. Renting a vehicle also allows drivers to operate exotic or newer model cars they would not otherwise get to use.

Summary of System Features

- Book a reservation for a specific vehicle and cancel a reservation, with real-time transactions and updates to the system
- Payment Authorization
- Option to purchase cars, with the sale being conducted by a representative of the company
- View catalog of vehicles, and allow administrators to update
- Option to purchase insurance from rental company or use personal insurance when renting cars
- Efficient billing and payroll system for company
- Customer assistance email service

Iteration 2

Mar 24, 2019

http://

Project manager	
Project dates	Jan 17, 2019 - Mar 27, 2019
Completion	0%
Tasks	23
Resources	6

Tasks

Name	Begin date	End date
Iteration 1	1/17/19	2/12/19
Timecards	1/17/19	2/11/19
Teamwork Plan	1/17/19	2/12/19
Project Vision	1/17/19	1/22/19
Requirements	1/23/19	2/9/19
Use Cases	1/24/19	2/9/19
SSDs	1/27/19	2/9/19
UI Wireframes	2/7/19	2/9/19
Domain Model	2/9/19	2/9/19
Actors	2/9/19	2/10/19
Iteration 2	2/19/19	3/26/19
Timecards	2/19/19	3/26/19
Teamwork Plan	2/19/19	2/19/19
Updated SSDs	2/20/19	2/21/19
Updated Domain Model	2/20/19	2/21/19
Updated UI Wireframes	2/22/19	2/24/19
Design Class Model	2/22/19	2/24/19
Test Coverage Plan	2/25/19	2/28/19
Design for Demo Code	3/1/19	3/8/19
Code Implementation	3/18/19	3/24/19
Quality Control for Iteration 1 and 2	3/18/19	3/24/19
Final Testing	3/25/19	3/25/19
Video	3/25/19	3/25/19

Resources

Name	Default role
Andrew Case	Project Lead
Matthew Darby	Coder
Mark Du	Designer
Maggie Burton	Quality Control
Stevie Damrel	Coder
Weston Straw	Designer

Gantt Chart

Name	Begin date	End date
Iteration 1	1/17/19	2/12/19
Timecards	1/17/19	2/11/19
Teamwork Plan	1/17/19	2/12/19
Project Vision	1/17/19	1/22/19
Requirements	1/23/19	2/9/19
Use Cases	1/24/19	2/9/19
SSDs	1/27/19	2/9/19
UI Wireframes	2/7/19	2/9/19
Domain Model	2/9/19	2/9/19
Actors	2/9/19	2/10/19
Iteration 2	2/19/19	3/26/19
Timecards	2/19/19	3/26/19
Teamwork Plan	2/19/19	2/19/19
Updated SSDs	2/20/19	2/21/19
Updated Domain Model	2/20/19	2/21/19
Updated UI Wireframes	2/22/19	2/24/19
Design Class Model	2/22/19	2/24/19
Test Coverage Plan	2/25/19	2/28/19
Design for Demo Code	3/1/19	3/8/19
Code Implementation	3/18/19	3/24/19
Quality Control for Itera...	3/18/19	3/24/19
Final Testing	3/25/19	3/25/19
Video	3/25/19	3/25/19

Resources Chart

C.A.R - GRASP Implementations

- **Creator** – Classes that edit other objects have creator status
  - The RepresentativeController is the creator for *Car* objects
  - The AdminController is the creator for *User* objects
  - The UserController is the creator for *Rental* objects
- **Low Coupling** – classes communicate minimally with each other
  - *Cars* have no knowledge of *Users*
  - *Cars* and *Representatives* have no knowledge of *Receipts*
  - *Users* (Customers) can only view *Cars*
  - *Receipts* have no knowledge of *Cars* or *Representatives*
  - *Receipts* only show data from *Users* and *Rentals*
- **High Cohesion** – classes have narrow and specific responsibilities
  - Controllers take input and create appropriate objects
  - *User* (and subtypes), *Receipt*, *Rental*, *Car* store only data pertinent to the individual object
- **Controller** – handles input and object creations
  - *RepresentativeController* handles input for new *Car*
  - *AdminController* takes input for new *User*
  - *UserController* takes input for new *Rental*
- **Polymorphism** – Subclasses are implemented to expand functionality of Superclasses
  - *Representative* extends *User*; it is a specialized type of *User*, with broader access to data in the system
  - *Administrator* extends *Representative*; it is a subtype of *Representative*, who has clearance to edit and view larger amounts of data
- **Indirection** – Classes call other classes to accomplish tasks
  - User calls *UserController* to view *Cars*
  - Administrator calls *AdminController* to edit *Users*

C.A.R. Testing Coverage Plan

In order to ensure correctness in the C.A.R. app, we will use Unit Testing, implemented with jUnit, to test the Create Account, Login, Select Reservation, Personal Insurance Policy, Payment/Billing System, Maintain Payroll and Manage Personnel functions. These being the functions where a user, whether customer or administrator, could enter data that is invalid. We will run the following tests:

Create Account:

- All fields filled, through bounds checking
- Credit Card Number correct length and format, through bounds checking and passing bad data
- Account created upon button click, through action listeners
- Password meets criteria, through checking for required characters

Login:

- Username and Password correspond and are valid, through validating existence in database
- Login initiated upon button click, through action listeners

Select Reservation:

- Car is available for selected dates, through searching in all active reservations

Personal Insurance Policy:

- All fields filled, through bounds checking
- Insurance current, through validating expiration date is in the future

Payment/Billing Information:

- Administrator can edit bill but cannot delete, through entering null data

Maintain Payroll:

- Payment meets federal minimum wage, through testing invalid input
- Administrator cannot edit his/her own payroll, through bounds checking
- Hours logged does not exceed possible hours, through bounds checking

Manage Personnel

- Employee profiles cannot be duplicated, through testing invalid input
- New Employee profile must be complete, through bounds checking

Iteration 2 Suggested Points Distribution

All six team members participated and shared the workload evenly. Therefore, it is recommended that each team member be given equal points of the calculated grade.

Iteration 2 Timecards

Name	Time Spent
Andrew Case	37h 00m
Matthew Darby	22h 15m
Mark Du	22h 37m
Maggie Burton	19h 30m
Stevie Damrel	14h 30m
Weston Straw	27h 14m

:Administrator
: WindowView
: Database
1: getAddCar()
1.1: CreatorPage()
2: ClickSubmitCar()
2.1: AddCar(Car)
2.1.1: SaveCar(Car)
2.1.2: ConfirmSaveCar()
2.2: Refresh()
Powered By Visual Paradigm Community Edition

Add Car

: Administrator
: WindowView
: Database
1: getRemoveUser()
1.1: RemoveUserPage()
2: getSubmitUser()
2.1: RemoveUser()
3: deleteUser(User)
4: ConfirmDelete
4.1: Refresh()
Powered By Visual Paradigm Community Edition

Remove User

sd Sequence Diagram Create Rental
: User
: WindowView
: UserController
: DatabaseAdapter
1: clickRent()
1.1: clickRent()
1.1.1: createRental(Rental rent)
1.1.1.1: saveRental()
1.1.1.2: success : boolean
1.1.1.2.1: createSuccessFailPage(boolean success)
Powered By Visual Paradigm Community Edition

sd Sequence Diagram Create User
: Administrator
: WindowView
: AdminController
: DatabaseAdaptor
1: clickAddUser()
1.1: createUserPage()
2: clickSubmitUser()
2.1: clickSubmitUser()
2.1.1: createUser()
2.1.1.1: saveUser(User user)
2.1.1.2: success : boolean
2.1.1.2.1: createSuccessFailPage()
Powered By Visual Paradigm Community Edition

:Administrator
: WindowView
: AdminController
: DatabaseAdapter
1: clickUpdateUser()
1.1: updateUserView()
2: clickSubmitUser()
2.1: clickUpdateUser()
2.1.1: demoteUser()
2.1.1.1: saveUser(User user)
2.1.1.2: userStatus
2.1.1.2.1: userStatus
2.1.1.2.1.1: refreshView()
Powered By Visual Paradigm Community Edition

Demote User

Edit Account

:User
: WindowController
: UserController
: Database
1: getPayRental()
1.1: PayRentalPage()
2: getSubmitPay()
2.1: SubmitPay(User)
2.1.1: ValidatePayment()
2.1.2: PayRental(User)
2.1.2.1: SaveRentalHistory()
2.1.2.2: SaveReceipt()
2.1.2.3: getUserReceipts(User)
2.1.3: ReceiptPage()
Powered By Visual Paradigm Community Edition

Pay Rental

Print Receipt

Promote User

sd Sequence Diagram Remove Car
: Representative
: WindowView
: RepresentativeController
: DatabaseAdapter
1: clickRemoveCar()
1.1: createConfirmPage()
2: clickConfirm()
2.1: clickConfirm()
2.1.1: removeCar(Car car)
2.1.1.1: removeCar(Car car)
2.1.1.2: success : boolean
2.1.1.2.1: createSuccessFailPage(boolean success)
Powered By Visual Paradigm Community Edition

Reset Password

:User
: WindowView
: UserController
: Database
1: getSearchVehicle()
1.1: vehicleView()
2: ClickSubmitCar()
2.1: searchCar(Vehicle)
2.1.1: GetCar(Vehicle)
2.1.2: Car
2.2: Car
2.3: Refresh()
Powered By Visual Paradigm Community Edition

Search Car

sd Sequence Diagram Update Car
: Representative
: WindowView
: RepresentativeController
: DatabaseAdapter
1: clickUpdateCar()
1.1: createUpdateCarView()
2: clickConfirm()
2.1: clickConfirm()
2.1.1: updateCar()
2.1.1.1: saveCar(Car car)
2.1.1.2: success : boolean
2.1.1.2.1: createSuccessFailPage(boolean success)
Powered By Visual Paradigm Community Edition

sd Update User
: Administrator
: WindowView
: AdminController
: DatabaseController
1: clickUpdateUser()
1.1: updateUserView()
2: clickSubmitUser()
2.1: clickSubmitUser()
2.1.1: updateUser()
2.1.1.1: saveUser(User)
2.1.1.2: success: boolean
2.1.1.2.1: createSuccessFailPage()
Powered ByVisual Paradigm Community Edition

sd Sequence Diagram Validate Card
: User
: WindowView
: UserController
1: clickConfirm()
1.1: clickConfirm()
1.1.1: validateCard()
1.1.2: success : boolean
1.1.2.1: createSuccessFailPage(boolean success)
Powered By Visual Paradigm Community Edition

sd Sequence Diagram Validate Email
: User
: WindowView
: UserController
1: clickConfirm()
1.1: clickConfirm()
1.1.1: validateEmail()
1.1.2: success : boolean
1.1.2.1: createSuccessFailPage(boolean success)
Powered By Visual Paradigm Community Edition

sd Sequence Diagram Validate Password
: User
: WindowView
: UserController
1: clickConfirm()
1.1: clickConfirm()
1.1.1: validatePassword()
1.1.2: success : boolean
1.1.2.1: createSuccessFailPage(boolean success)
Powered By Visual Paradigm Community Edition

Validate Username

C.A.R Vision

Revision History

Version	Date	Description	Author
Iteration 3 Draft	May 2, 2019	Draft to be presented in Iteration 3	Mark Du Maggie Burton
Iteration 2 Draft	Mar. 27, 2019	Draft to be presented in Iteration 2	Andrew Case
Inception Draft	Jan. 22, 2019	First draft. To be refined during elaboration prior to Iteration 1.	Andrew Case
Iteration 1 Draft	Feb. 9, 2019	Draft to be presented in iteration 1	Andrew Case Maggie Burton

Vision Statement

This system provides a simple, efficient, robust application for the provision and acquisition of a wide range of rental cars, promoting the reuse of vehicles.

Business Opportunity

Currently, the vehicle industry is riddled with problems pertaining to recalls and maintenance. This process promotes more complexity and involvement of a driver's busy lifestyle, not to mention the price associated with each repair. Throughout the life of a vehicle, general repairs can be easily avoided through the rental of vehicles. Renting a vehicle also allows drivers to operate exotic or newer model cars they would not otherwise get to use.

Summary of System Features

- Book a reservation for a specific vehicle and cancel a reservation, with real-time transactions and updates to the system
- Payment Authorization
- Option to purchase cars, with the sale being conducted by a representative of the company
- View catalog of vehicles, and allow administrators to update
- Option to purchase insurance from rental company or use personal insurance when renting cars
- Efficient billing and payroll system for company
- Customer assistance email service

C.A.R.
User’s Manual

Table of Contents

Introduction *3*
What is C.A.R.?..... 3
Features 3
System Requirements *3*
Minimum System Requirements:..... 3
Install Instructions *4*
How to Use..... *4*
Basics..... 4
Login Menu..... 4
Select Car Menu 5
User Profile 5
Active Rentals..... 6
History..... 6
Account Details..... 7
Car Selection Settinngs.....7
Credits *8*
Analysis:..... 8
Design: 8
Programming: 8
Graphics: 8
Alpha Testing:..... 8
Beta Testing:..... 8

Introduction

What is C.A.R?

C.A.R. is a car rental service that aims to be the premiere car rental service. It was designed with the user in mind and is meant to be intuitive, useful, and incredibly efficient. Users will be amazed how easy it is to find any car they want and subsequently rent any car they find. Users will be able to create an account through the C.A.R application and can rest assured as their login credentials will be safe because we very much value security.

Thank you for taking a ride with C.A.R!

Features

The features of the C.A.R include(but are not limited to):

1. Creating an account
2. Renting a car
3. Viewing active rentals
4. Search for rentals using filters
5. Logout properly
6. View account details
7. Login with a created account
8. Add a custom profile photo
9. (Admin)Add cars
10. (Admin)Remove cars

System Requirements

Minimum System Requirements:

OS:	Any operating system that supports JRE 1.8 or
CPU:	later Pentium 2 266 MHz processor on Windows
RAM:	and Mac 120 MB
Hard Disk Space:	~160 KB of free disk space

Install Instructions

1. Start computer.
2. Go to https://baylorsailor.github.io/C.A.R./site/ and click “Download Iteration 3.zip” button.
3. Open the folder where the zip file is located.
4. Decompress the zip file to the desired location.
5. Open the *currentApp* folder.
6. Open *target* folder.
7. Double click on “CAR-jar-with-dependencies” application and the game should start.

How to Use C.A.R

Basics

C.A.R. is a car rental service built around the idea of ease of use and making sure the user is
never lost.

The application is used by first creating an account. To do so a user will need an email, user name, first name, last name, card number, and a password that has at least one special character
and one capital letter. You can even add an optional profile photo!

Once an account is created, users can do various actions as listed above. The primary use of this application is to rent cars and this can be done by searching for cars that match the defined filters set by the user. If the user finds a car that matches their liking they will then have the option of
renting the car.

Login Screen

The login screen is where the user can login using a previously created account or they will need to create one.

Figure 1: Login Menu screen

Create Account

The create account screen allows for a new user to create an account that can be used to login into the system and use the application in its entirety. New users will need to specify their fullname, email, credit card, card number, and a password. The password should contain an uppercase character as well as a number and a special character. Users will also have the option of uploading an image to be used as their profile image, this is not mandatory.

View Local Listings (Home Screen)

This is the primary screen in which users will more than likely spend most of their time. On this screen, users can apply filters to search for a car that will match their specifications. Filters can be applied via the drop-down menu allowing users the option of searching using different filter options. From this screen users can navigate to the following screens History, Active Rentals, and Account Details.

Figure 3: View History

View History

On this page users can view their previously rented cars. Allowing users to maintain a form of history in case they ever need something for their personal records and book keeping, this will allow them to do just that.

View Active Rentals

This page will allow users to view their currently rented cars. This is to allows users access to the knowledge that cars are rented under their name and this hopefully let them not make mistakes and rent more cars than intended.

Account Details
This is the account details pane. From here users can see everything about their account and can change their card number and type. They can also change their password.

Alpha Testing:
Andrew Case
Maggie Burton
Mark Du
Matthew Darby
Stevie Damrel
Weston Straw
Cedric Boston
Anthony Solorio

Beta Testing:
Andrew Case
Maggie Burton
Mark Du
Matthew Darby
Stevie Damrel
Weston Straw

Design Patterns

1. **Abstract Factory** – Different user types (Administrator, Representative, User) are implemented via factory classes. The factory classes implement the AbstractFactory interface.
2. **Singleton -** The factories are implemented with single instances (static variable of the instance)
3. **Adapter** – Database adapter interacts with the back end data in the files. Also handles requests.
4. **Flyweight** – Used in our factories. Uses a hash map that stores references to users that have already been created. Returns the reference if the user exists, otherwise it makes a new user.
5. **MVC (Model View Controller) -** Our model represents the user objects that hold information about the various users. Our view is represented through swing windows displaying the visualization for each of the models of the application. Our controllers interact with both the view and model. The controller listens for button clicks and updates the view accordingly.
6. **Command -** GUI Toolkits (Java Swing – interface Action, method actionPerformed (~Execute)
7. **State** – Each car object has various states as active, reserved, rented. If the car is active, it will be displayed in the main menu view, otherwise if it is reserved/ rented, it will not be available to view.

Git Analysis

Software Engineering 1 Group Project

495 commits | 4 branches | 0 releases | 1 environment | 6 contributors

Software Engineering 1 Group Project

495 commits | 4 branches | 0 releases | 1 environment | 6 contributors

Additions and Deletions per week
55.0k
50.0k
45.0k
40.0k
35.0k
30.0k
25.0k
20.0k
15.0k
10.0k
5.0k
0
01/27
02/03
02/10
02/17
02/24
03/03
03/10
03/17
03/24
03/31
04/07
04/14
04/21

April 1, 2019 – May 1, 2019

Period: **1 month**

Overview

1 Active Pull Request

15 Active Issues

1	**0**	**13**	**2**
Merged Pull Request	Proposed Pull Requests	Closed Issues	New Issues

Excluding merges, **8 authors** have pushed **171 commits** to master and **172 commits** to all branches. On master, **187 files** have changed and there have been **60,795 additions** and **1,721 deletions**.

FindBugs Analysis Settings

FindBugs-IDEA: found 0 bugs in 0 class

using FindBugs-IDEA 1.0.1 with Findbugs version 3.0.1

CurrentApp[CurrentApp]

Compile Output Path (IDEA)

/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/src/main/java
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/src/main/resources
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/src/test/java

Sources Dir List (3)

/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/src/main/java
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/src/main/resources
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/src/test/java

Analyzed Classes List (31)

/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/models/CarModel.c
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/models/ActiveRenta
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/models/HistoryMod
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/models/UserModel.
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/models/Administrat
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/models/EditUsersTa
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/adapters/Database.
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/factories/Administra
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/factories/Represent
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/factories/AbstractUs
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/factories/UserFactor
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/main/CAR.class
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/controllers/MainMer
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/controllers/EditUser
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/controllers/MainMer
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/controllers/AccountI
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/controllers/MainMer
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/controllers/AccountI
/Users/matt/Desktop/Code/CSI-3471/Project/C.A.R./CurrentApp/target/classes/controllers/HelpCon

Close

Point Redistribution(Out of 100)

Maggie Burton – 16.6667
Weston Straw - 16.6667
Andrew Case - 16.6667
Matthew Darby - 16.6667
Stevie Damrel - 16.6667
Mark Du - 16.6667

Iteration 3 Timecards

Name	Time Spent
Andrew Case	85h 15m
Matthew Darby	36h 33m
Mark Du	42h 31m
Maggie Burton	31h 45m
Stevie Damrel	31h 30m
Weston Straw	43h 01m

Conclusion

Ascension Technologies' C.A.R. application allows all Users to rent cars and Administrators and Representatives to maintain the data it contains. It was designed using the the Unified Modeling Language, and implemented with the Abstract Factory, Singleton, Adapter, Flyweight, Model View Controller, Command and State design patterns. The code was analyzed with a combination of FindBugs, JUNIT and blackbox testing. It was a collaborative effort, with team members working a combined 270hrs 35min to build the project. Thank you for your interest in our product.