*Requirements Definition*

*Group 4*

**Dan’s Shady Business**

**1. Introduction and Context**

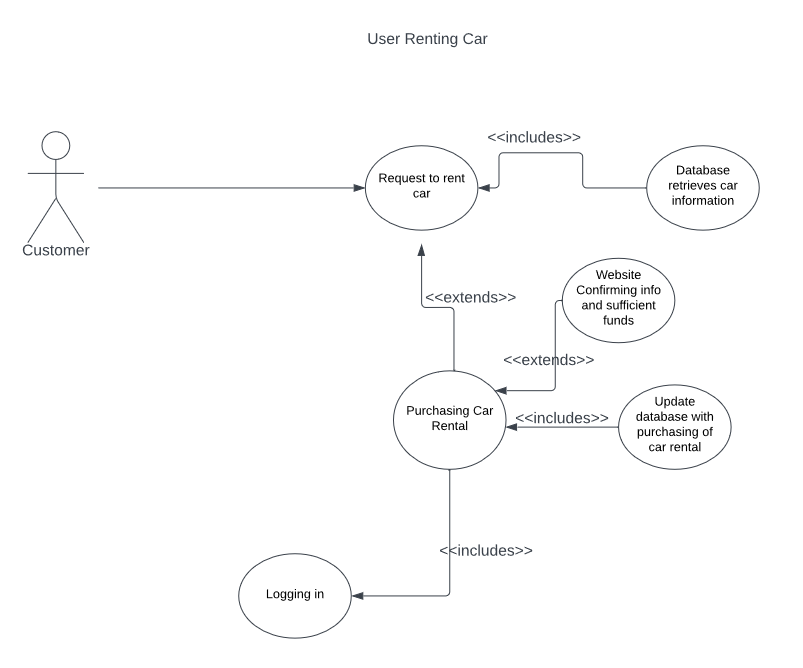
Dan’s Shady Business is a car rental company which allows customers to rent a variety of cars at fixed per day rates. Our software project aims to provide a simple, yet useful set of tools to help customers of Dan’s as well as management and employees to run the business.

The application will allow customers to choose from among a list of available cars that come in three different models. These cars may be rented for multiple days at a time. At the time of reservation a customer must pay the rental fee, and will be prompted to pay a fee for insurance. If the customer refuses to pay the insurance, sometime during the rental period their car will be LoJacked, and an employee will be dispatched to “fix” and replace the rental with another available car. A $300 service fee will be charged in this case.

Managers using our app will be able to pay employees hire new employees from the customer base as well as pay outstanding wages. Employees will be able to approve car checkouts, as well as LoJack cars that have not been returned within the agreed rental period.

**2. Users and Their Goals**

***Case 1 User Renting Car***

******

Participating Actor:

* Customer

Entry Conditions:

* Customer wants to rent car
* Customer has sufficient funds

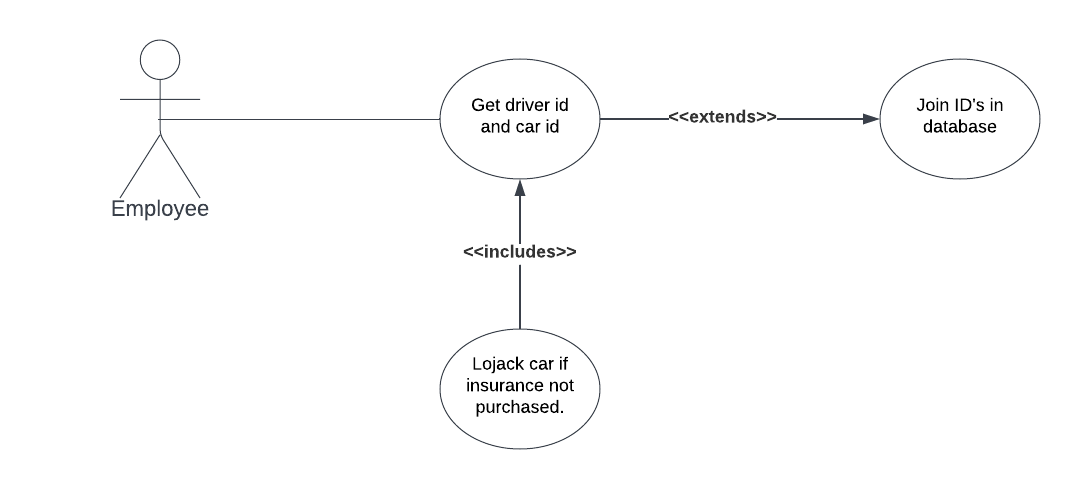
Exit Conditions:

* Customer rents car
* Customer has insufficient funds

Event Flow:

* Customer logs into app
* Customer navigates to rental page
* App queries database for available cars
* Customer chooses car, and rental period
* Customer prompted to pay insurance
* App confirms sufficient funds
* Rental confirmed, and database updated.

***Case 2 Car Pickup***



Participating Actor:

* Employee

Entry Conditions:

* Customer receives keys to car

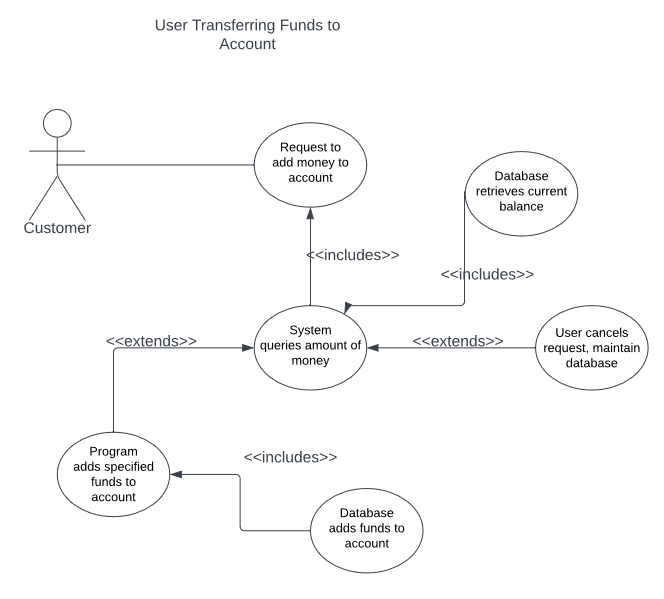
Exit Conditions:

* Transaction completes and customer leaves with car

Event Flow:

* Customer retrieves keys for the car
* Employee either LoJacks the car if insurance not accepted or doesn’t if insurance was purchased
* Employee gets ID of car and matches it to customer in database
* Customer drives off

***Case 3 User Adds Money to Wallet***

******

Participating Actor:

* Customer

Entry Conditions:

* Customer wishes to add funds to account

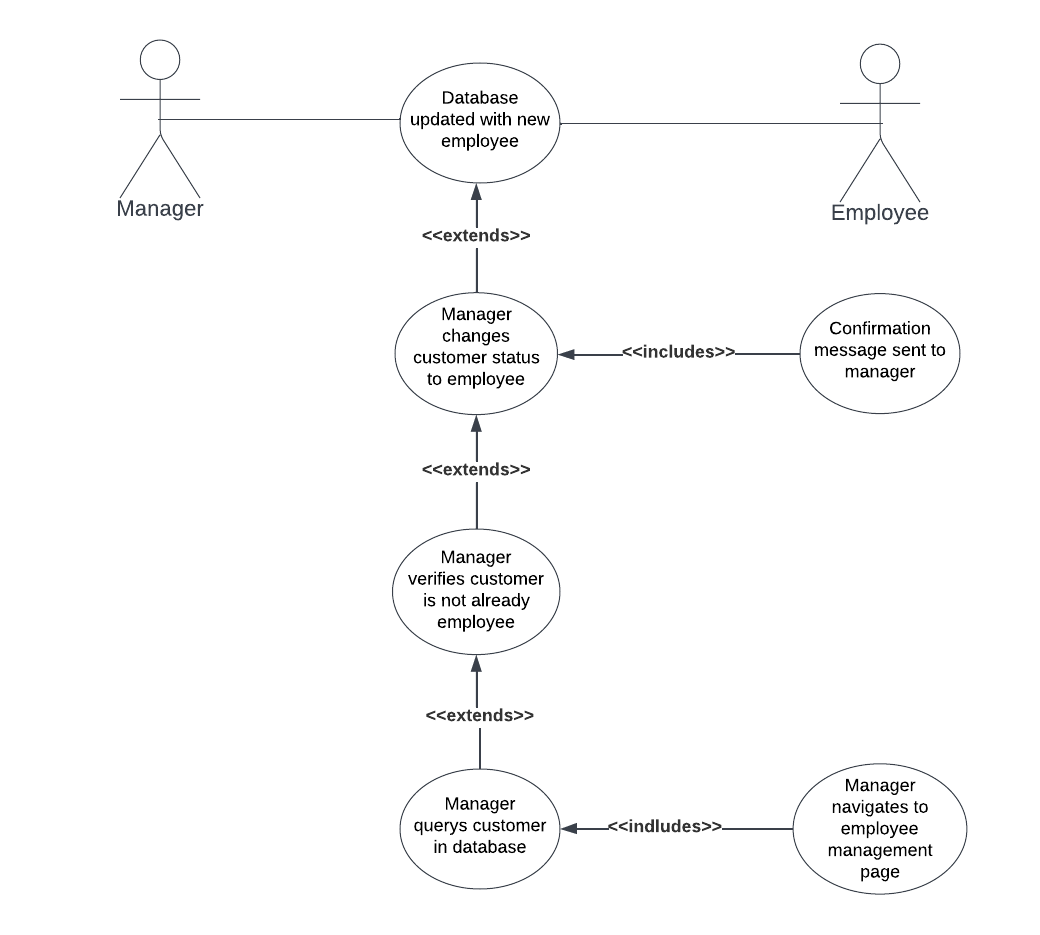
Exit Conditions:

* Customer has added funds to the account
* Customer decides not to add funds

Event Flow:

* Customer logs into app
* Customer navigates to account/funds page
* Customer inputs value to transfer, and submits request
* Database updates with new fund total
* Confirmation of payment is given to customer

***Case 4 Manager Hires New Employee***



Participating Actor:

* Manager
* Customer

Entry Conditions:

* Manager wishes to hire customer as employee
* Customer has account with app

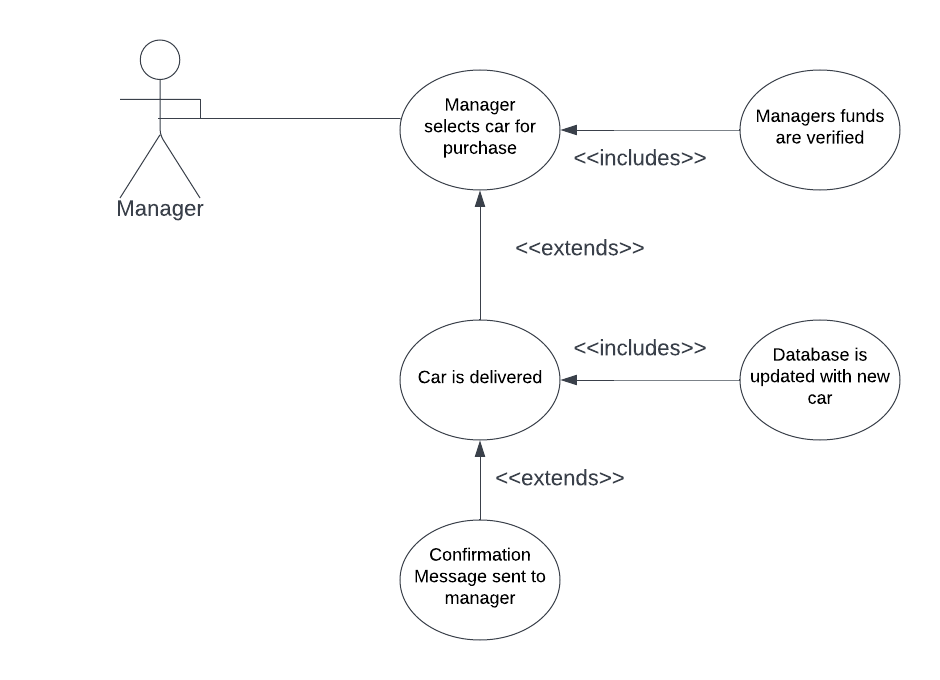
Exit Conditions:

* Customer is now an employee

Event Flow:

* Manager logs into app
* Manager navigates to employee management page
* Manager selects hire new employee option
* Manager presented with list of all customers
* Manager selects customer, and clicks hire
* Database updates the status of the account

***Case 5 Manager Purchases New Car***

******

Participating Actor:

* Manager

Entry Conditions:

* Manager has enough funds to purchase a vehicle
* Manager wants to purchase a vehicle
* Manager needs to purchase a vehicle

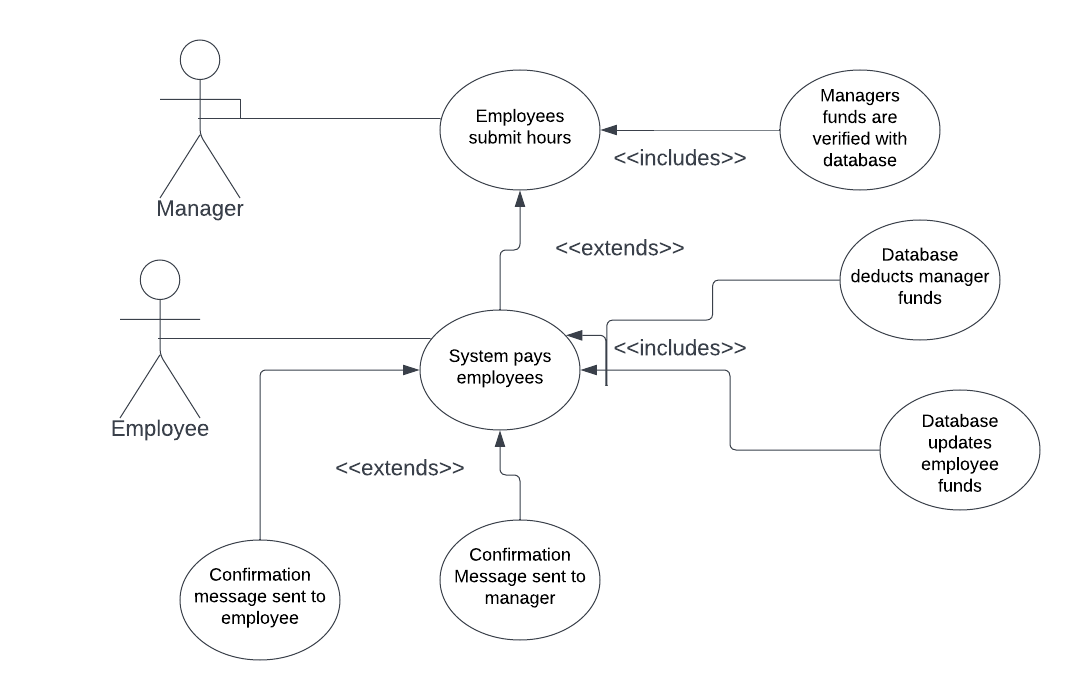
Exit Conditions:

* Manager has purchased a vehicle

Event Flow:

* Manager navigates to vehicle purchase page
* Manager finds vehicle wanted for purchase
* App checks to see if enough funds are in account for purchase
* Manager selects “purchase”
* New car is added to database
* Funds deducted from managers account
* Manager receives a confirmation message from dealer

***Case 6 Manager Pays Employees***

******

Participating Actor:

* Manager
* Employee

Entry Conditions:

* One or more employees has unpaid hours
* Manager has funds in his account
* Manager wants to pay employees

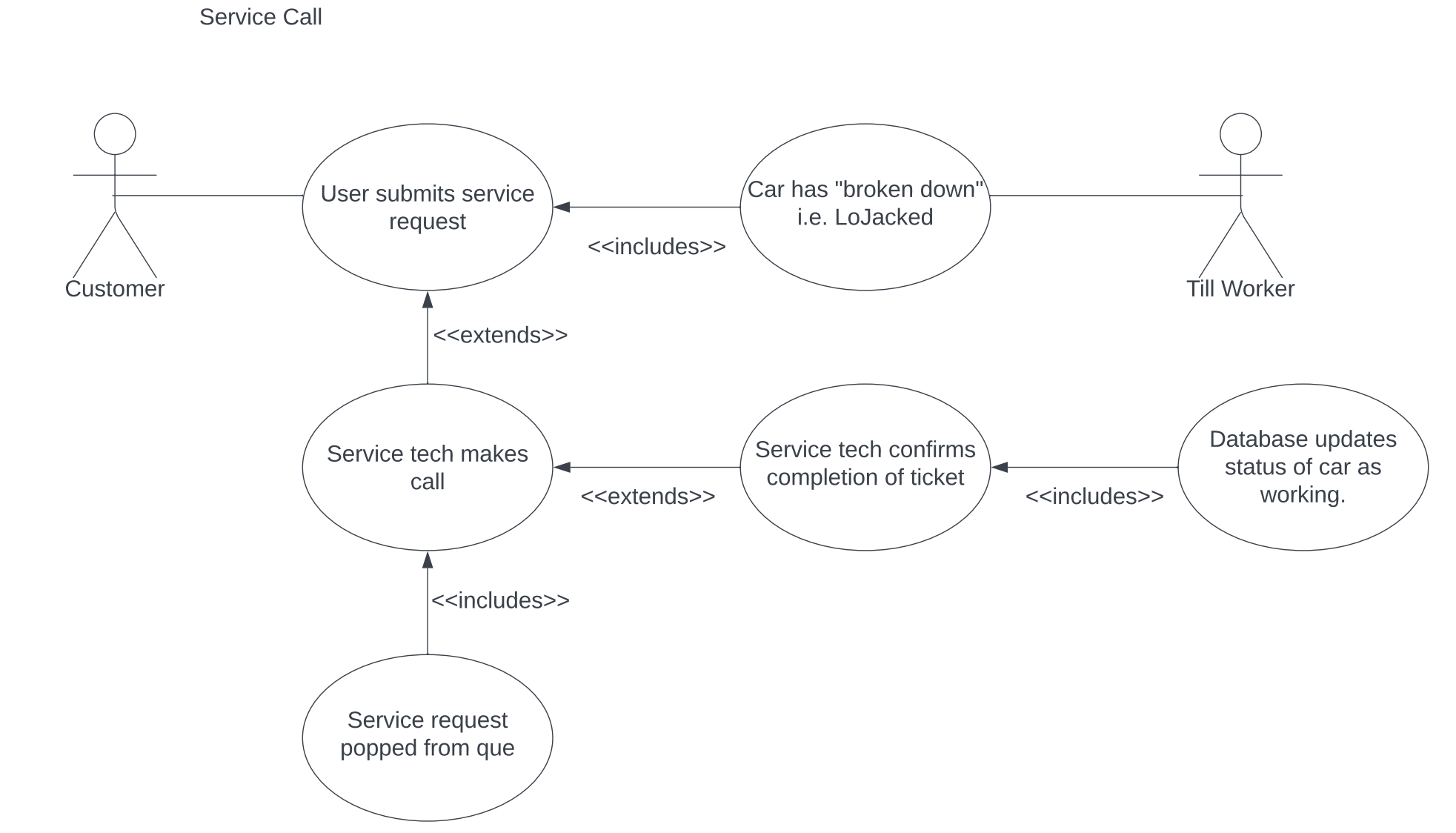
Exit Conditions:

* Manager has paid some or all employees

Event Flow:

* Manager navigates to employee management page
* Manager selects one or more employees with unpaid hours
* Manager selects pay wages button
* App checks if there are enough funds in the managers account to pay the wages
* Wages are added to appropriate employee accounts
* Confirmation of payment is given to the manager.

***Case 7 Service Call***

******

Participating Actor:

* Till worker
* Service worker
* Employee

Entry Conditions:

* Employee declined insurance fee
* Employee has Lojacked car

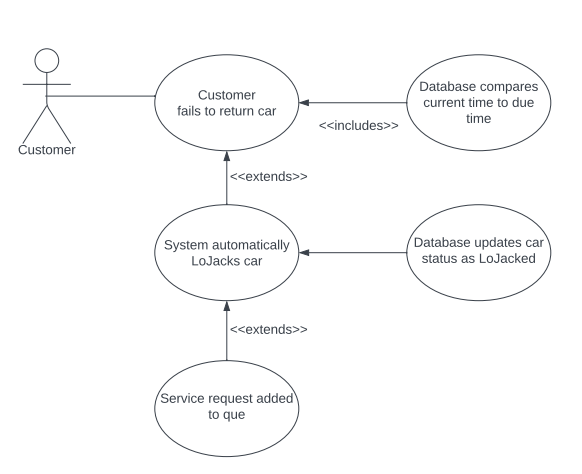
Exit Conditions:

* Service worker has replaced car and submitted confirmation

Event Flow:

* Customer submits service call request
* Request is added to a queue
* Service worker accepts request
* Service worker replaces car with new one
* Service worker confirms completion on the app

***Case 8 Overdue Car is LoJacked***

******

Participating Actor:

* Customer

Entry Conditions:

* Customer has not returned car by due date

Exit Conditions:

* Car has been remotely deactivated

Event Flow:

* Rented car has not been returned within deadline
* Car is automatically Lojacked
* Repo call ticket is created and added to que

**3. Functional Requirements**

1. User profiles and account security
   1. The app will have unique passwords and usernames for each user. These will be used when logging in to the app.
   2. The app will have three distinct use types, customer, employee, and manager.
      1. Customers will have permission to rent cars, buy insurance, add funds, and request service calls.
      2. Employees will have permission to report hours, LoJack cars, and accept service requests.
      3. Managers will have permission to pay outstanding wages, hire new employees, and buy new cars.
   3. Managers will be able to hire new employees by elevating the account status of a customer to the employee level.
2. Car inventory
   1. The app will have a database of the cars currently under the ownership of Dans Shady Business.
      1. The cars will either be reserved, or free to reserve. The period of reservation will be at minimum one day, and at maximum one week.
      2. The app database will keep track of whether a car is currently insured, and whether it is currently LoJacked.
   2. Cars will come in one of three types of vehicles, these will cost $100, $50, $10 per day of rental.
      1. Insurance may be paid for any of the car types and is a flat $50 fee.
   3. The manager has the ability to purchase new cars to be added to the inventory.
      1. The money for this comes out of the manager's/business account.
3. Fund wallet
   1. Each account will have a wallet that holds funds associated with it.(adding funds)
      1. Customers may opt to add money into their respective wallets.
      2. Employees will have money added to their accounts when the manager pays their reported hours.
      3. The manager has access to the funds derived from rentals, insurance, and late fees.
4. Employee management
   1. Managers may hire employees by changing the account type of customers to have employee access.
   2. Employees will have an hourly wage associated with their accounts
   3. Employees can submit their hours to the manager.
   4. The manager has the ability to pay all of the employees outstanding wages or the wages of one employee at a time.
5. LoJack
   1. Manual LoJacking can be done by the employees or manager.
   2. Automatic LoJacking will take place when a car is rented past the return date.

**4. Non-functional Requirements**

1. This application will use the Django framework
2. We will be building this app with the theme of a car rental company that rents its cars primarily to criminals i.e. getaway cars.
3. The version control for this app will be accomplished using git and github.
4. The backend of this app will be written in python.
5. This application will use the database provided by the django python framework.
6. Our team will be recording, in a clear and neat manner, the entire design/development process of the app.
7. Barring a few exceptions this app will provide only the features listed above, in order to prevent feature bloat.

**5. Future Features**

This is a list of possible features that may be looked into if we determine the launch our current app to be a success

* Inter employee messaging or chat functions
* Gps tracking of service workers during service calls
* Profile pictures for employees and customers
* One time insurance policies
* Variable rates based upon car availability

**Glossary**

* Django - A high level Python web framework.
* Git - Open source version control software.
* Github - A cloud based git repository.
* LoJack - A stolen vehicle recovery system.