FINAL REPORT

Analysis of the *Auto-Repair Shop* Project

Srikar Namburi
Emily Hart
5/21/2021

TABLE OF CONTENTS

Executive Summary

The beginning of this final document starts with an explanation of the initial problem that is the core of this project. After describing the initial problem, this document goes into further depth on the overall design of the solution. It also contains discussions of the front-end, back-end, and the key challenges that arose during the implementation of this project. Lastly, there is an analysis of the strengths and weaknesses of the solution created to solve the initial problem. To supplement the report, a relational schema, ER diagram, and a detailed explanation of the ER diagram are also provided.

Introduction

This document is the final report on the Auto Repair Shop project developed by Srikar Namburi and Emily Hart. The primary purpose of this final report is to address the problem and provide an in-depth analysis of the solution. This document will also discuss the effectiveness of the project towards resolving the initial problem. In the analysis section, the report will provide direct feedback from the project co-creators on the strengths and weaknesses of this project and which potential improvements could have been made.

Problem Description

Many small and local auto repair shops still store their information in physical file cabinets. This storage system is potentially dangerous as handwriting can be illegible, erased/smudged/crumpled, and that paper can be misplaced or destroyed very easily via fire or other natural disasters. Data loss could lead to a loss in customers and drastically affect the financial stability of the local auto repair shop. Our solution to this problem is to provide a simple and efficient way to track an auto repair shop's various data points. This project gives local auto shops a safe way of storing their data and a much faster way of providing data and data analytics on their business.

Ultimately, the team's goal was to provide an efficient way for managers of small auto repair shops to handle large amounts of data so that they could focus on the more important/relevant tasks in their business. This application would be beneficial across the nation as it would be free and easy to use.

1	Manager Login	Provides a detailed view of the shop's data and allows for all CRUD operations.
2	Employee Login	Provides a view of all relevant data for an employee and allows for updates to be made.
3	Customer Login	Provides a read-only view of all relevant data for a customer.

Solution Description

The solution decided upon by the Team was to create a Java Application that uses Java and Microsoft SQL Server v15 to maintain all Manager, Customer, Employee, and relevant relationship Data.

Front End Discussion

The front-end of the application was developed primarily with Java Swing. The graphical user interface (GUI) was created with three different users in mind: Manager, Employee, and Customer. First, a Login page is created, which allows the user to input a username and password. If the user did not yet have a username and password, they could click on the register button, which allowed them to input their first name, last name, desired username, password, and user type (manager, employee, or customer), and register for application use.

Once logged in, the user then had access to a customized page depending on their user type.
A manager has access to read, update, and delete all data stored in the database, except for removing other managers. The manager can also assign tasks for an employee to work on and specify which repair the task should be done for.
An employee has access to the tasks that they need to complete, which repairs those tasks are on, and the relevant vehicles. The employee can mark a task as complete, and once complete, the task will no longer show up on the employee's task page.
A customer can view which repairs they have paid for, and what vehicles the repairs were on. The customer is unable to alter data in any way.

Each of the pages available was created with a combination of the JTextField, JButton, JLabel, JComboBox, JToolbar, JMenu, and JMenuItem components from Java Swing.

Back End Discussion

The GUI was run with the AutoRepairShop_S2G2 database. These tables focused on the entities like User, Customer, Employee, Insurance, manager, Part, Repair, Task, Repair, Vehicle, and the relationships that existed between these entities as seen in the entity-relationship diagram(see appendix). A more in-depth description of the database can be found in the Database Design section.

Key Challenges

Challenge: Only two people were able to work on this project.

Solution: This work for this project was divided into primarily front-end development and back-end development, although there was some overlap. Srikar Namburi was primarily responsible for back-end development, and Emily Hart was primarily responsible for front-end development.

Analysis: Overall, this approach worked well. Work was able to be completed at the same time, while one team member worked in SQL Server Management Studio and the other in Eclipse. Towards the end of the project, both team members were working collaboratively through Microsoft Teams to pair-program, test, and debug. All documentation was co-created by both Srikar and Emily.

Challenge: Integrating use of Cascade or Set Null for Foreign Key Constraints

Solution: It was brought to the teams' attentions that running into a "circular queue" is highly probable when using Cascade, so Set Null was chosen as the Foreign Key Constraint.

Analysis: The team learned that there should be a policy of "testing as soon as progress is made." If this was the case, then the error would have been caught quite soon instead of a day before the status report.

Challenge : Both team members lived in different time zones.

Solution: Since this project was completely online, the team was able to collaborate through Microsoft Teams for most communication. They used Outlook Calendar to schedule meetings that both could attend.

Analysis: Living in different time zones provided unique challenges. To schedule meetings that both team members would be available for, it typically resulted in one member working late into the night, or very early in the morning. Overall, using Outlook provided a decent form of keeping track of meeting times since it automatically converted meeting times to the appropriate time zone.

Database Design

Refer to the Entity-Relationship Diagram and Relational Schema, included in the Appendix.

Security Measures - Stored Procedures & User Permissions

In this project, all SQL commands are run through stored procedures. This increases database performance, productivity, and scalability. Most importantly, users would not be able to run SQL injection attacks as all the parameters used are casted as Int, string, or bytes(not as potential SQL code). Another main security issue is on what data users can see. Managers have a wide reach but are only able to change Customers, Employees, but not affect relationships between other managers and customers/employees. Also, Managers are not able to delete/update/add another manager. Customers and Employees are restricted in that they can only view certain data and use only certain stored procedures that further restrict their access.

Integrity Constraints

The following is a list of the domain integrity constraints that exist in the system.

- Users UserType can only be Employee, Manager, and Customer
- StartDate must be before EndDate.
- Repair Discount must be between 1 and 100.
- Repair Price Completed must be either 0 or 1.
- Vehicle Year must be a valid year.
- VIN must be 17 characters.
- TaskID and RepairID increment automatically.
- Insurance Claim Number must be 12 character and PolicyNumber is 11 characters.
- Customer NumberOfVisits cannot be negative.

Stored Procedures

Insert Stored Procedures

InsertCustomer	Takes parameters username, first name, last name, and number of visits, and inserts the data into the customer table.
InsertEmployee	Takes parameters username, first name and last name, and inserts the data into the employee table.
InsertFor	Takes parameters part number and task id, and inserts the data into the for table, which represents which parts are needed for which tasks.
InsertGets	Takes parameters VIN and repair id, and inserts the data into the gets table, which represents which vehicle the repair is for.
InsertHas	Takes parameters repair id, VIN, and task id, and inserts the data into the has table, which represents which tasks are needed for a given repair.
InsertInsurance	Takes parameters claim number, policy number, and deductible, and inserts the data into the insurance table.
InsertOrders	Takes parameters manager username and part number, and inserts the data into the orders table, which represents which parts were ordered by which manager.
InsertPaidBy	Takes parameters VIN, repair id, customer username, and receipt, and inserts the data into the paidBy table, which represents which repairs a customer has paid for.
InsertPaidFor	Takes parameters VIN, repair id, insurance claim number, and receipt, and inserts the data into the paidFor table, which represents which repairs an insurance policy has paid for.
InsertPart	Takes parameters part number, name, and price, and inserts the data into the parts table.

InsertRepair	Takes parameters start date, end date, description, discount, and total cost, and inserts the data into the repair table.
InsertTask	Takes parameters name, description, and price, and inserts the data into the task table.
InsertVehicle	Takes parameters VIN, year, model, mileage, and bodytype, and inserts the data into the vehicle table.

Read Stored Procedures

ReadAssign	Reads which employees are assigned to which tasks.
ReadCustomer	Reads customers.
ReadCustomerView	Reads customers with limited data access.
ReadEmployee	Reads employees.
ReadEmployeeTasks	Reads all tasks for a given employee.
ReadEmployeeView	Reads employees with limited data access.
ReadFinalPaidBy	Reads final customer payment information.
ReadFor	Reads which parts are needed for which task.
ReadGetRepairTaskForEmployee	Reads which tasks an employee is assigned to for a given repair.
ReadGets	Reads which vehicle is for a given repair.
ReadHas	Reads which tasks are for a given repair.
ReadInsurance	Reads all insurance information
ReadManagerCustomerView	Reads the data in the customer view
ReadManagers	Reads all managers usernames
ReadOrders	Reads all information for all Orders issued by Managers for parts
ReadPaidBy	Reads all information in the Customer

	Payments
ReadPaidFor	Reads all information for Insurance Payments
ReadPart	Reads all possible parts in the shop
ReadRepair	Reads all repairs conducted in the shop(completed or not)
ReadTask	Reads all tasks conducted in the shop(completed or not)
ReadVehicle	Reads all vehicles that have entered the shop

Update Stored Procedures

UpdateAssign	Updates employee task assignments.
UpdateCustomer	Updates customer information.
UpdateEmployee	Updates employee information.
UpdateFor	Updates parts needed for a task.
UpdateGets	Updates which vehicle is getting a repair.
UpdateHas	Updates which tasks are for a given repair.
UpdateInsurance	Updates insurance information.
UpdateManager	Updates manager information.
UpdateOrders	Updates which manager ordered which part.
UpdatePaidBy	Updates customer payment information.
UpdatePaidFor	Updates insurance payment information.
UpdatePart	Updates part information.
UpdateRepair	Updates repair information.
UpdateTask	Updates task information.
UpdateVehicle	Updates vehicle information.

Delete Stored Procedures

DeleteAssign	Deletes a given employee task assignment.
DeleteCustomer	Deletes a given customer.
DeleteEmployee	Deletes a given employee.
DeleteFor	Deletes a given for, which represents which parts are needed for which tasks.
DeleteGets	Deletes a given gets, which represents which vehicle the repair is for.
DeleteHas	Deletes a given has, which represents which tasks are needed for a given repair.
DeleteInsurance	Deletes a given insurance policy.
DeleteOrders	Deletes a given orders, which represents which part was ordered by which manager
DeletePaidBy	Deletes a given customer payment.
DeletePaidFor	Deletes a given insurance payment.
DeletePart	Deletes a given part.
DeleteRepair	Deletes a given repair.
DeleteTask	Deletes a given task.
DeleteVehicle	Deletes a given vehicle.

Other Stored Procedures

AssignTask	Allows a manager to assign a task to an employee.
CompleteTask	Marks the given Task as complete.
GetVehicleByVIN	Reads all data from the vehicle table where the VIN number is equal to the given VIN.
LoginChecker	Checks to see if the User correctly inputs the username and password
Register	Registers a User with a specific user type, username, password, first name, last name,

	and with a salted and hashed password

Views

The AutoRepairShop_S2G2 has 3 views.

- o Customer View
- o Employee View
- o Manager View

These views are accessible by the GUI and are made to make viewing all Customers, Managers, and Employees much easier to identify. Before, the Customer, Employee, and Managers could only be viewed by the username which is not helpful as people do not recognize people by usernames only.

Indexes

The only indexes included in this database are the ones only used on primary key values. There are not enough procedures with non-primary key values on which searching is done to justify making non-clustered indexes. Therefore, there are no additional indexes in the AutoRepairShop_S2G2 database than the one automatically generated.

Triggers

There are 3 triggers that exist in this database.

1. PaidStatusPaidFor: Once a user inserts a row in the PaidForthen the trigger automatically uses the Repair ID(found in the inserted table) to set the Repair PriceCompleted to a 1.
2. PaidStatusPaidBy: Once a user inserts a row in the PaidBythen the trigger automatically uses the Repair ID(found in the inserted table) to set the Repair PriceCompleted to a 1.
3. taskRepair: Once a task is switched to as completed, a check is done to see if all tasks for a repair have finished. If this is true, then the trigger automatically switches the Repair Completion to a 1.

Design Analysis

Strengths

- o Appropriate domain integrity constraints are made to ensure accuracy of the data.
- o The System has a high level of security as all SQL commands can only be run through stored procedures.
- o The System limits access to only validated users and gives limited power to each of those users including the Manager User.
- o Most of the stored procedures use only the primary keys so costly overhead is avoided from not having to use secondary indexes.
- o All complex stored procedures have detailed testing, comments to identify what each test is, and a return code for the developers of the team to easily figure out what may have gone wrong.

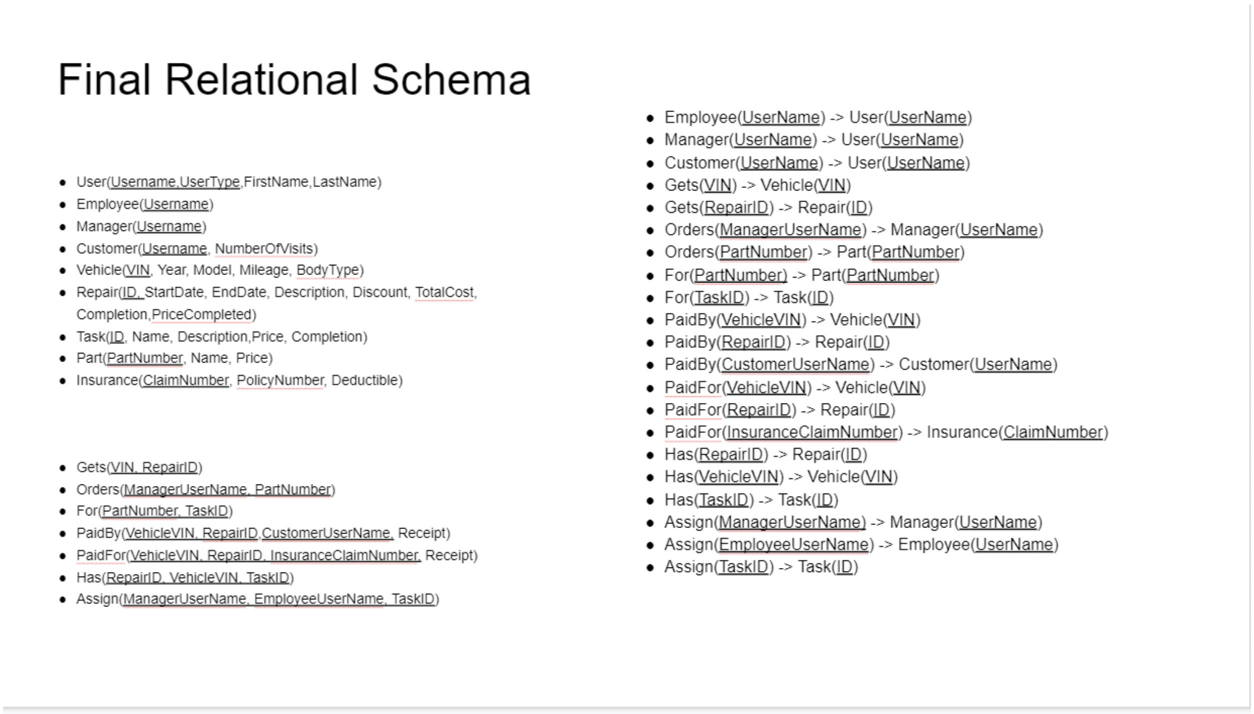
- No dangling tuples are created as when a delete command is run, the stored procedures ensure that all data associated with a key are deleted following a path from relationships to tables.
- Passwords are Salted and Hashed to prevent someone from stealing passwords.
- Attribute names are clear and descriptive.

Weaknesses

- With the concern over security, the program has reduced flexibility in the SQL commands that can be operated. Every time a need for an additional selection is required, the developers of this team must spend some time to create the stored procedure, test the stored procedure, and the incorporate it into the GUI.
- The Users are primarily restricted by the GUI, but the problem remains in that someone could alter the front-end code and be able to access other users quite easily. With additional time, the team would implement user access in the back end as well.

Appendix

Relational Schema



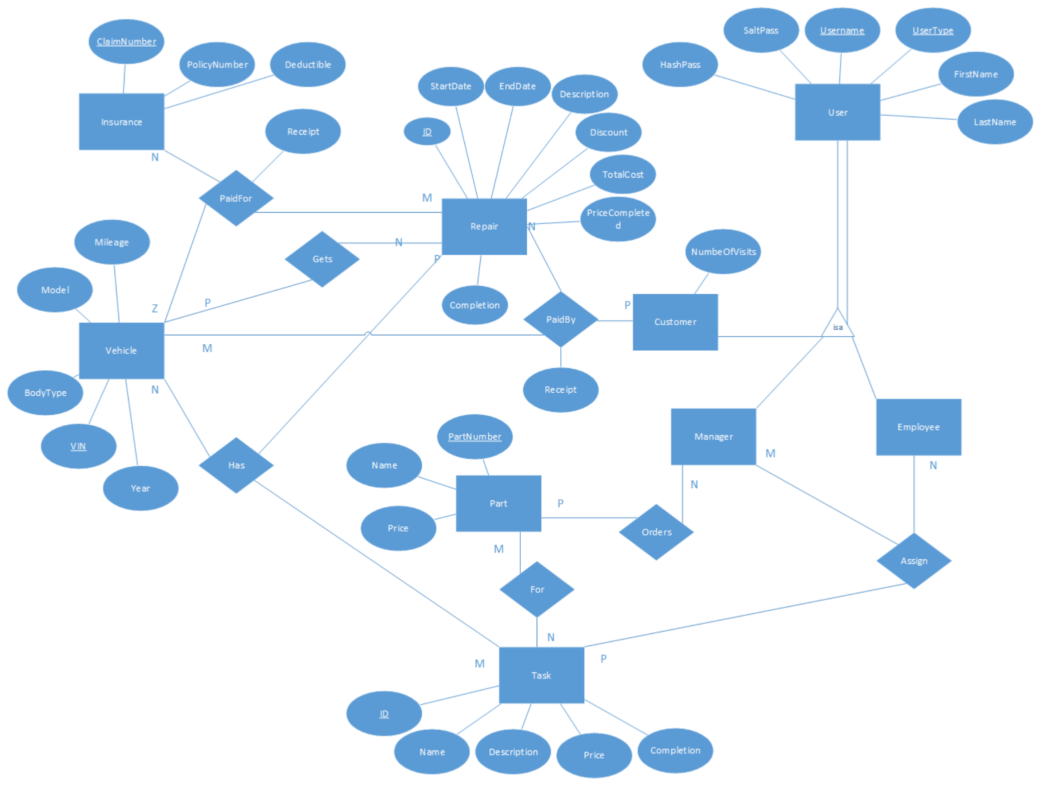
Final Relational Schema

- User(Username,UserType,FirstName,LastName)
- Employee(Username)
- Manager(Username)
- Customer(Username, NumberOfVisits)
- Vehicle(VIN, Year, Model, Mileage, BodyType)
- Repair(ID, StartDate, EndDate, Description, Discount, TotalCost, Completion,PriceCompleted)
- Task(ID, Name, Description,Price, Completion)
- Part(PartNumber, Name, Price)
- Insurance(ClaimNumber, PolicyNumber, Deductible)

- Gets(VIN, RepairID)
- Orders(ManagerUserName, PartNumber)
- For(PartNumber, TaskID)
- PaidBy(VehicleVIN, RepairID,CustomerUserName, Receipt)
- PaidFor(VehicleVIN, RepairID, InsuranceClaimNumber, Receipt)
- Has(RepairID, VehicleVIN, TaskID)
- Assign(ManagerUserName, EmployeeUserName, TaskID)

- Employee(UserName) -> User(UserName)
- Manager(UserName) -> User(UserName)
- Customer(UserName) -> User(UserName)
- Gets(VIN) -> Vehicle(VIN)
- Gets(RepairID) -> Repair(ID)
- Orders(ManagerUserName) -> Manager(UserName)
- Orders(PartNumber) -> Part(PartNumber)
- For(PartNumber) -> Part(PartNumber)
- For(TaskID) -> Task(ID)
- PaidBy(VehicleVIN) -> Vehicle(VIN)
- PaidBy(RepairID) -> Repair(ID)
- PaidBy(CustomerUserName) -> Customer(UserName)
- PaidFor(VehicleVIN) -> Vehicle(VIN)
- PaidFor(RepairID) -> Repair(ID)
- PaidFor(InsuranceClaimNumber) -> Insurance(ClaimNumber)
- Has(RepairID) -> Repair(ID)
- Has(VehicleVIN) -> Vehicle(VIN)
- Has(TaskID) -> Task(ID)
- Assign(ManagerUserName) -> Manager(UserName)
- Assign(EmployeeUserName) -> Employee(UserName)
- Assign(TaskID) -> Task(ID)

Entity Relationship Diagram



Explanation of Entity Relationship Diagram

The ER diagram outlines the relationships between all entities in the application and the attributes each entity has. Every user of the application is either a manager, an employee, or a customer. Records of all Tasks, Parts, and Vehicles needed for a Repair can be stored and accessed. There is Insurance information that allows for insurance payment to be stored.