|  |
| --- |
|  |
| The Cycle Manager |
| CIS591 Analysis Report |

|  |
| --- |
| Glenn Larson  10/24/2011 |

# Contents

[Contents 1](#_Toc307227460)

[System Requirements Models 2](#_Toc307227461)

[1. System Overview By Scope 2](#_Toc307227462)

[2. Functional Requirements 2](#_Toc307227463)

[3. Non Functional Requirements 3](#_Toc307227464)

[4. Use Case Diagrams 3](#_Toc307227465)

[By Actor 4](#_Toc307227466)

[By Managerial System 5](#_Toc307227467)

[By Sales System 6](#_Toc307227468)

[5. Domain Model Class Diagram 7](#_Toc307227469)

[6. Use Case Descriptions 8](#_Toc307227470)

[7. System Sequence Diagrams 23](#_Toc307227471)

[8. Attribute Definitions 27](#_Toc307227472)

[Alternative design Concepts and recommendations 30](#_Toc307227473)

[1. Alternative Design Concepts 30](#_Toc307227474)

[2. 4 Concrete Design Alternatives 30](#_Toc307227475)

[3. My Selected Design Alternative 30](#_Toc307227476)

[Design and implementation to date 31](#_Toc307227477)

[1. Technology, Working Environment, and Architecture 31](#_Toc307227478)

[2. Implemented to Date 32](#_Toc307227479)

[Use Cases 32](#_Toc307227480)

[Screen Shots of Enter Service Ticket 33](#_Toc307227481)

[Source Code 41](#_Toc307227482)

[3. Menu Hiearchy 42](#_Toc307227483)

[4. Reports 43](#_Toc307227484)

# System Requirements Models

1. System Overview By Scope -

The system is a management system for a bike shop that will allow them to enter and manage service tickets, purchases, it will allow them to store information about their customers and their customers bikes. They will be able to add update and delete products, employees and bikes that they sell. They will also be able to view reports that tell them how many services the employees have completed and the bonus they get for a given period. Then they will be able to view how many of each item they have sold to date and during a given period.

1. Functional Requirements – Event List and resulting use case

|  |  |
| --- | --- |
| Event | Use Cases |
| Employee Gets Fired or Quits | Delete Employee |
| Employee Information Changes | update Employee |
| New Employee Hired By Store | Create Employee |
| Employee Shows Up to Work | Employee Clocks In |
| Employee Leaving Work | Employee Clocks out |
| Customer Wants To make Purchase or Drip Bike Off | Customer Looked Up |
| New Customer wants to make purchase or drop bike off | Create New Customer |
| Customer Brings Bike In for Service | Enter Service Ticket |
| Employee Creates new service | Create New Service item |
| Information About Service Changes | Update Service item |
| Service No Longer Offerd | Delete Service Item |
| Application Start | Load service Queue |
| Shop Recieves a new Item that isn't in system | Create Product |
| Employee Updates Product Item | Update Prodcut Item |
| New Bike Not in system already | Cerate New Bike Item |
| Employees Checks Out Customer | Check Out Cutstomer |
| Complete Bike Service | Update Bike Service Status |
| Employe Wants to change Password | Change Password |
| Manager Wants Employee Service Report | Employee Service Report |
| Manager Wants Item Sales Report | Item Sales Report |

1. Non Functional Requirements -

The system needs to be easily usable by any employees no matter their skill level. It needs to be straight forward and easy that won’t require a lot of training. The system also needs to be 100% reliable because if the system goes down then it could hurt their business for the time it is down. Also if they where to lose their database then that could cause issues as they would lose everything so there needs to be a way to back up the database. The system needs to perform reasonably well, that means that it can’t have long pauses between user input and the display of data retrieved from the database. For security the system needs to be secure. A bunch of things need to be encrypted such as credit card numbers, and then employee passwords. Also the system needs to inaccessible from the outside of the system so that unwanted people can’t get in and steal information.

1. Use Case Diagrams -on following 3 pages

By Actor 

By Managerial System 

### By Sales System



1. Domain Model Class Diagram. There is currently only 9 classes but another one will be added before completion.



1. Use Case Descriptions – Half are fully developed half are simple.

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Update Employee | |
| Scenario: | Update Employees Information | |
| Triggering Event: | The employee has a change in their personal information | |
| Brief Description: | The Employee Has a change of name, address, or phone number so they update their information in the system | |
| Actors: | Employee | |
| Stakeholders: | Employees, and Employee | |
| Preconditions: | Employee Needs to be in the system | |
| Postconditions: | Employees Data Must Be updated. | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Delete Employee | |
| Scenario: | Employee gets deleted | |
| Triggering Event: | The employee Gets Fired Or Quits | |
| Brief Description: | The Employee Needs to be deleted out of the system when they quite or get fired. | |
| Actors: | Employee | |
| Stakeholders: | Employee, Managers | |
| Preconditions: | Employee Needs to be in the System | |
| Postconditions: | Employee Must No longer Exist  Employee Credentials must no longer exist | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Create Employee | |
| Scenario: | Hire New Employee | |
| Triggering Event: | A New Employee Gets Hired By the Company | |
| Brief Description: | When the company hires a new employee they enter the employees information into the system and then the system generates a starting password | |
| Actors: | Employee | |
| Stakeholders: | Managers, New Hire, Employees | |
| Preconditions: | None | |
| Postconditions: | Employee Must Be Created  Credentials must be created | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Create Customer | |
| Scenario: | A New Customer Wants to make Purchase | |
| Triggering Event: | Customer Not in System | |
| Brief Description: | A New Customer Gets Created when they come in and make their first Purchase. | |
| Actors: | Employee | |
| Stakeholders: | Customer, Employee | |
| Preconditions: | None | |
| Postconditions: | Customer Must Exist In system | |
| Flow of Activities: | Actor | System |
| 1.Employee Will Click create new customer Button  2.The Employee Will enter Customers Information into Form  3.The Employee Will Click Create Customer | 1.1. The Create New Customer Screen Will Appear  3.1The system Will Create New Employee object and store in database  3.2 The system will display msg stating weather customer was created or not.  3.3 The system will carry Customer Information back to main screen and populate text boxes |
| Exception Conditions: | 1.If All text Boxes are filled then the system will tell employee to check information and then continue.  2. If data entered doesn’t match data type the system will ask employee to check information | |
|  |  | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Lookup Customer | |
| Scenario: | Find Customer Based On First and Last Name | |
| Triggering Event: | Customer Comes in to Make Purchase or drop off bike | |
| Brief Description: | The customer comes in to make a purchase or drop off bike | |
| Actors: | Employee | |
| Stakeholders: | Customer, Employee | |
| Preconditions: | Customer Must Exist In System | |
| Postconditions: | Customers Name, Address City State Zip and phone number will populate customer text boxes. | |
| Flow of Activities: | Actor | System |
| 1.Enters Customers First Name And Last Name  2.Employee Clicks Find Customer  3.Employee Verifies Information With Customer | 2.1 Gathers Customer Information  2.2 The system Populates the Customer Text boxes With Customer Information |
| Exception Conditions: | 1.If Customer Does Not Exists The employee will create New Customer  2.If Employee Only Enters First Name Or Last Name the System Will ask the employee to enter full name.  3.System Will not Allow numbers to be entered into text boxes.  4.If boxes are left empty the system will say no name entered and ask employee to enter name | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Create Product | |
| Scenario: | Create New Product Item | |
| Triggering Event: | Shop receives a new item that isn’t in computer | |
| Brief Description: | When the shop receives a new product | |
| Actors: | Employee | |
| Stakeholders: | Employee Customer Owner(s) | |
| Preconditions: | None | |
| Postconditions: | Product Must Exist In system  Product Must be found when products are being searched | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Update Product | |
| Scenario: | Product Info needs to be updated | |
| Triggering Event: | Something with the product | |
| Brief Description: | There is a change in some sort of information for a product such as price. A employee goes in and updates the product information | |
| Actors: | Employee | |
| Stakeholders: | Employee Customer Owner(s) | |
| Preconditions: | Product Must Exist | |
| Postconditions: | Product Must Be updated  Any Time Product IS loaded except for previous purchases the product must reflect Change | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Delete Product | |
| Scenario: | Product Gets Deleted From System | |
| Triggering Event: | Store no longer Carries product | |
| Brief Description: | A product is no longer carried by store so they delete the product out of the system | |
| Actors: | Employee | |
| Stakeholders: | Employee, Owner(s) | |
| Preconditions: | Product Must Exist In System | |
| Postconditions: |  | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Create Service | |
| Scenario: | A new Service Gets created | |
| Triggering Event: | A new Service gets Developed | |
| Brief Description: | When a new service needs to be created to handle the change in cycling technology. | |
| Actors: | Employee | |
| Stakeholders: | Employees, customers, owner(s) | |
| Preconditions: |  | |
| Postconditions: |  | |
| Flow of Activities: | Actor | System |
| 1.Employee Clicks on Services Admin in the Administration Menu  2.The Employee will fill in appropriate information  3.the employee will click create Service  4. Employee will repeat till done  5. Employee clicks close button | 1.1. The Service Admin Window Will appear  3.1The system will create service and send a response msg saying that service was created  3.2 The Service Admin Window will refresh to show the new service in the service list.  5.1 The Service Admin Screen will close |
| Exception Conditions: | 1.The service description will only hold 255 characters  2.If any required information is not filled in then system will ask employee to fill in the rest of the service information. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Update Service | |
| Scenario: | Service Information Needs To be Updated | |
| Triggering Event: |  | |
| Brief Description: | When some form of information for a service needs to updated such as price or description | |
| Actors: | Employee | |
| Stakeholders: | Employee, Owner(s) | |
| Preconditions: | Service Must Exist | |
| Postconditions: | Service Needs To reflect any changes from this point on. | |
| Flow of Activities: | Actor | System |
| 1.Employee Clicks Service Administration menu item  2.Employee Selects Service to update from list  3. Employee clicks update button | 1.1. Service Administration window launches.  2.1Text boxes get populated with service information  3.1 system updates service  3.2 Msg stating service was updated successfully  3.3 List gets refreshed to reflect changes made. |
| Exception Conditions: | 1.If service can’t be updated msg saying service can’t be updated will appear | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Delete Service | |
| Scenario: | Service needs to be deleted | |
| Triggering Event: | Service No longer offered by shop | |
| Brief Description: | The service is no longer offered by shop so it needs to be deleted out of system | |
| Actors: | Employee | |
| Stakeholders: | Employee, Owner(s) | |
| Preconditions: | Service Needs To exist | |
| Postconditions: | Service shouldn’t exist anymore | |
| Flow of Activities: | Actor | System |
| 1.Employee clicks the service administration menu item  2.Employee selects Service From list  3.Employee clicks Delete button | * 1. Service Administration window is displayed   2. Service list gets populated   2.1.text boxes get populated with service information  3.1 system updates service  3.2 Msg stating service was deleted successfully  3.3 List gets refreshed to reflect the deletion |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Create Customer Bike | |
| Scenario: | Customer Brings Bike in for service | |
| Triggering Event: | Create Service Ticket | |
| Brief Description: | A Customer Brings in a new bike for service that isn’t already in the system. | |
| Actors: | Employee | |
| Stakeholders: | Employee Customer | |
| Preconditions: | Must Be Entereing Service Ticket | |
| Postconditions: | New Bike Must Show in Select Bike Screen | |
| Flow of Activities: | Actor | System |
| 1.Employee will select create new customer button from select customer bike screen  2.Employee will enter appropriate information  3.Employee Will click create | 1.1. Create New Customer Bike screen will appear  3.1. The customer bike will be stored  3.2 the form will close  3.3 The select bike form will refresh bike list. |
| Exception Conditions: | 1.If any of the text boxes are left empty the system will request employee to enter appropriate information | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Enter Service ticket | |
| Scenario: | Bike needs to be Entered For service | |
| Triggering Event: | Customer brings Bike in For service | |
| Brief Description: | When a customer brings bike in for service an employee needs to entere it into the system and choose the services the customer wants | |
| Actors: | Employee | |
| Stakeholders: | Employee Customer | |
| Preconditions: | Customer Must Exist  Employee Needs To be Clocked In | |
| Postconditions: | Bike Service Needs To be Added to service Queue. | |
| Flow of Activities: | Actor | System |
| 1.Customer Brings Bike In For Service  2.Employee Enters Customer Information(Uses Lookup Customer)  3.The Employee Then Selects the enter Bike For Service Button  4. The Employee Will Verify The Customer Information on screen  5.The Employee Will click the select bike button  6 If Customer Bike no listed the employee will click create new bike button(use create New Customer bike use case)  7. Employee will select bike from list(use Select Customer Bike Use Case)  8. The employee click add service button  9 the employee will select services that are wanted  10. the employee will enter description  11. The employee saves service ticket | 2.1 Customer Information Loaded  3.1 The System Loads the enter Bike for Service Form  5.1The select Bike Screen will show with a list of the customers bikes.  7.1.a the bike information will populate the bike service form  8.1 The add service screen will appear  9.1 The services will populate the selected services list |
| Exception Conditions: | 1.Customer Must be loaded or it will ask employee to lookup customer  2. Once Bike Service Screen Shows it will not allow it to be saved until a bike is selected and a service is selected. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Select Customer Bike | |
| Scenario: | Customer Brings Bike in for Service | |
| Triggering Event: | Enter Bike For Service Use Case | |
| Brief Description: | When Enter Bike For service The Employee Will have to choose a bike from the list of the Customers Bikes in the System | |
| Actors: | Employee | |
| Stakeholders: | Employee Customer | |
| Preconditions: | Customer Must Be Entered  Service Ticket Must Be being Created | |
| Postconditions: | Bike information must populate the service ticket screen | |
| Flow of Activities: | Actor | System |
| 1.Employee Will be entereing Service ticket  2.Employee will click the select bike button  3.the Employee will choose the bike that matches the one the customer brought | 1.1. Service ticket screen will be visible  2.1The select bike screen will appear  2.2 A list of the bikes associated with that customer will load.  3.1The bike information will populate the Service Ticket entering screen. |
| Exception Conditions: | 1.If bike doesn’t exist then employee will choose enter new customer bike button.  2. If no bike selected it will ask customer to select bike. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Load Bike Service Queue | |
| Scenario: | Load The Service Queue | |
| Triggering Event: | Load Of Program | |
| Brief Description: | At Start Of Program The Bike Service Queues Need to Be Populated | |
| Actors: | system | |
| Stakeholders: | Employee | |
| Preconditions: |  | |
| Postconditions: |  | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Create New Bike | |
| Scenario: | New Bike Needs To Be Created | |
| Triggering Event: | The shop gets a new bike not in the system | |
| Brief Description: | The employee will create a new bike item when they get a new bike that isn’t in the system | |
| Actors: | Employee | |
| Stakeholders: | Employee, Owner(s) | |
| Preconditions: |  | |
| Postconditions: |  | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Delete New Bike | |
| Scenario: | New Bike Needs To be Deleted | |
| Triggering Event: | The shop stops carrying new bike model | |
| Brief Description: | An employee needs to delete an new bike model because the shop no longer carries it. | |
| Actors: | Employee | |
| Stakeholders: | Employee, owner(s) | |
| Preconditions: |  | |
| Postconditions: |  | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Update new Bike | |
| Scenario: | New Bike Needs to be Updated | |
| Triggering Event: | Change in something related to new bike | |
| Brief Description: | A new Bike needs to be updated because something has changed such as price | |
| Actors: | Employee | |
| Stakeholders: | Employee, owner(s) | |
| Preconditions: |  | |
| Postconditions: |  | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Check Out Customer | |
| Scenario: | Customer wants to buy something | |
| Triggering Event: | Customer requests to buy something | |
| Brief Description: | The customer requests to purchase an item and the employee will enter it into the system and check the customer out. | |
| Actors: | Employee | |
| Stakeholders: | Employee and Customer | |
| Preconditions: |  | |
| Postconditions: |  | |
| Flow of Activities: | Actor | System |
| 1.  2.  3. | 1.1.  2.1  3.1 |
| Exception Conditions: | 1. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Employee Clock in | |
| Scenario: | Employee Clocks into system | |
| Triggering Event: | Employee Clocks in | |
| Brief Description: | The employee will clock into the system using their credintials; | |
| Actors: | Employee | |
| Stakeholders: | Employee | |
| Preconditions: |  | |
| Postconditions: |  | |
| Flow of Activities: | Actor | System |
| 1.Employee Will click on the employee Clock In Menu Item  2.Employee will enter employee number and password  3.Employee will click Clock in button | 1.1. Will display clock in screen  3.1 System will create credential object  3.2 will retrieve retrieve credintials linked to employee  3.3 Will compare password store in system with that of what employee entered  3.4 Will add employee to Working employee list. |
| Exception Conditions: | 1.If password incorrect message saying password is incorrect will appear.  2.If no password or employee number entered msg will be displayed | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Employee Clock Out | |
| Scenario: | Employee Clocks out of system | |
| Triggering Event: | Employee Clocks out of system | |
| Brief Description: | The employee will clock out of system using their credentials | |
| Actors: | Employee | |
| Stakeholders: | Employee | |
| Preconditions: | Employee Must Exist  Employee Must Have Credentials | |
| Postconditions: | Employee Should no Longer appear in Working employee list | |
| Flow of Activities: | Actor | System |
| 1.Employee Will click on the employee Clock Out Menu Item  2.Employee will enter employee number and password  3.Employee will click Clock out button | 1.1. Will display clock out screen  3.1 System will create credential object  3.2 will retrieve retrieve credintials linked to employee  3.3 Will compare password store in system with that of what employee entered  3.4 Will remove employee from Working employee list. |
| Exception Conditions: | 1.If password incorrect message saying password is incorrect will appear.  2.If no password or employee number entered msg will be displayed | |

1. System Sequence Diagrams- For all fully developed use cases



1. Attribute Definitions

|  |  |  |  |
| --- | --- | --- | --- |
| Credentail Class |  |  |  |
| EmpID | employee ID | int |  |
| Password | Employee Entered Password | string |  |
| NewPassword | Employee New Password | string |  |
| confirmPass | Employee Entered Confirm New Password | string |  |
| passCompare | Password Pulled from database to compare to | string |  |
| loginPass | if password was good | boolean |  |
|  |  |  |  |
| Employee Class |  |  |  |
| employeeID | employee ID | int |  |
| employeeFirstName | Employee First Name | string |  |
| employeeLastName | Employee Last Name | string |  |
| employeeAddress1 | Employee Address Line 1 | string |  |
| employeeAddress2 | Employee Address Line 2 | string |  |
| employeeCity | Employee City | string |  |
| employeeState | Employee State | string | 2 characters |
| employeeZip | Emmployee Zip Code | int | 5 characters |
| employeePhone | Employee Phone | string | 7 characters |
| employeeAreaCode | Employee Area Code | string | 3 characters |
| emplpyeePayRate | Employee Pay Rate | double |  |
|  |  |  |  |
| Reciept Class |  |  |  |
| aCustomer | A customer Object | customer object |  |
| aEmployee | A Employee Object | employee object |  |
| items | a arraylist of item objects | arraylist |  |
| subtotal | Subtotal or purchase | double |  |
| total | total cost of purchase after tax | double |  |
| paymentType | type of payment(cash or creditcard) | string |  |
| creditCardNumber | credit card number | string |  |
| paymentAmount | the amount of the payment | double |  |
|  |  |  |  |
| Service Class |  |  |  |
| serviceID | ID of a service | int |  |
| serviceName | name of a service | string |  |
| serviceType | type of service | string |  |
| serviceDescrip | description of service | string | 255characters |
| serviceCost | The cost of the service | double |  |
|  |  |  |  |
| BikeService Class |  |  |  |
| bServiceID | id of service ticket | int |  |
| aCustomer | a customer object | customer object |  |
| aCustomerBike | a customer bike object | customer bike object |  |
| bServiceDescription | desciption of service | string | 255 characters |
| bServices | list of service objects customer would like | arraylist of services |  |
| bServiceEnteredBy | name of employee whoo entered service | string |  |
| bServiceEmpNmae | name of service tech | string |  |
| bServiceStatus | status of the service | string |  |
| bCostofServices | total cost of services | double |  |
|  |  |  |  |
| CustomerBike Class |  |  |  |
| cBikeID | id of Customer Bike | int |  |
| cBikeBrand | brand of customer Bike | string |  |
| cBikeModel | Model of customer bike | string |  |
| cBikeSerialNumber | Serial Number of Customer Bike | string |  |
|  |  |  |  |
| New Bike Class |  |  |  |
| nBikeID | Id of A new Bike | int |  |
| nBikeBrand | Brand of new bike | string |  |
| nBikeModel | model of new bike | string |  |
| nBikeSerialNumber | sieral number of new bike | string |  |
| nBikeType | type of new bike | string |  |
| nBikeCost | cost of new bike | double |  |
|  |  |  |  |
| ProductClass |  |  |  |
| prodID | id of product | int |  |
| prodSKU | sku of product | string |  |
| prodName | name of product | string |  |
| prodType | type of prodcut | string |  |
| prodCost | cost of product | double |  |
|  |  |  |  |
| Customer Class |  |  |  |
| customerID | Customer Id | int |  |
| customerFirstName | Customer First Name | string |  |
| customerLastName | Customer Last Name | string |  |
| customerAddress1 | Customer Address Line 1 | string |  |
| customerAddress2 | Customer Address line 2 | string |  |
| customerCity | Customer City | string |  |
| customerState | customer state | string | 2characters |
| customerZip | customer zip | int | 5charcters |
| customerPhone | customer phone | string | 7characters |
| customerAreaCode | customer area code | string | 3 charaters |

# Alternative design Concepts and recommendations

1. Alternative Design Concepts

Some of the ways I could do this differently is I could have broken it down into a module that allows for purchases and then a module that allows for the management of the services. I could have also had it as a web based application. If I made it a web based application I could have done it as a asp.net application or using another language such as java. I could have also implemented it differently by not using as many separate windows and using just one that adapts to the use case that is being executed at the time being.

1. 4 Concrete Design Alternatives

Web Based – This is pretty feasible as it would be simply just converting what I have done and just converting it to asp.net web forms. I would be able to use the same development suite.

Two Separate Applications – I could split the application into two separate applications one that runs on the floor of the store for purchases and then one in the back of the store that is for managing service tickets that have been entered and then any other administrative purposes. They would talk to the same database.

Java Based – I could do it as a java application that either ran in windows or web based. If I used this I would use eclipse as the development environment

Procedural Program – this could have been a procedural program that would just go through and collect all needed information as it needed it

1. My Selected Design Alternative

I would choose a web application as it is the easiest to do and by having it all web based it would allow the store to access some things online. This could also allow for the system to have an online order system.

# Design and implementation to date

1. Technology, Working Environment, and Architecture

The technology I am using visual studios 2010 professional to code in. The language I am using I c#. For the database I am just using a simple access database. The architecture I am using is a three layer design, so there is a presentation layer, a business layer and then a data access layer. The approach I am taking is object oriented design.



1. Implemented to Date

### **Use Cases**

Delete employee

Update employee

Create employee

Employee clocks in

Employee clocks out

Customer looked up

Create new Customer

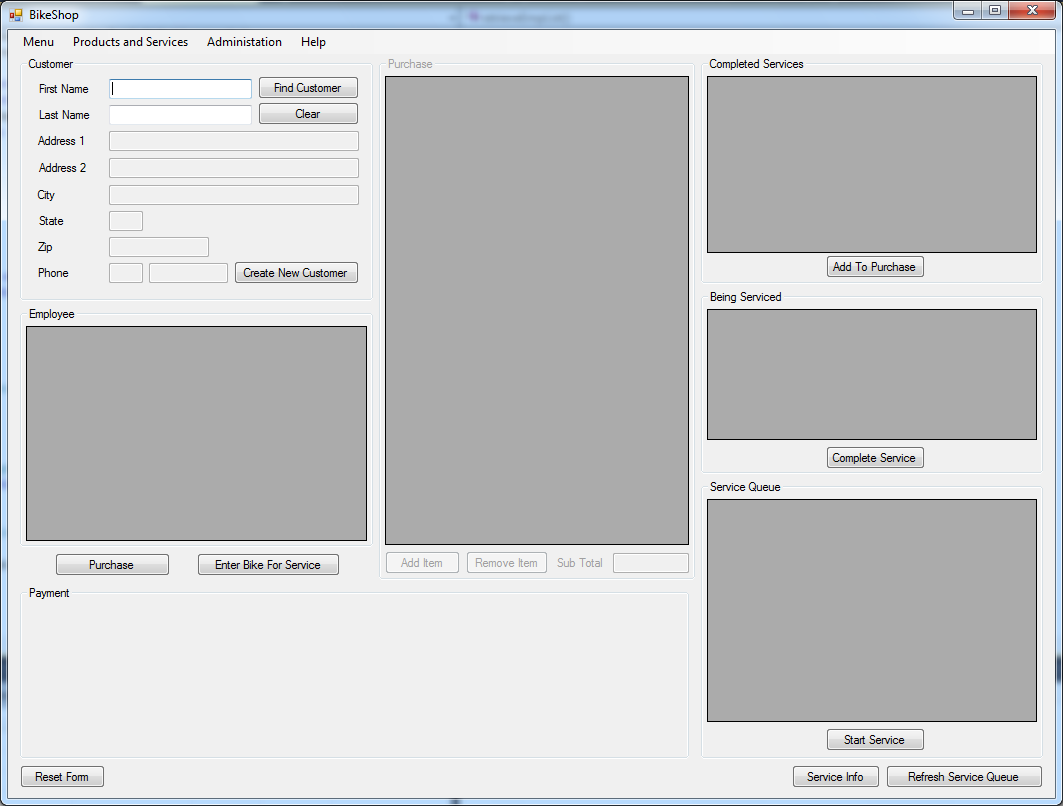
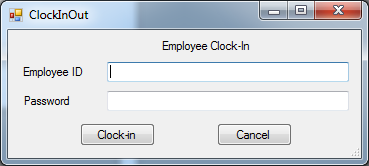
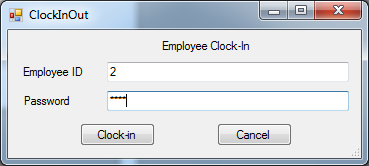
Create Customer Bike

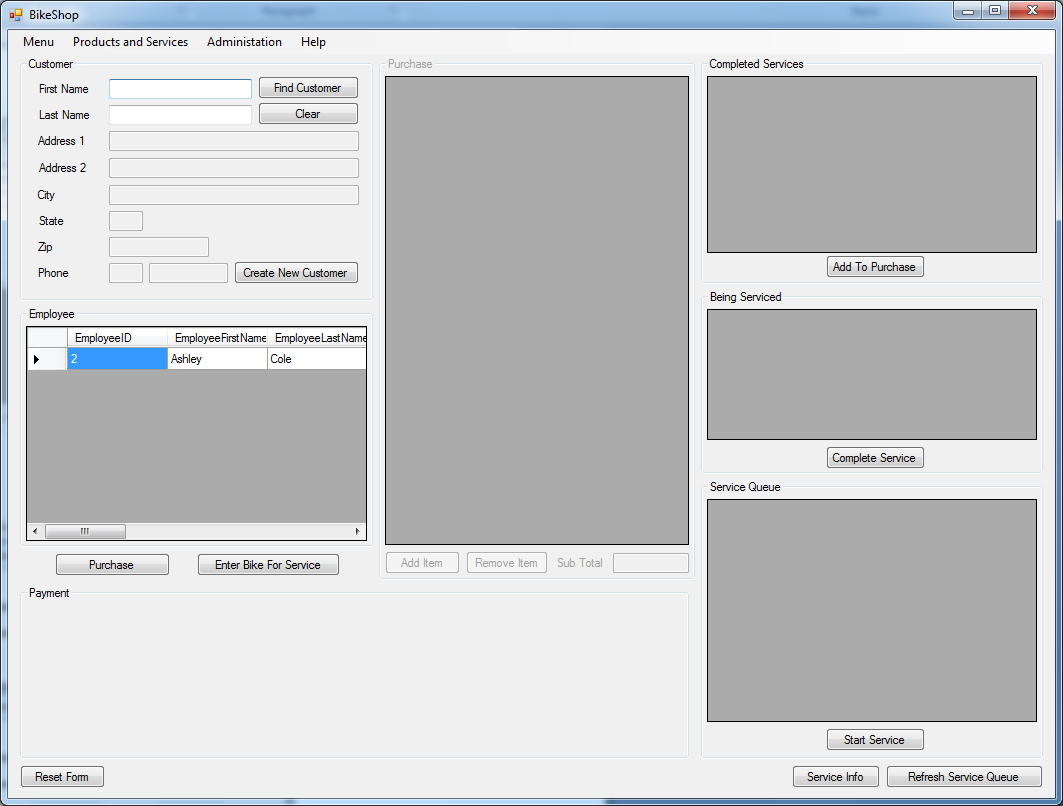
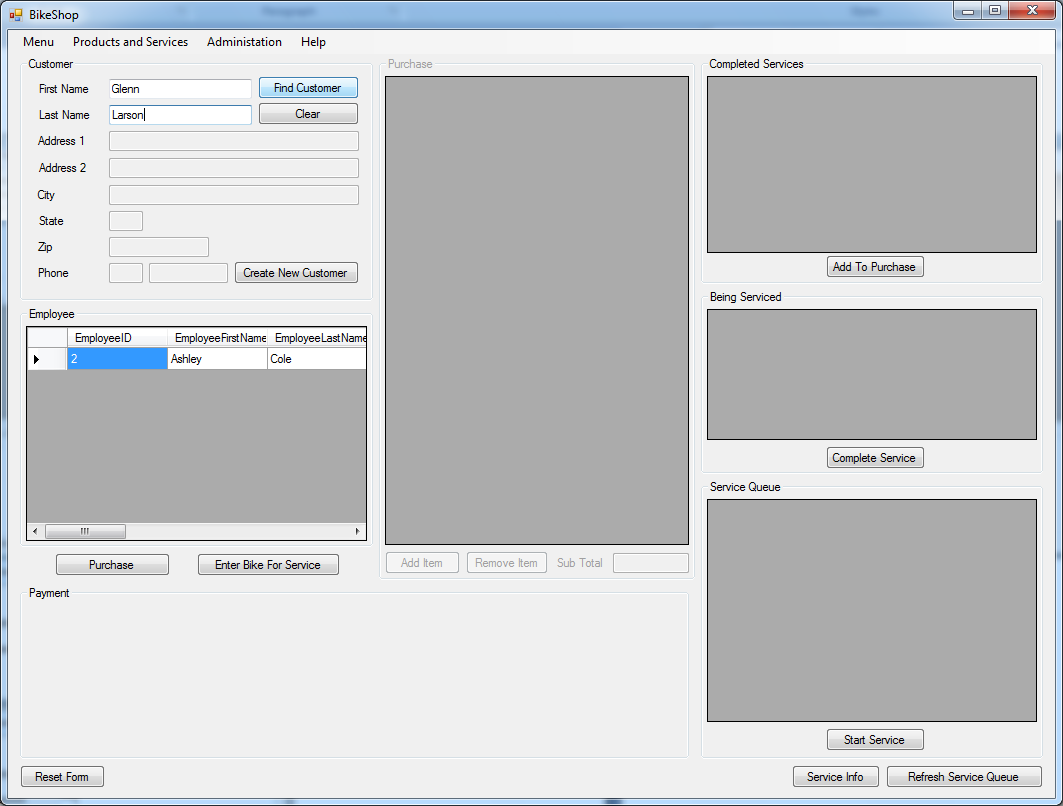
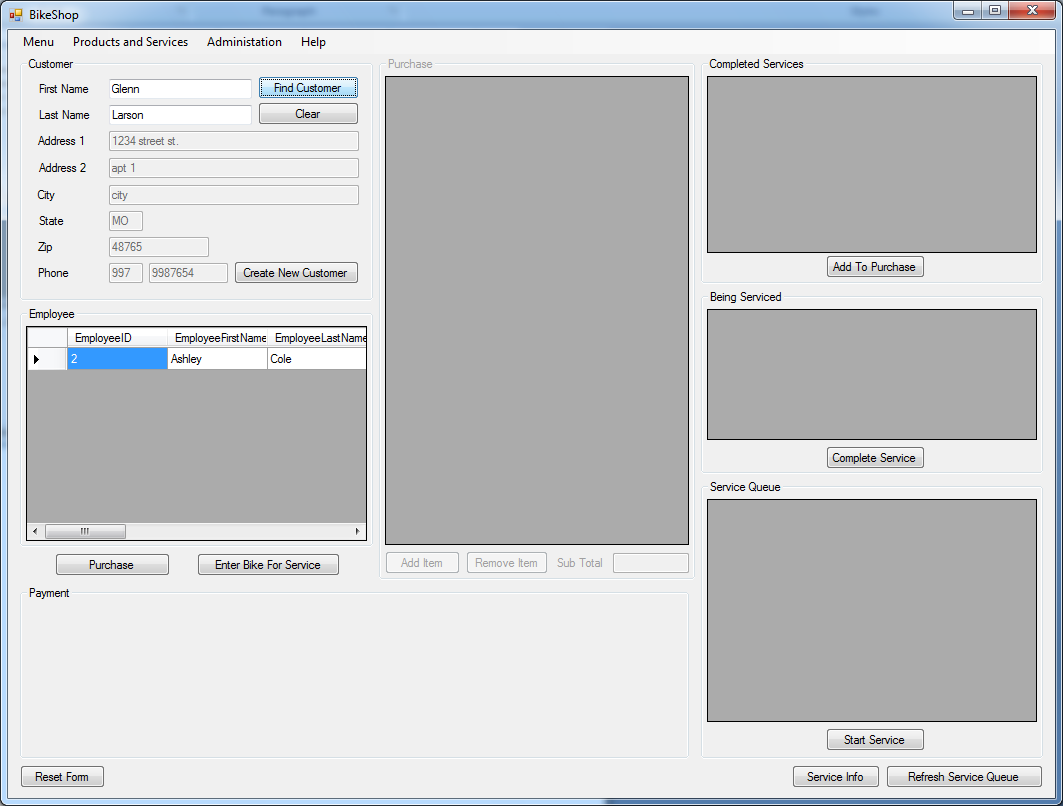
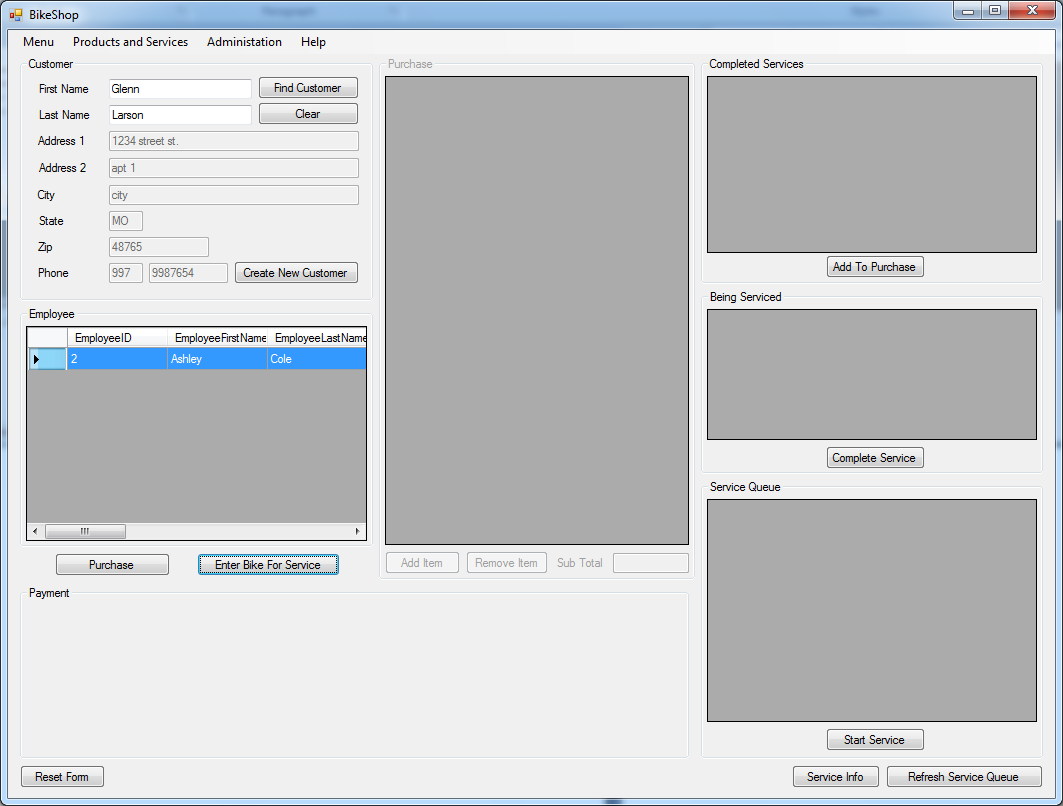
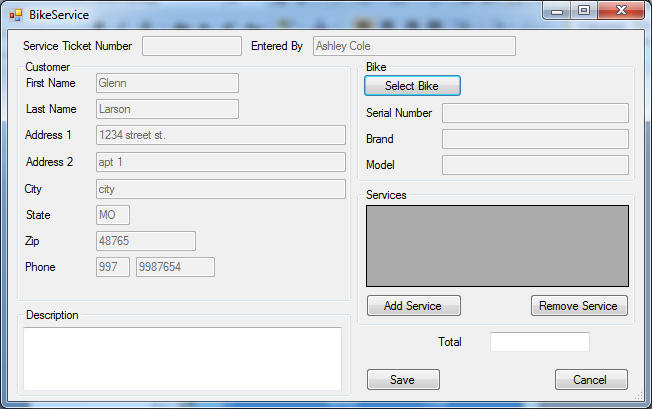
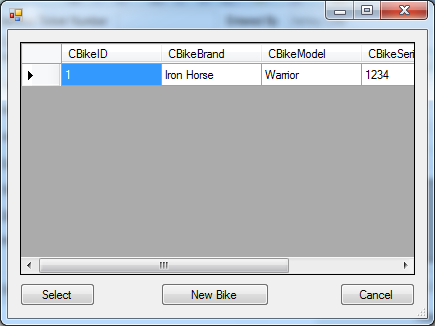
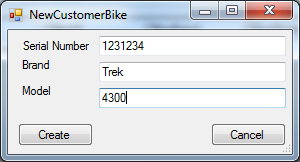
Select Customer Bike

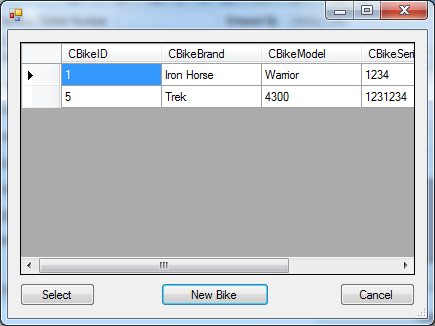
Enter service ticket

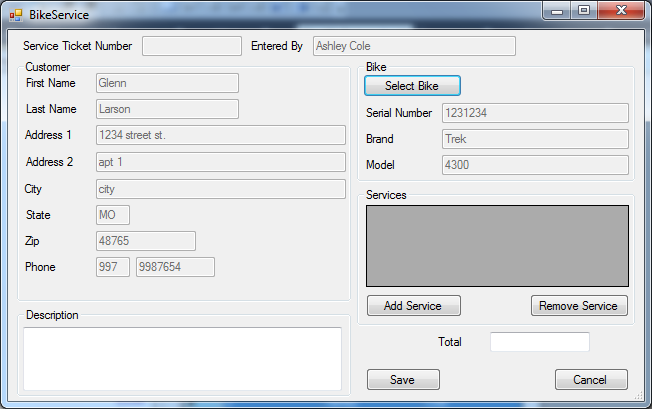
Change password

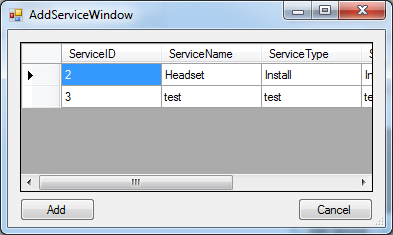
### Screen Shots of Enter Service Ticket

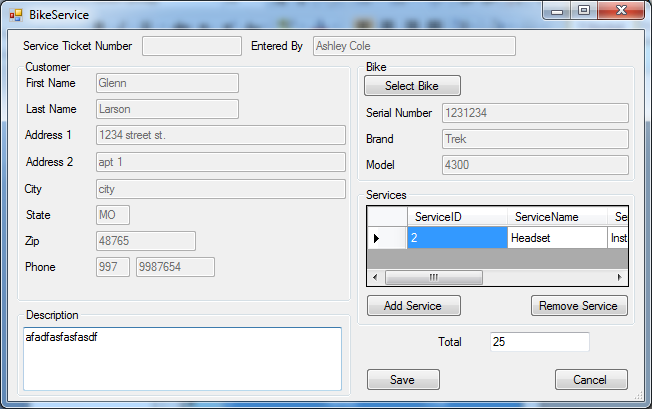
  

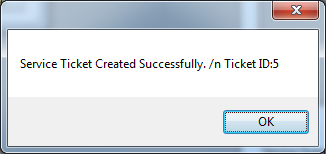
      











Source Code - attached at end

1. Menu Hiearchy

The Initial menu hierarchy is as follows. The way I have the application setup right now the menu doesn’t change based on the use case or the user that is using the system.

These will be available for all users

Menu

Employee Clock In

Employee Clock Out

Exit

Products and Services

Manage Products

Manage Services

Manage New Bikes

Administration

Employees

Help

Help

1. Reports

Employee Service Bonus Report

Date Range

10/1/2011 – 10/31/2011

Employee Name Number Of Services Most Common Service Bonus

Glenn Larson 5 Headset $20.00

Ashley Cole 10 TuneUp $40.00

Bart Simpson 0 none $0.00

Charlie Sheen 25 Winning $100.00

Glenn Larson 5 Headset $20.00

Ashley Cole 10 TuneUp $40.00

Bart Simpson 0 none $0.00

Charlie Sheen 25 Winning $100.00

Items Sales Report

Date Range

10/1/2011 – 10/31/2011

Item Name Number Sold This Month Number Sold To Date

Handle Bar 5 10

Headset 2 5

Suspension Fork 1 1

Trek Bike 2 3

Specialized Bike 6 3

Services 30 100