**Software engineering**

**project 3**

**CSCI3251.21 - Project 3**

**Course Instructor: Dr. Yong-Ming Tang**

Shunqi Zheng

Shih-Hung

Problem Statement:

EZPass(NJ) System is an electronic toll collection system for drivers easily to pay the toll When they drive through the toll facilities that offer the E-ZPass. Every driver can use this convenient System by attaching a small electronic device, called a tag or a transponder, to their vehicles.

The EZPass(NJ)System will retrieve each driver’s information and collect them to send it to the server. Each time when using a toll facility where E-ZPass is offered, an antenna at the toll station will read the vehicle and account information contained in everyone’s tag. The appropriate toll is then electronically debited from each prepaid account. Each transaction will be included in the driver’s periodic statement.

Drivers can open an account in two different options(individual and business), they need to complete an application online or call the phone 1-888-AUTO-TOLL or just come into a Customer Service Centers with a list of private information(Name, Zip Code, Cell Phone and so on), for business, it is the same process. Even rental cars can be tagged on the account.

Registrars have the ability to know about the System(how to run, where they can use) including get access to the database, retrieve customer information, modify system architectural, etc.

//Missed the list of activities. You have it in your Peoject 1A.

Functional model:

*Sign up*

*Scenario Name* Failed: Sign up

*Participating Actor* Edward: Customer

*Instance*

*Flow of Events* 1. Edward selects Title \*\*\*\*, Suffix \*\*\*\*, Email Monthly(first 6months are free) and enters Contact Name Edward, Mailing Address 1000 River Road, Teaneck, NJ, Zip Code 07666, Email Address [Edward@gmail.com](mailto:Edward@gmail.com), Retype Email [Edward@gmail.com](mailto:Edward@gmail.com), Cell Phone 201-845-1234, Landline/Alternate Phone:204-123-1234, enters Username Edward123, Password \*\*\*\*\*\*, Retype Password \*\*\*\*\*\* Security Question Name of your parent? Security Answer Edwardd, 4-Digit PIN 1234

2. “Sign up Failed because of invalid Contact Name” is showed to Edward.

*Scenario Name* Succeeded: Sign up

*Participating Actor* Edward: Customer

*Instance*

*Flow of Events* 1. Edward selects Title M, Suffix I, Email Monthly(first 6months are free)

and enters Contact Name Edward, Mailing Address 1000 River Road, Teaneck,

NJ, Zip Code 07666, Email Address [Edward@gmail.com](mailto:Edward@gmail.com), Retype Email

[Edward@gmail.com](mailto:Edward@gmail.com), Cell Phone 201-845-1234, LandLine/Alternate Phone:204-

123-1234, enters Username Edward123, Password \*\*\*\*\*\*, Retype Password

\*\*\*\*\*\* Security Question Name of your parent? Security Answer Edwardd, 4-

Digit PIN 1234

2. “Sign up Successfully” is showed to Edward.

Initial Use Case:

*Use Case Name* Sign up

*Participating Actor* Initiated by a Customer

*Flow of Events* 1. Customer selects Title M, Suffix I, Email Monthly(first 6months are free)

and enters Contact Name Edward, Mailing Address 1000 River Road, Teaneck,

NJ, Zip Code 07666, Email Address [Edward@gmail.com](mailto:Edward@gmail.com), Retype Email

[Edward@gmail.com](mailto:Edward@gmail.com), Cell Phone 201-845-1234, Landline/Alternate Phone:204-

123-1234, enters Username Edward123, Password \*\*\*\*\*\*, Retype Password

\*\*\*\*\*\* Security Question Name of your parent? Security Answer Edwardd, 4-

Digit PIN 1234 //Does not have to include the specific data in a use case.

2. The system shows “Sign up Successfully” or “Sign up Failed because of invalid Contact Name” to the customer.

*Entry Conditions* Webpage for sign up has been loaded

*Exit Conditions*  Sign up is confirmed of success or failure or session times out.

*Quality Requirements* The Username must be 6 to 20 characters and contain at least 1 letter and 1 number. Username is not case sensitive.

The Password must be 8 to 12 characters and contain at least 1 uppercase letter, 1 lowercase letter, and 1 number.

Final Use Case:

*Use Case Name* Sign up

*Participating Actor* Initiated by a Customer

*Flow of Events* 1. Customer selects Title, Suffix, Email Monthly and enters Contact Name Edward, Mailing Address, Zip Code, Email Address, Retype Email, Cell Phone, LandLine/Alternate Phone, enter Username, Password, Retype Password, Security Question, Security Answer, 4-Digit PIN, and clicks on Signup contained in SignupBO

2. SignupControl is created by SignupBO.

3. The control object creates the Account object and invokes an operation of Account to verify the existing username.

4. SignupControl creates ConfirmationBO that shows “Sign up Successfully” or Sign up Failed because of invalid Contact Name” to the customer.

4. “Sign up Successfully” is showed to the customer.

*Entry Conditions* Webpage for signup has been loaded

Exit Conditions Sign up is confirmed of success or failure or session times out.

*Quality Requirements* The Username must be 6 to 20 characters and contain at least 1 letter and 1 number. Username is not case sensitive.

The Password must be 8 to 12 characters and contain at least 1 uppercase letter, 1 lowercase letter, and 1 number.

*Login*

*Scenario Name* Failed:Login

*Participating Actor* Edward:Customer

*Instance*

*Flow of Events* 1. Edward enters Username Edward123, Password \*\*\*\*\*\*, and clicks Login.

2. “Login Failed because of invalid username or password” is shown to Edward.

*Scenario Name* Succeeded:Login

*Participating Actor* Edward: Customer

*Instance*

*Flow of Events* 1. Edward enters Username Edward123, Password \*\*\*\*\*\*\*\*, and clicks Login.

2. are showed to Edward. //Why banking? Careless error.

*Scenario Name* AccountSuspened:Login

*Participating Actor* Edward: Customer

*Instance*

*Flow of Events* 1. Edward enters Username Edward123, Password \*\*\*\*\*\*\*\*, and clicks Login.

2. “Your Account is suspended. Please call customer service.” is showed to Edward.

Initial Use Case:

*Use Case Name* Login

*Participating Actor* Initiated by a Customer

*Flow of Events* 1. The customer enters his/her Username, Password, and clicks Login.

2. The system shows Account Summary, Bill Pay, Transfer, …banking services or “Login Failed because of invalid username or password” or “Your Account is suspended. Please call customer service.” to the customer.

*Entry Conditions* Webpage for login has been loaded.

*Exit Conditions* Login is confirmed of success or failure.

*Quality Requirements* Password’s length is at least 8 characters; The confirmation must be received in 0.01 seconds; Https is required.

Refined Use Case:

*Use Case Name* Login

*Participating Actor* Initiated by a Customer

*Flow of Events* 1. The customer enters a username, password, and clicks on Login contained in LoginBO.

2. LoginControl is created by LoginBO.

3. The control object creates the Account object and invokes an operation of Account to verify the correctness of the username and password.

4. LoginControl creates ConfirmationBO that shows Account Summary, Bill Pay, Transfer, …banking services or “Login Failed because of invalid username or password” or “Your Account is suspended. Please call customer service.” to the customer.

*Entry Conditions* Webpage for login has been loaded.

*Exit Conditions* Login is confirmed of success or failure.

*Quality Requirements* Password’s length is at least 8 characters; The confirmation must be received in 0.01 seconds; Https is required.

*Changepassword*

*Scenario Name* Succeeded: Changepassword

*Participating Actor* Edward: Customer

*Instance*

*Flow of Events* 1. Edward enters original password \*\*\*\* and new password \*\*\*\*, and click the “Change Password” button. //Missed confirm password.

3. “change password successfully” is shown to Edward.

*Scenario Name* Failed: Changepassword

*Participating Actor* Edward:Customer

*Instance*

*Flow of Events* 1. Edward enter original password \*\*\*\* and new password \*\*\*\*, and click “Change Password” button.

3. “change password failed because the password your entered is unqualified” or “Change passwords failed because your password can not be the same ” is shown to Edward.

Initial Use Case:

*Use Case Name* Changepassword

*Participating Actor* Initiated by a Customer

*Flow of Events* 1. The customer enter original password \*\*\*\* and new password \*\*\*\*, and click “Change Password” button.2. The system shows “Change password Successfully” or “Change passwords failed because your password can not be the same ” or “because the password your entered is unqualified” to the customer.

*Entry Conditions* Webpage for Change password has been loaded.

*Exit Conditions* Change password is confirmed of success or failed.

*Quality Requirements* Password’s length is at least 8 characters, password must be included at least one upper letter and one lower letter; The confirmation must be received in 0.01 seconds; Https is required.

Final Use Case:

*Use Case Name* Changepassword

*Participating Actor* Initiated by a Customer

*Flow of Events* 1. enter original password \*\*\*\* and new password \*\*\*\*, and click “Change Password” button.2.Changepassword Control is created by ChangepasswordBO.

3.The control object creates Account object and invokes an operation of Account verify the existing password 4. The system shows “Change password Successfully” or “Change passwords failed because your password can not be the same ” or “because the password your entered is unqualified” to the customer.

*Entry Conditions* Webpage for Change password has been loaded.

*Exit Conditions* Changepassword is confirmed of success or failed.

*Quality Requirements* Password’s length is at least 8 characters, password must be included at least one upper letter and one lower letter; The confirmation must be received in 0.01 seconds; Https is required.

*PayToll*

*Scenario name* PaidSuccess:PayToll

*Participating Actor* Jonny:Driver

*Instance*

*Flow of Events* 1. Jonny is driving through EZPass Lane 4 at GWB on 02/13/2018 at 11:15AM with his EZ tag code 20180213123.

2. “Paid” is signaled in the Green Light on left side of the lane.

*Scenario name* NotSuccess:PayToll

*Participating Actor* Joanna:Driver

*Instance*

*Flow of Events* 1. Joanna is driving through EZPass Lane 5 at GWB on 02/13/2018 at 11:15AM with her EZ tag code 20180213321.

2. “Error” is signaled in the Green Light on left side of the lane.

*Scenario name* LowBal:PayToll

*Participating Actor* Edward:Driver

*Instance*

*Flow of Events* 1. Josh is driving through EZPass Lane 6 at GWB on 02/13/2018 at 11:15AM with her EZ tag code 20180213333.

2. “Low” is signaled in the Green Light on left side of the lane.

*Initial use case*

*Use case name* PayToll

*Participating Actor* Initiated by a Driver //Driver or Customer? Keep the consistence for all the use cases.

*Flow of Events* 1. The driver is driving through an EZPass Lane at a Toll plaza on a specific date at a specific time with his/her EZ tag code.

2. “Paid”, “Error” or “Low” is signaled in the Green Light on left side of the lane.

*Entry conditions* The driver is approaching an EZ gate and enters the detection range of EZ Tag Reader.

*Exit conditions* A confirmation is signaled.

*Quality Requirements* The process must be done in less than 3 seconds.

Final Use Case:

*Use Case Name* PayToll

*Participating Actor* Initiated by a Driver

*Flow of Events* 1. The driver is driving through an EZPass Lane at a Toll plaza on a specific date at a specific time with his/her EZ tag code. //The BO here is TagReader.

2.PayTollControl is created by TagReader.

3.The control object creates Tag object and Account and invokes an operation of Tag and Account to verify the existing Tag information and Account Balance.

4.PayTollControl creates ConfirmationBO that shows “Error” or “Low” to the driver.

*Entry Conditions* The driver is approaching an EZ gate and enters the detection range of EZ Tag

Reader.

*Exit Conditions* A confirmation is signaled.

*Quality Requirements* The process must be done in less than 3 seconds.

AddVehicle

*Scenario Name* AddSuccess:AddVehicle

*Participating Actor* Simon:User //For instances, must be underlined.

*Instance*

*Flow of Events* 1. Simon activates the “AddVehicle” function by putting in all the required information: License Plate: xyxyxyxy, State: New Jersey, Country: USA, Plate Type: Standard, MAKE: BMW, MODEL: 320i, YEAR: 2017, Vehicle Type: Sedan and clicked ADD Vehicle Button.

2. The system shows “Adding Successful”.

*Scenario Name* Cancel:AddVehicle

*Participating Actor* Simon:User Instance

*Flow of Events* 1. Simon activates the “AddVehicle” function by putting in all the required information: License Plate: xyxyxyxy, State: New Jersey, Country: USA, Plate Type: Standard, MAKE: BMW, MODEL: 320i, YEAR: 2017, Vehicle Type: Sedan and clicked ADD Vehicle Button.

2. Simon cancels the “AddVehicle” function by clicking the “CANCEL” button.

3. The system returns to the previous page.

*Initial use case*

*Use Case Name* AddVehicle

*Participating Actor* Initiated by User

*Flow of Events*  1. The user enters License Plate,

State, Country, Plate Type, MAKE, MODEL,YEAR and Vehicle Type.

2. The user clicks “Add Vehicle or “Cancel” button.

3. The system shows “Adding Successful” or canceled.

*Entry Conditions* The actors are logged to the EZPass system and can choose the Add a Vehicle function.

*Exit Conditions* The actors have successfully adding their vehicle or not.

*Quality Requirements* The confirmation must be received in 0.02 seconds; Https is required.

Final Use Case: AddVehicle

*Participating Actor* Initiated by User

*Instance*

*Flow of Events* 1. The user enters License Plate,

State, Country, Plate Type, MAKE, MODEL, YEAR and Vehicle Type in AddVehicleWindow.

2. The user activates the “AddVehicle” function of the system by clicking Add Vehicle button.

3. An object of AddVehicleControl is created.

4. The AddVehicleControl gets all the input data and compare them to their respective constraints.

5. If all of the data are correct then the AddVehicleControl creates an Vehicle object and invokes the add vehicle operation.

6. The AddVehicleControl object displays the message “Adding Successful” or “Adding Unsuccessful” in ShowAddResultWindow.

*Entry Conditions* The actors are logged to the EZPass system can choose the Add a Vehicle function.

Exit Conditions The actors have successfully adding their vehicle or not.

Quality Requirements The confirmation must be received in 0.02 seconds; Https is required.

Remove Vehicle

*Scenario Name* Succeeded:RemoveVehicle

*Participating Actor* Edward:Customer

*Instance*

*Flow of Events* 1. Edward select one or more vehicles//Here missed specific vehicle information and click Remove vehicle button.

2.”Remove vehicle Successfully” is shown to Edward

*Scenario Name* Failed:RemoveVehicle

*Participating Actor* Edward:Customer

*Instance*

*Flow of Events* 1. Edward select one or more vehicles and click Remove vehicle button.

2.”Remove vehicle Failed because the selected vehicle is not existed” is shown to Edward

*Initial use case*

*Use Case Name* RemoveVehicle

*Participating Actor* Initiated by a Customer

*Flow of Events* 1. Customer select one or more vehicles and click Remove vehicle button.

2. “Remove vehicle successfully”or “Remove vehicle Failed because the selected vehicle is not existed” is showed to Edward.

*Entry Conditions* Webpage for Remove vehicle has been loaded

Account has been login

*Exit Conditions* Remove vehicle is confirmed o success or times out.

*Quality Requirements* The confirmation must be received in 0.01 seconds; Https is required.

Final Use Case:

*Use case name* RemoveVehicle

*Participating Actor* Initiated by a Customer

*Instance*

*Flow of Events* 1. Customer select one or more vehicles and click Remove vehicle button.

2. RemoveVehicleControl is created by RemoveVehicleBO.

3. The control object creates vehicle object and invokes an operation of removeVehicle to

remove existing vehicle.

4. “Remove vehicle successfully”or “Remove vehicle Failed because the selected vehicle is not existed” is shown to customer.

*Entry Conditions* Webpage for Remove vehicle has been loaded.

Account has been login.

*Exit Conditions* Remove vehicle is confirmed o success or times out.

*Quality Requirements* The confirmation must be received in 0.01 seconds;

Https is required.

Recharge

*Scenario Name* Succeeded:Recharge

*Participating Actor* Edward:Customer

*Instance*

*Flow of Events* 1. Edward selects the amount of money by click “Recommended base on your recent toll history” or “Other” and typing 50$.

2. Edward select “Use Credit Card on the payment options” or “ Enter New Credit Card” and enter Credit Card information: credit card number 1234-1234-1234, expiration date 05/22, security code 123.

3. Edward select card type: Visa. He click “continue” button.

4. “Your account has been successfully recharged 50$” is shown to Edward.

*Scenario Name* Failed:Recharge

*Participating Actor* Edward:Customer

*Instance*

*Flow of Event* 1. Edward selects the amount of money by click “Recommended base on your recent toll history” or “Other” and typing 50$.

2. Edward select “Use Credit Card on the payment options” or “ Enter New Credit Card” and enter Credit Card information: credit card number 1234-1234-1234, expiration date 05/22, security code 123.

3. Edward select card type: Visa. He click “continue” button.

4. “Recharge failed, Your credit card balance is insufficient” is shown to Edward.

*Initial use case*

*Use Case Name* Recharge

*Participating Actor* Edward: Customer

*Flow of Events* 1.Customer selects the amount of money by click “Recommended base on your recent toll history” or “Other” and typing the amount of certain money..

2. Customer select “Use Credit Card on the payment options” or “ Enter New Credit Card” and enter Credit Card information: credit card number, expiration date, security code.

3. Edward select card type: Visa. He click “continue” button.

4. The system shows “Your account has been successfully recharged \*\*$”, or “Recharge failed, credit card declined” to Customer.

*Entry Conditions* Webpage for recharge has been loaded

*Exit Conditions* Recharge is confirmed of success or failure or session times out.

*Quality Requirements* The confirmation must be received in 1 seconds; Https is required.

Final Use Case:

*Use Case Name* Recharge

*Participating Actor* Edward: Customer

*Flow of Events* 1.Customer selects the amount of money by click “Recommended base on your recent toll history” or “Other” and typing the amount of certain money.

2. RechargeControl is created by RechargeBO.

3. The control object creates Account object and invokes an operation of recharge.

4. The Account object creates a CreditCard object and invokes the pay operation.

4. RechargeControl creates ConfirmationBO that shows “Your account has been successfully recharged \*\*$”, or “Recharge failed, credit card declined.” to the Customer.

*Entry Conditions* Webpage for recharge has been loaded

*Exit Conditions* Recharge is confirmed of success or failure or session times out.

*Quality Requirements* Credit card must be valid.

The confirmation must be received in 1 seconds; Https is required.

ViewTransactions

*Scenario Name* TransactionsFound:ViewTransactions

*Participating Actor* Simon:User

*Instance*

*Flow of Events* 1. Simon activates the “ViewTransactions” function by putting in all the required information, Date Type: Transaction date, Start Date: 07/31/2019, End Date: 09/17/2019, Toll Amount: ALL, Roads: ALL, and clicked VIEW Button.

2. The system shows one or multiple results, Posted Date: 09/16/2019, Transaction Date: 09/16/2019, Tag# / Plate #: ,Agency: NJ E-ZPass ,Description: Service Fee, Entry Time: , Entry Plaza: , Exit Time: 02:06:51, Exit Plaza:, Exit Lane: , Amount: ($1.00), Balance: $23.62

*Scenario Name* TransactionsNotFound:ViewTransactions

*Participating Actor* Simon:User

*Flow of Events* 1. Simon activates the “ViewTransactions” function by putting in all the required information, Date Type: Transaction date, Start Date: 07/31/2019, End Date: 09/17/2019, Toll Amount: ALL, Roads: ALL, and clicked VIEW Button.

2. The system shows “Not Found”.

*Initial use case*

*Use Case Name* ViewTransaction

*Participating Actor* Initiated by User

*Flow of Events* 1. The user enters Date Type, Start Date, End Date, Toll, Amount and Roads for the “ViewTransactions” function, The user clicking VIEW button.

2. The system shows one or more results (Posted Date, Transaction Date, Tag# / Plate #, Agency, Description, Entry Time, Entry Plaza, Exit Time, Exit Plaza, Exit Lane, Amount and Balance) or not found.

*Entry Conditions* The actors are logged to the EZPass system can choose the Transactions function. Use Transaction Date for Date Type, All for the toll amount, and All for all of the roads

*Exit Conditions* The actors have received the details of the transactions or nothing found.

*Quality Requirements* Showing the details of the transactions no later than 2 seconds.

Final Use Case: ViewTransaction

*Participating Actor* Initiated by User

*Instance*

*Flow of Events* 1. The user enters Date Type, Start Date, End Date, Toll Amount and Roads in ViewTransactionsWindow. The user clicks View button.

2. An object of ViewTransactionsControl is created.

3. The ViewTransactionsControl gets all the input data and creates a Transaction object and invokes the retrieve operation.

4. The ViewTransactionsControl receives the details of the Posted Date, Transaction Date, Tag# / Plate #, Agency, Description, Entry Time, Entry Plaza, Exit Time, Exit Plaza, Exit Lane, Amount and Balance.

5. The ViewTransactionsControl object displays the transaction details or not found in ViewTransactionsWindow.

*Entry Conditions* The actors are logged to the EZPass system can choose the Transactions function, Use Transaction Date for Date Type, All for toll amount, and All for all the roads

Exit Conditions The actors have received the details of the transactions or nothing found.

Quality Requirements Showing the details of the transaction no later than 2 seconds.

EnrollPlans

*Scenario Name* Succeeded:Enrollplans

*Participating Actor* Edward:Customer

*Instance*

*Flow of Events* 1.Edward select a Port Authority Carpool Plan/New York State Bridge Authority Discount Plan and click Add plan.

2. “Add plan successfully” is shown to Edward.

*Scenario Name* Failed:Enrollplans

*Participating Actor* Edward:Customer

*Instance*

*Flow of Event* 1.Edward select a Port Authority Carpool Plan/New York State Bridge Authority Discount Plan and click “Add plan” button.

2. “Add plan failed because you already have this plan!” is shown to Edward.

*Initial use case* Enrollplans

*Use Case Name* Initiated by a Customer

*Participating Actor* 1.Edward select a Port Authority Carpool Plan/New York State Bridge Authority Discount Plan and click “Add plan” button.

*Instance*

*Flow of Events* 2. The system shows “Add plan successfully” or “Add plan failed since you already have this plan!” to the customer.

*Entry Conditions* Webpage for enroll plan has been loaded

*Exit Conditions* Enroll plan is confirmed of success or failure or session times out.

*Quality Requirements* The confirmation must be received in 0.01 seconds; Https is required.

Final Use Case:

*Use Case Name* Enrollplans

*Participating Actor* Initiated by a customer.

*Flow of Events*

1.Edward select a Port Authority Carpool Plan/New York State Bridge Authority

Discount Plan and click “Add plan” button.

2. EnrollPlansControl is created by EnrollPlansBO.

3. The control object creates plan object and invokes the addPlan operaion.

4. EnrollPlansControl creates ConfirmationBO that shows “Add plan successfully” or “Add plan failed because you already have this plan!” to the customer.

*Entry Conditions* Webpage for enroll plan has been loaded

*Exit Conditions* Enroll plan is confirmed of success or failure or session times out.

*Quality Requirements* The confirmation must be received in 0.01 seconds; Https is required.

Logout

*Scenario Name* Logout

*Participating Actor* Enrollplans

*Instance* Edward:Customer

*Flow of Events*

1.Edward clicks “Log out” button.

2. “Log out successfully” is shown to Edward.

Initial Use Case:

*Use Case Name* Logout

*Participating Actor* Initiated by a Customer

*Flow of Events*  1.Customer clicks “Log out” button.

2. “Log out successfully” is shown to the customer

Entry Conditions Interface for logout has been loaded.

Exit Conditions Logout is a confirmed of success.

Quality Requirements The confirmation must be received in 0.01 seconds; Https is required.

*Final use case*

*Use Case Name* Logout

*Participating Actor* Initiated by a Customer

*Flow*

*Instanice of events*  1.Customer clicks “Log out” button.

2.LogoutControl is created by LogoutBO

3.The control object terminates the account.

4. LogoutControl creates ConfirmationBO that shows “Log out successfully” is shown to the customer..

*Entry Conditions* Interface for logout has been loaded.

*Exit Conditions* Logout is a confirmed of success.

*Quality Requirements* The confirmation must be received in 0.01 seconds; Https is required.

Dynamic model:





















State diagrams: //The initial state for Vehicle is Added. //You did state diagram for both Vehicle and Transaction in Project 2A. //Missed states LoggedIn, LoggedOut for object Account.















Class diagram: //Missed other attributes and operations of the entity classes. //All errors in Analysis model cost (-5)



Object diagram:



Object class: CreditCard

Design pattern: Bridge



//The pattern looks good. However, pay() may not have to be implememted differently just because of the different type.

Object Design:

1. **EzAccount**

**context** EzAccount **inv**:Username != null AND age >= 16 AND AcctNo !=NULL

**context** EzAccount::createAccount(String Username, String Password, String Name, Date DoB, String Address, String Phone) **pre**:

Username != null AND

Password != null AND

Name != null AND

DoB != null AND

Address != null AND

Phone != null

**context** EzAccount::createAccount(String Username, String Password, String Name, Date DoB, String Address, String Phone) **post**:

isActive = true

**context** EzAccount::deleteAccount (String Username) **pre**:

Username != null AND

isActive = true

**context** EzAccount::deleteAccount(String Username) **post**:

isActive = false

**context** EzAccount::retrieveAccountInfo(String Username) **pre**:

Username!= null AND

isServing = true

**context** EzAccount::retrieveAccountInfo(String Username) **post**:

**context** EzAccount::modifyAccount(String Username, String Password, String Name, Date DoB, String Address, String Phone) **pre**:

Username != null OR

Password != null OR

Name != null OR

DoB != null OR

Address != null OR

Phone != null

**context** EzAccount::modifyAccount(String NewUsername, String Password, String Name, Date DoB, String Address, String Phone) **post**:

Username == NewUsername OR //using different parameter names

Password = Password OR

Name = Name OR

DoB = DoB OR

Address = Address OR

Phone = Phone

**context** EzAccount::recharge (CreditCard card, Float Amount) **pre**:

cardNumber != null AND

Amount > 0

**context** EzAccount::recharge (CreditCard card, Float Amount) **post**:

Balance == Balance + Amount

**context** EzAccount::payWithBalance (Float Balance, Float Amount) **pre**:

amount > 0 AND

Balance > amount

**context** EzAccount::payWithBalance (Float Balance, Float Amount) **post**:

Balance == Balance – Amount

**context** EzAccount::verifyUsernameANDPassword (string Username, string Password) **pre**:

Username != null

Password != null

**context** EzAccount:: verifyUsernameANDPassword (string Username, string Password) **post**:

1. **EzTag**

**context** EzTag **inv**:Tag\_Number != null

**context** EzTag::addTag(String Tag\_Number, String Type, Date ValidDate) **pre**:

Tag\_Number != null AND

Type != null AND

ValidDate > CurrentDate

**context** EzTag::addTag(String Tag\_Number, String Type, Date ValidDate) **post**:

isActive = true

1. **Violation**

**context** Violation **inv**:Violation\_ID != null

**context** Violation::addViolation(String Violation\_ID, String LicensePlate, String State, String Type, Date Date&Time, String Facility, String Statue, Float toll) **pre**:

Violation\_ID != null AND

LicensePlate != null AND

State != null AND

Type != null AND

Date&Time != null AND

Facility != null AND

Statue != null AND

toll > 0

**context** Violation::addViolation(String Violation\_ID, String LicensePlate, String State, String Type, Date Date&Time, String Facility, String Statue, Float toll) **post**:

isPaid = false

**context** Violation::retrieveViolationInfo(String Violation\_ID) **pre**:

Violation\_ID != null

**context** Violation::retrieveViolationInfo(String Violation\_ID) **post**:

**context** Violation::modifyViolation(String Violation\_ID, String LicensePlate, String State, String Type, Date Date&Time, String Facility, String Statue, Float toll) **pre**:

Violation\_ID != null OR

LicensePlate != null OR

State != null OR

Type != null OR

Date&Time != null OR

Facility != null OR

Statue != null AND

toll > 0

**context** Violation::modifyViolation(String Violation\_ID, String LicensePlate, String State, String Type, Date Date&Time, String Facility, String Statue, Float toll) **post**: //Use different parameter names.

Violation\_ID == Violation\_ID OR //It’s ==, equivalent.

LicensePlate = LicensePlate OR

State = State OR

Type = Type OR

Date&Time = Date& OR

Facility = Facility OR

Statue = Statue OR

toll = toll

**context** Violation::isPaid(String Violation\_ID) **pre**:

Violation\_ID != null AND

isPaid == false

**context** Violation::isPaid(String Violation\_ID) **post**:

isPaid = true

**context** Violation::notifyOwner(Vehicle vehicle) **pre**:

vehicle.LicensePlate != null

**context** Violation::notifyOwner(Vehicle vehicle) **post**:

isNotified = true;

1. **Vehicle**

**context** Vehicle **inv**: LicensePlate != null

**context** Vehicle::addVehicle(String LicensePlate, String State, String Country, String PlateType, String MAKE, String MODEL, String YEAR, String VehicleType) **pre**:

LicensePlate != null AND

State != null AND

Country != null AND

PlateType != null AND

MAKE != null AND

MODEL != null AND

YEAR != null AND

VehicleType != null

**context** Vehicle::addVehicle(String LicensePlate, String State, String Country, String PlateType, String MAKE, String MODEL, String YEAR, String VehicleType) **post**:

isServing = true

**context** Vehicle::remove Vehicle(String LicensePlate) **pre**:

LicensePlate != null AND

isServing == true

**context** Vehicle::remove Vehicle(String LicensePlate) **post**:

isServing = false

**context** Vehicle::retrieveVehicleInfo(String LicensePlate) **pre**:

LicensePlate != null AND

isServing == true

**context** Vehicle::retrieveVehicleInfo(String LicensePlate) **post**:

**context** Vehicle::modifyVehicle(String LicensePlate, String State, String Country, String PlateType, String MAKE, String MODEL, String YEAR, String VehicleType) **pre**:

LicensePlate != null OR

State != null OR

Country != null OR

PlateType != null OR

MAKE != null OR

MODEL != null OR

YEAR != null OR

VehicleType != null

**context** Vehicle::modifyVehicle(String NewLicensePlate, String State, String Country, String PlateType, String MAKE, String MODEL, String YEAR, String VehicleType) **post**:

LicensePlate == NewLicensePlate OR

State = State OR

Country = Country OR

PlateType = PlateType OR

MAKE = MAKE OR

MODEL = MODEL OR

YEAR = YEAR OR

VehicleType = VehicleType

1. **Transaction**

**context** Transaction **inv**: Transactionid != null AND Balance != null

**context** Transaction::addTransaction(String Transaction\_ID, Date PostedDate, Date TransactionDate, String Tag# / Plate #, String Agency, String Description, Time EntryTime, String EntryPlaza, Time ExitTime, String ExitPlaza, Int ExitLane, Float Amount) **pre**:

Transaction\_ID != null AND

PostedDate != null AND

TransactionDate != null AND

Tag# / Plate # != null AND

Agency != null AND

EntryTime != null AND

EntryPlaza != null AND

ExitTime != null AND

ExitPlaza != null AND

ExitLane != null AND

Amount > 0 AND

PostetDate >= TransactionDate AND

EntryTime > ExitTime AND

EntryPlaza == ExitPlaza

**context** Transaction::addTransaction(String Transaction\_ID, Date PostedDate, Date TransactionDate, String Tag# / Plate #, String Agency, String Description, Time EntryTime, String EntryPlaza, Time ExitTime, String ExitPlaza, Int ExitLane, Float Amount) **post**:

isPosted = true AND

Balance = Balance - Amount

**context** Transaction::retrieveTransactionInfo(String Transaction\_ID) **pre**:

Transaction\_ID != null

**context** Transaction::retrieveTransactionInfo(String Transaction\_ID) **post**:

**context** Transaction::modifyTransaction(String Transaction\_ID, Date PostedDate, Date TransactionDate, String Tag# / Plate #, String Agency, String Description, Time EntryTime, String EntryPlaza, Time ExitTime, String ExitPlaza, Int ExitLane, Float Amount) **pre**:

Transaction\_ID != null OR

PostedDate != null OR

TransactionDate != null OR

Tag# / Plate # != null OR

Agency != null OR

EntryTime != null OR

EntryPlaza != null OR

ExitTime != null OR

ExitPlaza != null OR

ExitLane != null OR

Balance != null OR

PostetDate >= TransactionDate OR

EntryTime > ExitTime OR

EntryPlaza == ExitPlaza AND

Amount > 0

**context** Transaction::modifyTransaction(String Transaction\_ID, Date PostedDate, Date TransactionDate, String Tag# / Plate #, String Agency, String Description, Time EntryTime, String EntryPlaza, Time ExitTime, String ExitPlaza, Int ExitLane, Float Amount) **post**:

PostedDate = PostedDate OR //Similar issues here

TransactionDate = TransactionDate OR

Tag# / Plate # = Tag# / Plate # OR

Agency = Agency OR

EntryTime = EntryTime OR

EntryPlaza = EntryPlaza OR

ExitTime = ExitTime OR

ExitPlaza = ExitPlaza OR

ExitLane = ExitLane OR

Amount = Amount OR

Balance = Balance

1. **Credit Card**

**context** CreditCard **inv**: Card\_Number != null AND GoodThru >= CurrentDate

**context** CreditCard::addCard(String Card\_Number, String Owner\_Name, Date GoodThru, String CVV) **pre**:

Card\_Number != null AND

Owner\_Name != null AND

GoodThru != null AND

CVV != null

**context** CreditCard::addCard(String Card\_Number, String Owner\_Name, Date GoodThru, String CVV) **post**:

**context** CreditCard::removeCard(String Card\_Number) **pre**:

Card\_Number != null

**context** CreditCard::addCard(String Card\_Number) **post**:

Card\_Number = null AND

Owner\_Name = null AND

GoodThru = null AND

CVV = null

**context** CreditCard::modifyCard(String Card\_Number, String Owner\_Name, Date GoodThru, String CVV) **pre**:

Card\_Number != null OR

Owner\_Name != null OR

GoodThru != null OR

CVV != null

**context** CreditCard::modifyCard(String Card\_Number, String Owner\_Name, Date GoodThru, String CVV) **post**:

Card\_Number = Card\_Number OR

Owner\_Name = Owner\_Name OR

GoodThru = GoodThru OR

CVV = CVV

//Invoked when used by EzAccount to recharge

**context** CreditCard::pay(String Card\_Number, String Owner\_Name, Date GoodThru, String CVV, Float Amount) **pre**:

Card\_Number != null AND

Owner\_Name != null AND

GoodThru != null AND

CVV != null AND

Amount > 0

**context** CreditCard::pay(String Card\_Number, String Owner\_Name, Date GoodThru, String CVV, Float Amount) **post**:

Balance = Balance + Amount //The same as recharge operation so I don’t know if this is //necessary

1. **Plan**

**context** Plan **inv**: Plan\_ID != null

**多种操作（Add, Delete,** **SuspendPlan）**

**context** Plan::addPlan(String Plan\_ID, Date Date\_Added, Float Payment) **pre**:

Plan\_ID != null AND

Date\_Added >= CurrentDate AND

Payment > 0

**context** Plan::addPlan(String Plan\_ID, Date Date\_Added, Float Payment) **post**:

isActive = true

**context** Plan::deletePlan(String Plan\_ID) **pre**:

Plan\_ID != null AND

isActive = true

**context** Plan:: deletePlan(String Plan\_ID) **post**:

isActive = false

**context** Plan::suspendPlan(String Plan\_ID) **pre**:

Plan\_ID != null AND

isActive = true

**context** Plan:: suspendPlan(String Plan\_ID) **post**:

isActive = false

isSuspended = true

**context** Plan::retrievePlanInfo(String Plan\_ID) **pre**:

Plan\_ID != null AND

isActive = true

**context** Plan::retrievePlanInfo(String Plan\_ID) **post**:

//Verifying whether the account is eligible or not

**context** Plan::verifyPrerequisite(EzAccount Account) **pre**:

Account != null

**context** Plan::verifyPrerequisite(EzAccount Account) **post**:

isVerified = true OR

isVerified = false

//All the operations and attibutes here must be from Analysis Object Model to maintain the consistence.

//All errors in Object Design cost (-3)

System Design:

Subsystems



Software Architecture //The correct architecture for this system is the combination of Client/Server and MVC(Model, View, Controller). Then the subsystems follow the architecture.

End device

Transponder Antena

function

1.Vehicle verification 2.User Account data validation 3.Capture Antena Signal 4. Valid Toll Amount deduction 5.Generate Bill

process

1.Verfication and Security Management 2. Reuqired Amount validation and deduction process 3.Bill and Report generation 4.Send bill to customer 5.show signal

DataBase

1.Transcation Management

2.Vehicle DataBase

Deployment diagram



//Missed toll plaza on the user side.

//WebServer does not access DB. App Server does. So assume that your App server is the Windows server.

Global access table

(Driver, Ezaccount, createAccount()) //Show operations with ().

(Driver, Ezaccount, deleteAccount())

(Driver, Ezaccount, RetrieveAccountInfo())

(System administrator, Ezaccount, RetrieveAccountInfo())

(Driver, Ezaccount, ModifyAccount())

(Driver, Ezaccount, Recharge())

(Driver, Ezaccount, payWithBalance())

(Driver, EzTag, addTag())

(Driver, violation,retrieveViolationInfo())

(System administrator, violation, retrieveViolationInfo())

(System administrator, violation, addViolation())

(Driver, violation,isPaid())

(System administrator, violation, notifyOwner())

(System administrator, violation, modifyViolation())

(Driver, vehicle, addVehicle())

(Driver, vehicle, remove Vehicle())

(Driver, vehicle, retrieveVehicleInfo())

(System administrator, vehicle, retrieveVehicleInfo())

(Driver, vehicle, modifyVehicle())

(Driver, Transaction, addTransaction())

(Driver, Transaction, retrieveTransactionInfo())

(System administrator, Transaction, modifyTransaction())

(Driver, CreditCard, addCard())

(Driver, CreditCard, removeCard())

(Driver, CreditCard, modifyCard())

(Driver, CreditCard, pay())

(Driver, Plan, addPlan())

(Driver, Plan, addPlan())

(Driver, Plan, deletePlan())

(Driver, Plan, retrievePlanInfo())

(System administrator, Plan, retrievePlanInfo())

(System administrator, Plan, verifyPrerequisite())

(System administrator, Plan, suspendPlan())

Strategies for global control:

Supposed to be Event-driven control

Boundary use case

*Use Case Name* StartServer

*Flow of Events 2.* Upson successful login, *the* EzPassSystemAdministrator executes the startEzPassSystem command.

3. If the EzPassSystem was previously shut down normally, the server reads the list o legitimate Customers and the index of the active account. If the EzPassSystem had crashed, it notifies the EzPassSystemAdministrator and performs a consistency check on the AccountDBStore.

*Entry condition* 1. The EzPassSystemAdministrator logs into the server machine.

*Exit Conditions* 4. The EzPassSystem is available and waits for connections from Antenna.

//Missed Quality Requirements

//All the errors in System Design cost (-5)