SSA PROJECT REPORT

I. MDA-EFSM model for the GP components

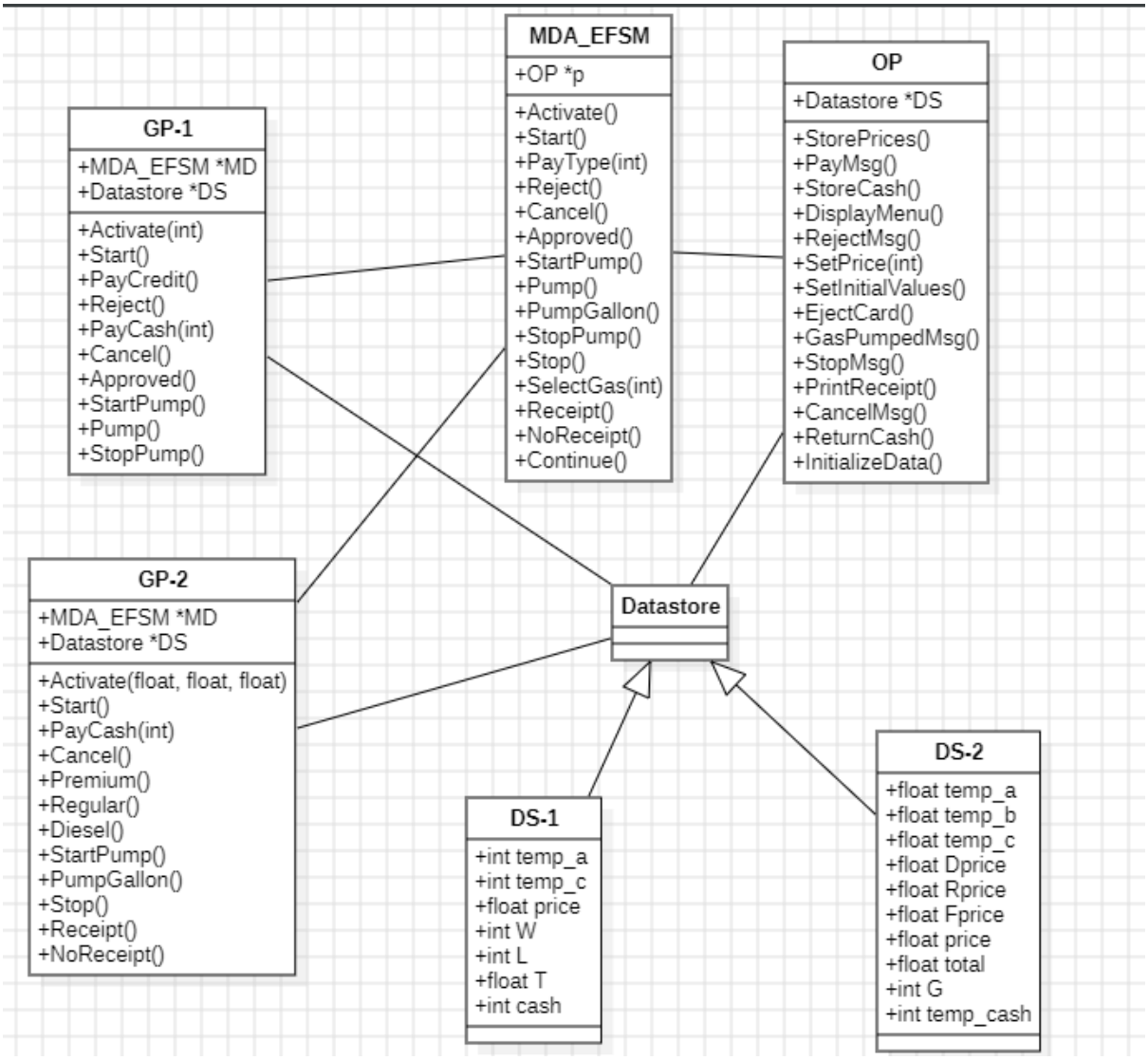
a. MDA-EFSM Events:

- Activate()
- Start()
- PayType(int t) // credit: t=1; cash: t=0;
- Reject()
- Cancel()
- Approved()
- StartPump()
- Pump()
- PumpGallon()
- StopPump()
- Stop()
- SelectGas(int g) // Regular: g=1; Premium: g=2; Diesel: g=3
- Receipt()
- NoReceipt()
- Continue()

b. MDA-EFSM Actions:

- StorePrices // stores price(s) for the gas from the temporary data store
- PayMsg // displays a type of payment method
- StoreCash // stores cash from the temporary data store
- DisplayMenu // display a menu with a list of selections
- RejectMsg // displays credit card not approved message
- SetPrice(int g) // set the price for the gas identified by g
- SetInitialValues // set G (or L) and total to 0;
- Ejectcard // display card is ejected
- GasPumpedMsg // displays the amount of disposed gas
- PrintReceipt // print a receipt
- CancelMsg // displays a cancellation message
- ReturnCash // returns the remaining cash
- InitializeData // set the value of cash to 0

c. State Diagram of MDA-EFSM:



d. Pseudo Code of Input Processor of Gas Pumps: (Gas Pump – 1) & (Gas Pump – 2)

<u>Operations of Input Processor (Gas Pump - 1):</u>

- public void Activate(int a) {

```
if ((a > 0)) {
    if(DS.getTemp_a() == 0) {
    DS.setTemp_a(a);
    DS.setprice(DS.getTemp_a());
    }
```

```
                MD.Activate();
            }
        }
```

- ```
  public void Start() {
      MD.Start();
  }
  ```
- ```
  public void PayCredit() {
      MD.PayCredit();
  }
  ```
- ```
  public void Reject() {
      MD.Reject();
  }
  ```
- ```
  public void PayCash(int c) {
      if(c>0) {
          DS.setTemp_c(c);
          DS.setcash(DS.getTemp_c());
          DS.setW(0);
          MD.PayCash();
      }
  }
  ```
- ```
  public void Cancel() {
      MD.Cancel();
  }
  ```
- ```
  public void Approved() {
      DS.setW(1);
      MD.Approved();
  }
  ```
- ```
  public void StartPump() {
      DS.setL(0);
  ```
  ```

```
        DS.setTotal(0);
        MD.StartPump();
    }
```

- public void Pump() {

```
        if(DS.getW()==1||DS.getW()==0 && DS.getcash() >=
        DS.getprice()*DS.getL()+1) {
            DS.setL(DS.getL()+1);
            DS.setT(DS.getprice()*DS.getL());
            MD.Pump();
        }
        else if(DS.getW()==0 && DS.getcash() < DS.getprice()*DS.getL()+1){
            MD.Pump();
            MD.Receipt();
        }
    }
```

- public void StopPump() {

```
        MD.StopPump();
        MD.Receipt();
    }
```

Operations of Input Processor (Gas Pump - 2):

- public void Activate(float a, float b, float c) {

```
        if ((a > 0) && (b > 0) && (c > 0)) {
            if(DS.gettemp_a()==0 && DS.gettemp_b() == 0 && DS.gettemp_c() ==
            0) {
            DS.settemp_a(a);
            DS.settemp_b(b);
            DS.settemp_c(c);
            DS.setDprice(DS.gettemp_c());
            DS.setRprice(DS.gettemp_a());
            DS.setFprice(DS.gettemp_b());
```

```
                }
                MD.Activate();
            }
        }
```

- public void Start() {

```
            MD.Start();
        }
```

- public void PayCash(int c) {

```
            if(c>0) {
                DS.setTemp_cash(c);
                DS.setcash(DS.getTemp_cash());
                MD.PayCash();
            }
        }
```

- public void Cancel() {

```
            MD.Cancel();
        }
```

- public void Premium() {

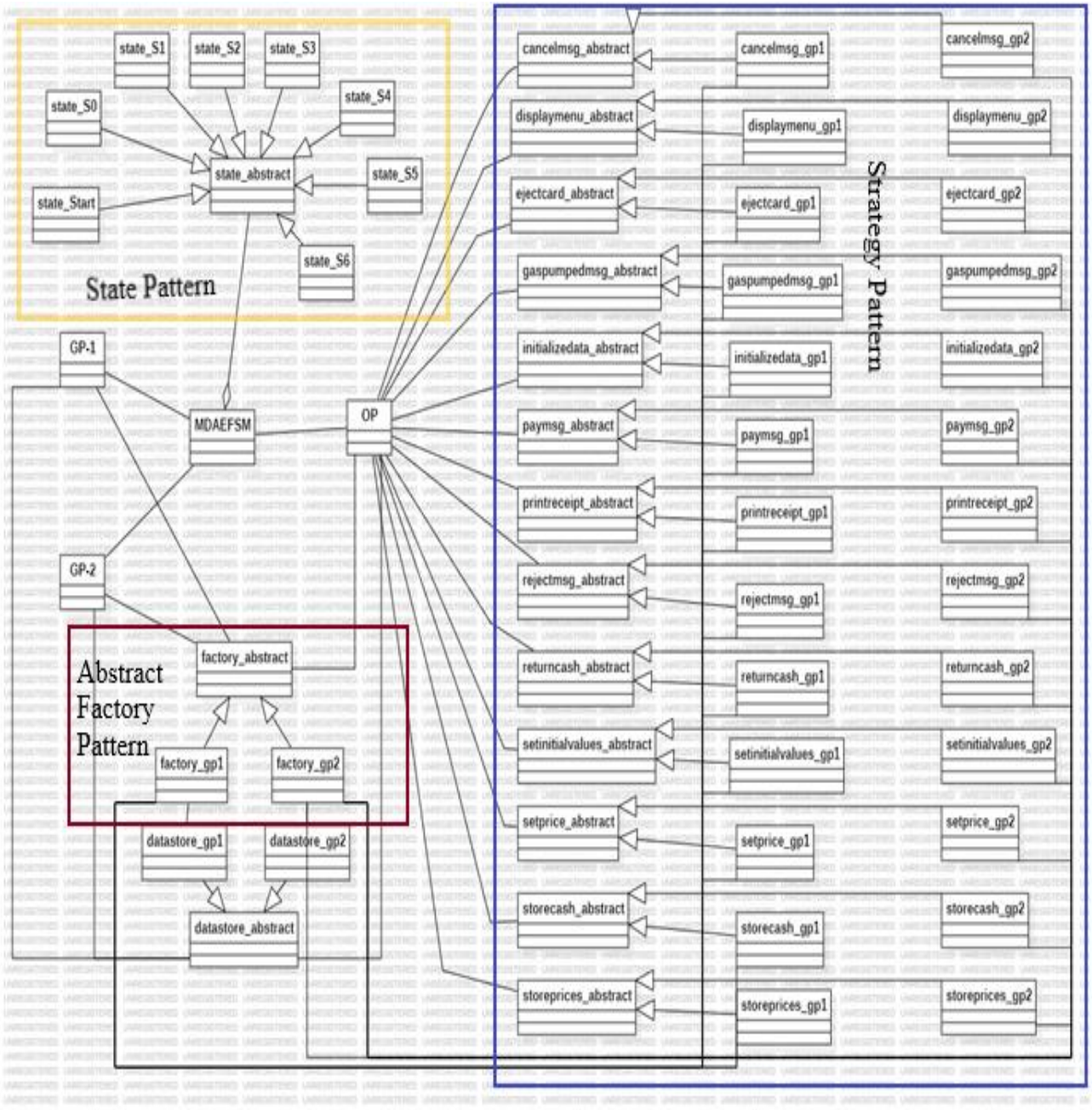
```
            if(DS.getprice()==0) {
            DS.setprice(DS.getFprice());
            MD.SelectGas(2);
            }
        }
```

- public void Regular() {

```
            if(DS.getprice()==0) {
            DS.setprice(DS.getRprice());
            MD.SelectGas(1);
            }
        }
```

- public void Diesel() {
  if(DS.getprice()==0) {
  DS.setprice(DS.getDprice());
  MD.SelectGas(3);
  }
  }
- public void StartPump() {
  DS.setG(0);
  DS.setTotal(0);
  MD.StartPump();
  }
- public void PumpGallon() {
  if (DS.getcash() > DS.getprice() * (DS.getG() + 1)) {
  DS.setG(DS.getG()+1);
  DS.setTotal(DS.getprice()*DS.getG());
  MD.PumpGallon();
  }
  else {
  MD.PumpGallon();
  }
  }
- public void Stop() {
  MD.Stop();
  }
- public void Receipt() {
  MD.Receipt();
  }
- public void NoReceipt() {
  MD.NoReceipt(); }

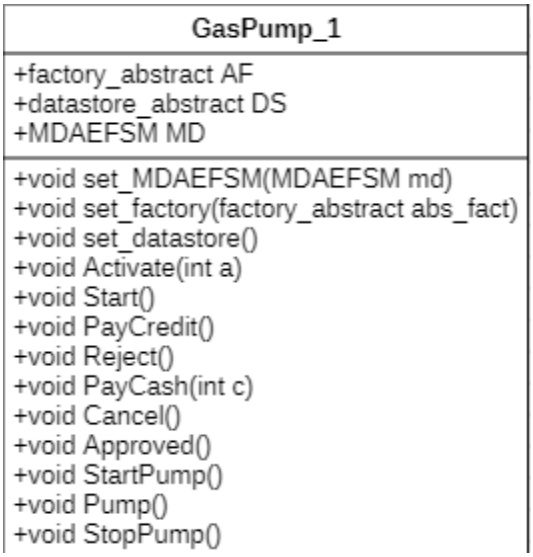
II. High-Level Class diagram(s) of the MDA of the GP components: State, Strategy & Abstract Factory Pattern



III. For each class in the Class Diagram(s): Purpose of the class & Responsibility of each operation supported by each class

Class Pattern

- GASPUMP_1



factory_abstract AF - store pointer to abstract factory for GasPump_1

datastore_abstract DS - store pointer to datastore for GasPump_1

MDAEFSM MD - store pointer to MDA-EFSM

void set_MDAEFSM(MDAEFSM md) - sets MDAEFSM pointer to object created in driver

void set_factory(factory_abstract abs_fact) - set GasPump_1 factory pointer to object created in driver

void set_datastore() - get datastore for GasPump_1 from abstract factory

void Activate(int a) – sets temp_a value & sets price value and invokes Activate in MDAEFSM

void Start() - invokes Start in MDAEFSM

void PayCredit() - invokes PayType in MDAEFSM with value 1 as argument

void Reject() – invokes Reject in MDAEFSM

void PayCash(int c) – sets temp_cash value & sets cash value invokes PayType in MDAEFSM with value 0 as argument

void Cancel() – Invokes Cancel on MDAEFSM

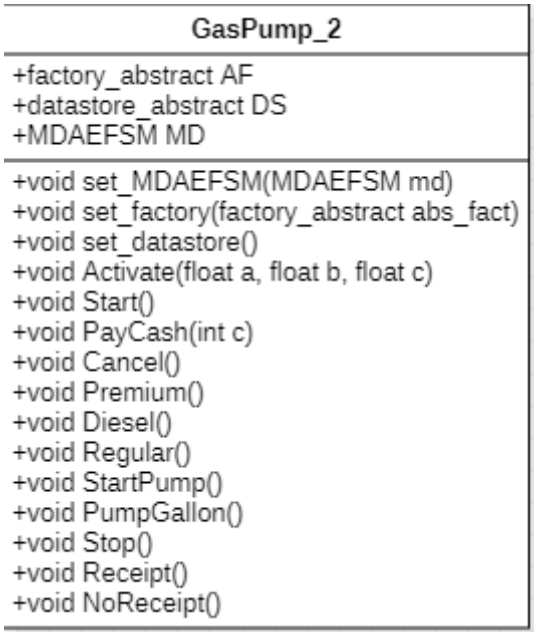
void Approved() – Invokes Approved on MDAEFSM

void StartPump() – StartPump on MDAEFSM

void Pump() – sets t & L values and invokes Pump on MDAEFSM

void StopPump() - Invokes StopPump and Receipt on MDAEFSM

- GASPUMP_2



factory_abstract AF - store pointer to abstract factory for GasPump_2

datastore_abstract DS - store pointer to datastore for GasPump_2

MDAEFSM MD - store pointer to MDA-EFSM

void set_MDAEFSM(MDAEFSM md) - sets MDAEFSM pointer to object created in driver

void set_factory(factory_abstract abs_fact) - set GasPump_2 factory pointer to object created in driver

void set_datastore() - get datastore for GP2 from abstract factory

void Activate(float a, float b, float c) – sets temp_a , temp_b and temp_c values & sets Rprice, Dprice and Fprice values & invokes Activate in MDAEFSM

void Start() – invokes Start in MDAEFSM

void PayCash(int c) – sets temp_cash value & sets cash value invokes PayType in MDAEFSM with value 0 as argument

void Cancel() – Invokes Cancel on MDAEFSM

void Diesel() – Invokes SelectGas on MDAEFSM with value 3

void Premium() – Invokes SelectGas on MDAEFSM with value 2

void Regular() – Invokes SelectGas on MDAEFSM with value 1

void StartPump() – Invokes StartPump on MDAEFSM

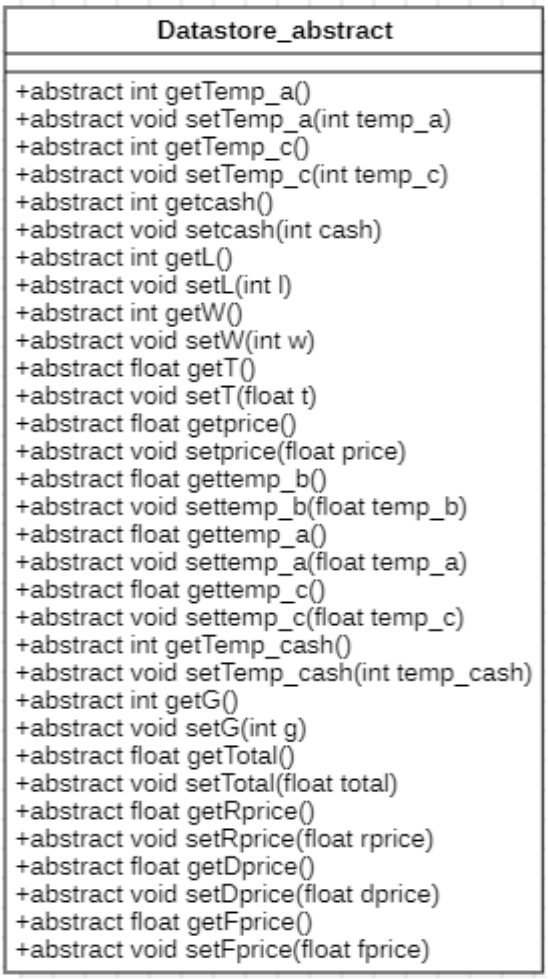
void PumpGallon() – sets G & Total values and invokes PumpGallon on MDAEFSM

void Stop() - Invokes Stop on MDAEFSM

void Receipt() - Invokes Receipt on MDAEFSM

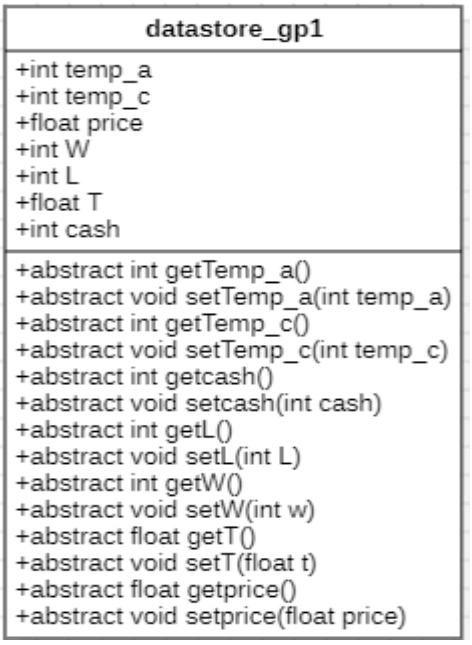
void NoReceipt() - Invokes NoReceipt on MDAEFSM

- DATASTORE_ABSTRACT



This is the data abstract class which the datastore for Gas Pump 1 and 2 inherit from. It contains abstract methods of both gas pumps. The functions are defined in datastore_gp1 and datastore_gp2.

- DATASTORE_GP1



temp_a, temp_c - temporary variables used by input processor

price – price of the gas

W – pay type selected by user

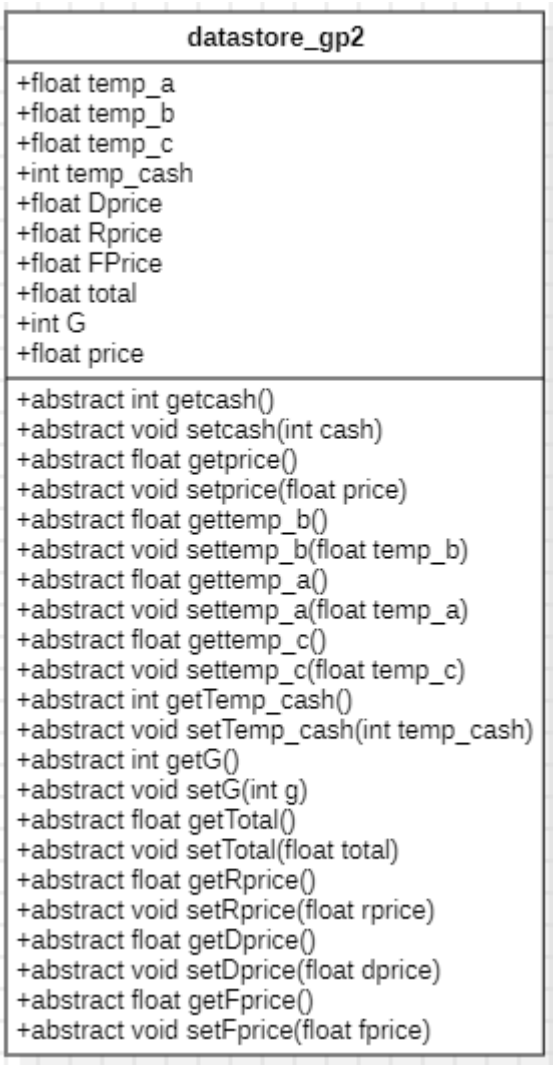
L – store quantity (liters)

T – total amount of the pump

cash – cash paid by user

This class contains a getter, setter function for all variables of GP1 since the variables are private, setter methods are used to store the value in the variables & getter methods are used to read the saved values.

- DATASTORE_GP2



temp_a,temp_b, temp_c - temporary variables used by input processor

Dprice – Diesel gas price

RPrice – Regular gas price

FPrice – Premium gas price

price – price of the gas selected

G – store quantity (gallons)

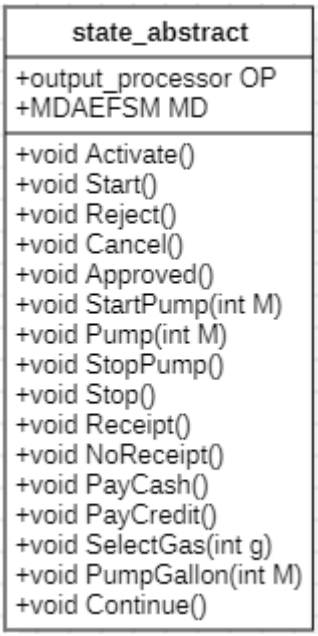
total – total amount of the pump

temp_cash – cash paid by user

This class contains a getter & setter function for all variables of GP2 since the variables are private, setter methods are used to store the value in the variables & getter methods are used to read the saved values.

a. State Pattern

- STATE_ABSTRACT



OP – pointer to the output processor which is set during the creation of the instance of the state
MD – pointer to MDAEFSM object (decentralized version)
The abstract state class contains abstract methods for which the concrete definition is specified in the respective state classes for which the action is valid. The invalid functions for any state are empty.

- STATE_LIST
  - State_Start

state_start
void Activate()

Activate( ) - invokes StorePrices on output processor and changes state to 0.

  - State_S0

state_S0
void Start()

Start( ) - invokes PayMsg on output processor and changes state to 1.

  - State_S1

state_S1
void PayCash() void PayCredit()

PayCash() - invokes StoreCash and DisplayMenu on output processor and changes state to 3

PayCredit() – changes state to 2

- State_S2

state_S2
void Reject() void Approved()

Approved() - invokes DisplayMenu on output processor and changes state to 3.
Reject() - invokes RejectMsg on output processor and changes state to 0.

- State_S3

state_S3
void Cancel() void SelectGas() void StartPump(int M) void Continue()

Cancel() – invokes CancelMsg on output processor and changes state to 0.
SelectGas() - invokes SetPrice on output processor.
StartPump(int M) - invokes SetInitialValues on output processor and calls Continue in MDAEFSM.
Continue() – changes state to 4.

- State_S4

state_S4
void StartPump(int M) void Pump(int M) void StopPump()

StartPump(int M) - invokes SetInitialValues on output processor and changes state to 5

PumpGallon(int M) - invokes GasPumpedMsg on output processor.

StopPump() - invokes PrintReceipt on output processor and changes state to 0.

- State_S5

state_S5
void PumpGallon(int M) void Stop()

PumpGallon(int M) - invokes GasPumpedMsg on output processor.

Stop() - changes state to 6.

- State_S6

state_S6
void Receipt() void NoReceipt()

Receipt() - invokes PrintReceipt and ReturnCash on output processor and changes state to 0.
NoReceipt() - invokes ReturnCash on output processor and changes state to 0.

b. Strategy Pattern:

- CancelMsg
  - Cancelmsg_abstract

cancelmsg_abstract
abstract void CancelMsg()

This class stores the abstract method 'CancelMsg' for which the concrete methods are defined in inherited classes, cancelmsg_gp1 and cancelmsg_gp2.

  - Cancelmsg_gp1

cancelmsg_gp1
void CancelMsg()

CancelMsg() - This method displays the message "Transaction Cancelled" for GP1.

  - Cancelmsg_gp2

cancelmsg_gp2
void CancelMsg()

CancelMsg() - This method displays the message "Transaction Cancelled" for GP2.

- DisplayMenu
  - Displaymenu_abstract

displaymenu_abstract
abstract void DisplayMenu()

This class stores the abstract method DisplayMenu for which the concrete methods are defined in inherited classes, displaymenu_gp1 and displaymenu_gp2.

  - Displaymenu_gp1

displaymenu_gp1

void DisplayMenu()

DisplayMenu() - This method doesn't have to display any gas selection option

- Displaymenu_gp2

displaymenu_gp2
void DisplayMenu()

DisplayMenu() - This method displays the menu for gas selection in GP2(Regular, Diesel or Premium).

- EjectCard
  - Ejectcard_abstract

Ejectcard_abstract
abstract void EjectCard()

This class stores the abstract method EjectCard for which the concrete methods are defined in inherited classes, Ejectcard_gp1 and Ejectcard_gp2.

  - Ejectcard_gp1

Ejectcard_gp1
void EjectCard()

EjectCard() - This method displays the message "Card is Ejected" for GP1.

  - Ejectcard_gp2

Ejectcard_gp2
void EjectCard()

EjectCard() - This method displays the message "Card is Ejected" for GP2.

- GasPumpedMsg
  - Gaspumpedmsg_abstract

gaspumpedmsg_abstract
abstract void GasPumpedMsg()

This class stores the abstract method GasPumpedMsg for which the concrete methods are defined in inherited classes, Gaspumpedmsg_gp1 and Gaspumpedmsg_gp2.

- Gaspumpedmsg_gp1

gaspumpedmsg_gp1
void GasPumpedMsg()

GasPumpedMsg - This class displays the quantity of gas pumped(liters) and the total amount for GP1.

- Gaspumpedmsg_gp2

gaspumpedmsg_gp2
void GasPumpedMsg()

GasPumpedMsg - This class displays the quantity of gas pumped(gallons) and the total amount for GP2.

- InitializeData
  - Initializedata_abstract

initializedata_abstract
abstract void InitializeData()

This class stores the abstract method InitializeData for which the concrete methods are defined in inherited classes, Initializedata_gp1 and Initializedata_gp2.

  - Initializedata_gp1

initializedata_gp1
void InitializeData()

InitializeData - This method initializes the cash to 0 for GP1.

  - Initializedata_gp2

initializedata_gp2
void InitializeData()

InitializeData - This method initializes the cash to 0 for GP2.

- PayMsg
  - Paymsg_abstract

Paymsg_abstract
abstract void PayMsg()

This class stores the abstract method PayMsg for which the concrete methods are defined in inherited classes, Paymsg_gp1 and Paymsg_gp2.

- Paymsg_gp1

Paymsg_gp1
void PayMsg()

PayMsg - This method displays the list of payment options GP1(Credit or Cash).

- Paymsg_gp2

Paymsg_gp2
void PayMsg()

PayMsg - This method displays the list of payment options GP2(Cash).

- PrintReceipt
  - Printreceipt_abstract

printreceipt_abstract
abstract void PrintReceipt()

This class stores the abstract method PrintReceipt for which the concrete methods are defined in inherited classes, Printreceipt_gp1 and Printreceipt_gp2.

- Printreceipt_gp1

printreceipt_gp1
void PrintReceipt()

Print Receipt - This method displays the receipt with the total amount for GP1.

- Printreceipt_gp2

printreceipt_gp2
void PrintReceipt()

Print Receipt - This method displays the receipt with the total amount for GP2.

- RejectMsg
  - Rejectmsg_abstract

rejectmsg_abstract
abstract void RejectMsg()

This class stores the abstract method RejectMsg for which the concrete methods are defined in inherited classes, Rejectmsg_gp1 and Rejectmsg_gp2.

- Rejectmsg_gp1

rejectmsg_gp1
void RejectMsg()

RejectMsg - This method displays credit card rejection message for GP1.

- Rejectmsg_gp2

rejectmsg_gp2
void RejectMsg()

RejectMsg - This method displays credit card rejection message for GP2.

- ReturnCash
  - Returncash_abstract

returncash_abstract
abstract void ReturnCash()

This class stores the abstract method ReturnCash for which the concrete methods are defined in inherited classes, Returncash_gp1 and Returncash_gp2.

- Returncash_gp1

returncash_gp1
void ReturnCash()

ReturnCash - This method is empty for GP1

- Returncash_gp2

returncash_gp2
void ReturnCash()

ReturnCash - This method displays the cash to be returned for GP2.

- SetInitialValues
  - Setinitialvalues_abstract

setinitialvalues_abstract
abstract void SetInitialValues()

This class stores the abstract method SetInitialValues for which the concrete methods are defined in inherited classes, Setinitialvalues_gp1 and Setinitialvalues_gp2.

  - Setinitialvalues_gp1

setinitialvalues_gp1
void SetInitialValues()

SetInitialValues - This method sets L and total to 0 to start measuring the quantity of pump and amount in GP1.

  - Setinitialvalues_gp2

setinitialvalues_gp2
void SetInitialValues()

SetInitialValues - This method sets G and total to 0 to start measuring the quantity of pump and amount in GP1.

- SetPrice
  - Setprice_abstract

setprice_abstract
abstract void SetPrice()

This class stores the abstract method SetPrice for which the concrete methods are defined in inherited classes, Setprice_gp1 and Setprice_gp2.

  - Setprice_gp1

setprice_gp1
void SetPrice()

SetPrice – This method is not used in GP1.

  - Setprice_gp2

setprice_gp2

void SetPrice()

SetPrice - This method sets the price based on gas selection in GP2.

- StoreCash
  - Storecash_abstract

storecash_abstract
abstract void StoreCash()

This class stores the abstract method StoreCash for which the concrete methods are defined in inherited classes, Storecash_gp1 and Storecash_gp2.

  - Storecash_gp1

storecash_gp1
void StoreCash()

StoreCash – This method sets value of cash provided by user in GP1.

  - Storecash_gp2

storecash_gp2
void StoreCash()

StoreCash – This method sets value of cash provided by user in GP2.

- StorePrices
  - Storeprices_abstract

storeprices_abstract
abstract void StorePrices()

This class stores the abstract method StorePrices for which the concrete methods are defined in inherited classes, Storeprices_gp1 and Storeprices_gp2.

  - Storeprices_gp1

storeprices_gp1
void StorePrices()

StorePrices - This method stores the price of gas during activation in GP1.

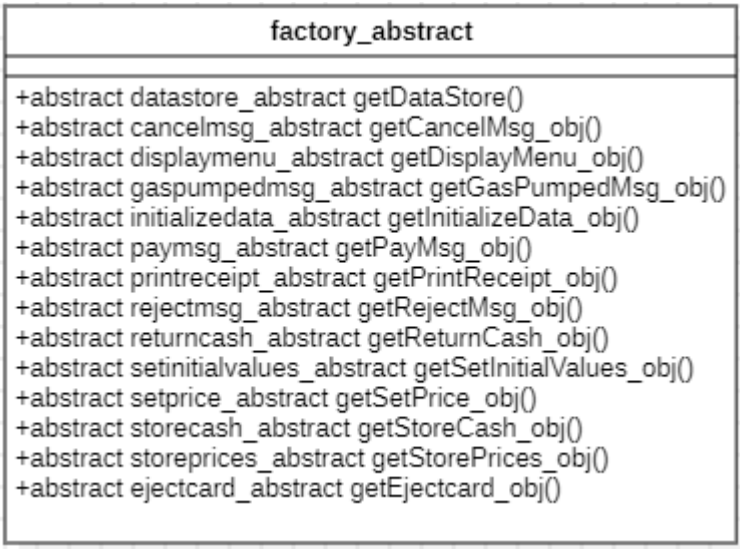
  - Storeprices_gp2

storeprices_gp1
void StorePrices()

StorePrices - This method stores the prices of gas during activation in GP2.

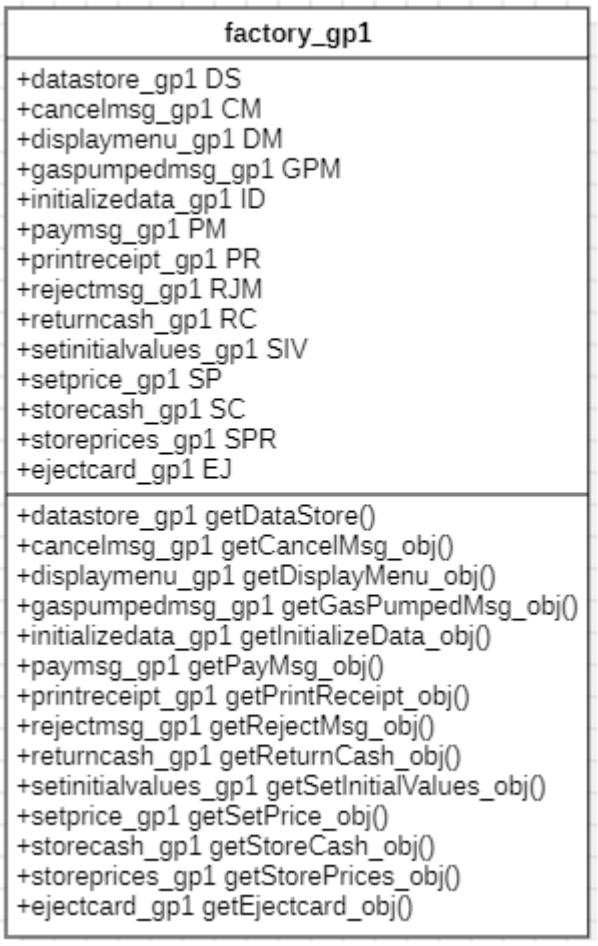
c. Abstract Factory Pattern:

- FACTORY ABSTRACT



This class contains abstract getter methods for which the concrete methods are specified in the inherited classes, factory_gp1 and factory_gp2.

- FACTORY_GP1

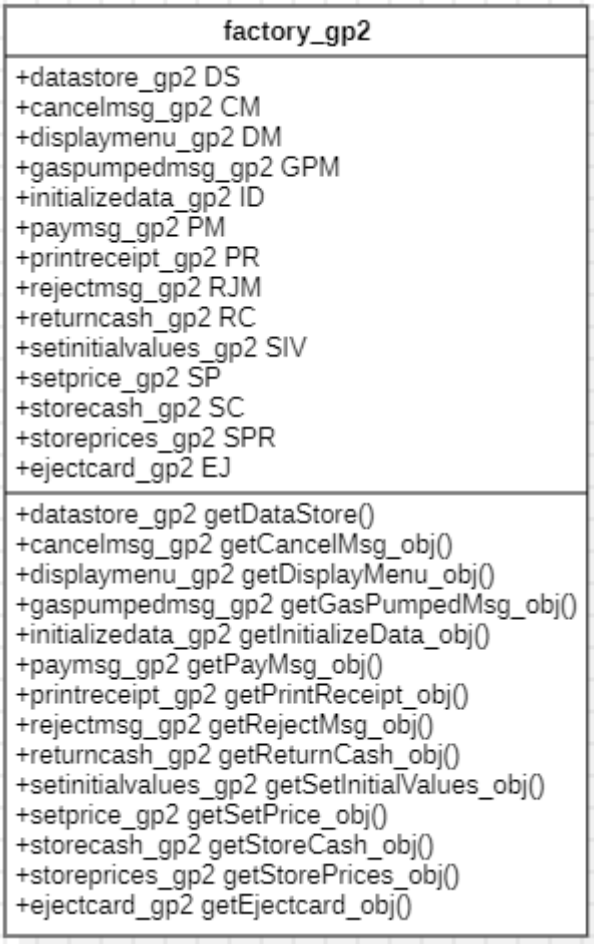


DS – pointer to the datastore for GP1

CM, DM, GPM, ID, PM, PR, RJM, RC, SIV, SP, SC, SPR, EJ – pointers to various action classes for GP1.

This class contains concrete getter methods that return the action class object for GP1 when requested for by the output processor. If no object exists, it creates a new one and saves the pointer in the factory and returns the pointer to that which is then used the next time the OP requests for the same object.

- FACTORY_GP2

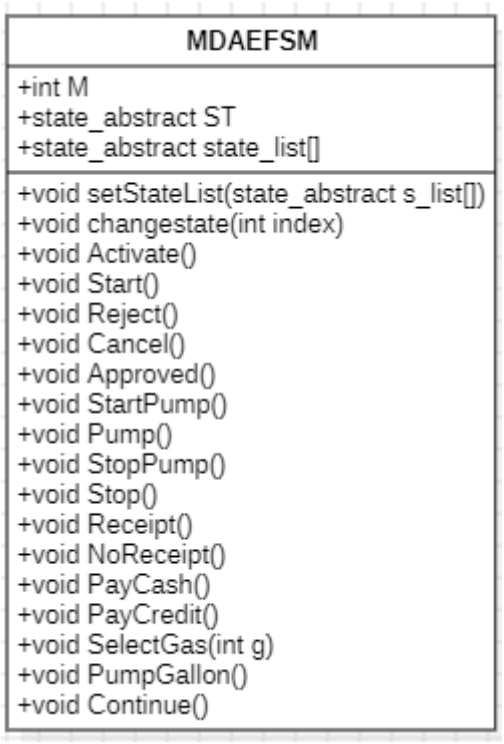


DS – pointer to the datastore for GP2

CM, DM, GPM, ID, PM, PR, RJM, RC, SIV, SP, SC, SPR, EJ – pointers to various action classes for GP2.

This class contains concrete getter methods that return the action class object for GP2 when requested by the output processor. If no object exists, it creates a new one saves the pointer in the factory, and returns the pointer to that which is then used the next time the OP requests for the same object.

- MDA-EFSM



M – stores 0 if cash payment or else stores 1
ST – pointer to the current state
state_list[] – stores the list of states

void setStateList(state_abstract s_list[]) – sets the list of states and sets the pointer ST to the first state

void changestate(int index) – changes state to index value(decentralized)

void Activate() – invokes Activate in the current state

void Start() – sets value of M to 1 and invokes Start in the current state

void Reject() – invokes Reject in the current state

void Cancel() – invokes Cancel in the current state

void Approved() – invokes Approved in the current state

void StartPump() – invokes StartPump in the current state

void Pump() – invokes Pump in the current state

void StopPump() – invokes StopPump in the current state

void Stop – invokes stop in the current state

void Receipt() – invokes Receipt in the current state

void NoReceipt() – invokes NoReceipt in the current state

void PayCash() – invokes PayCash in the current state

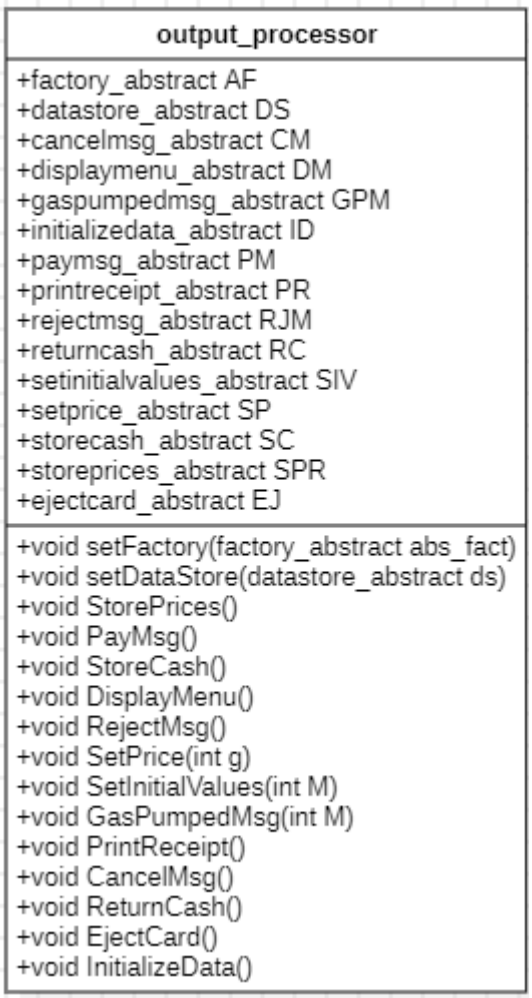
void PayCredit – invokes PayCredit in the current state

void SelectGas(int g) – invokes SelectGas in the current state

void PumpGallon – invokes PumpGallon in the current state

void Continue() – invokes Continue in the current state

- OUTPUT_PROCESSOR



AF – pointer to abstract factory 1 or 2 depending on the pump selection

DS – pointer to the datastore 1 or 2 depending on the pump selection

CM, DM, GPM, ID, PM, PR, RJM, RC, SIV, SP, SC, SPR, EJ – pointers to various action classes for GP1 or GP2.

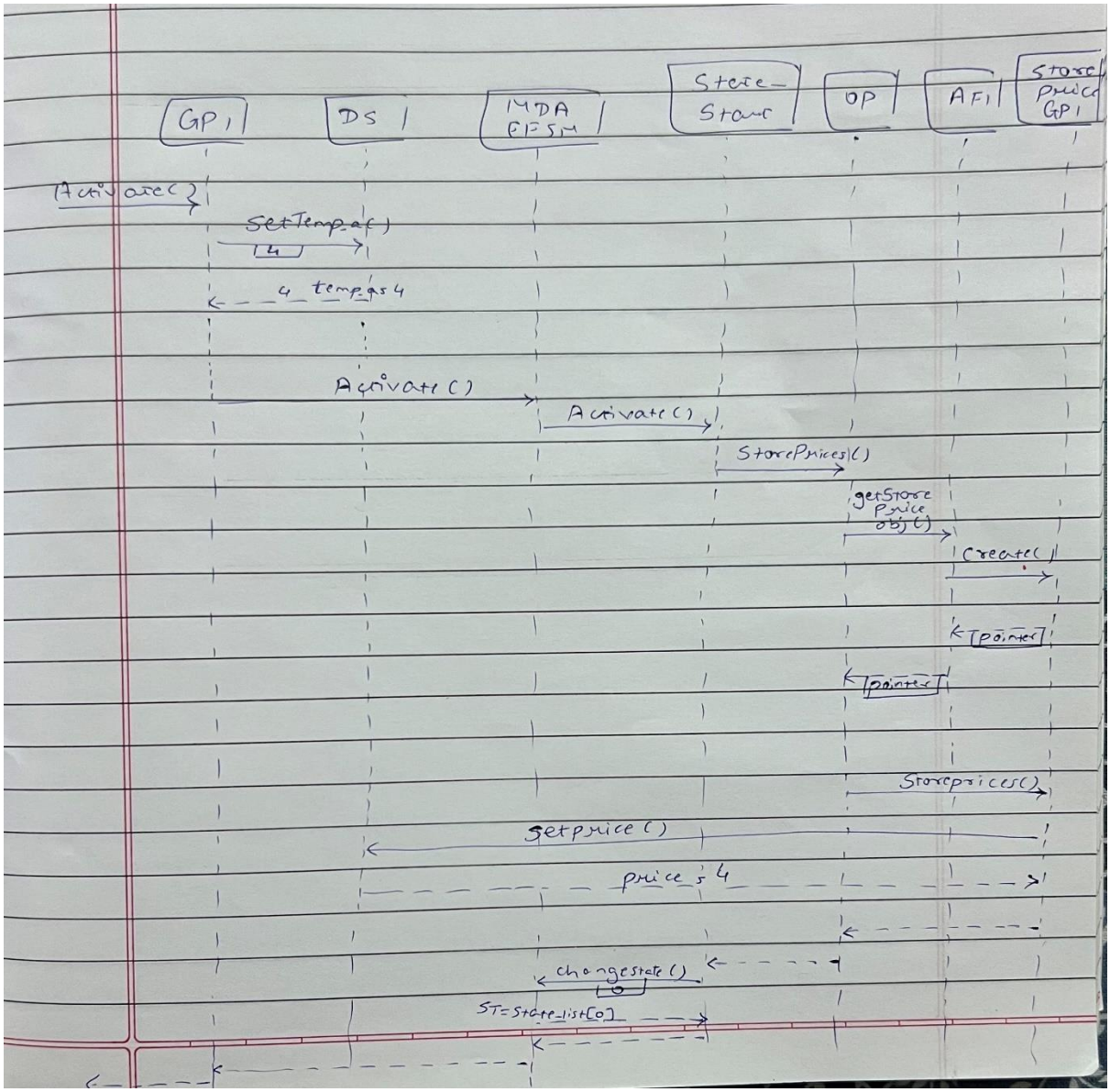
void setFactory(factory_abstract abs_fact) – set pointer to the factory object of GP1 or GP2 from the driver

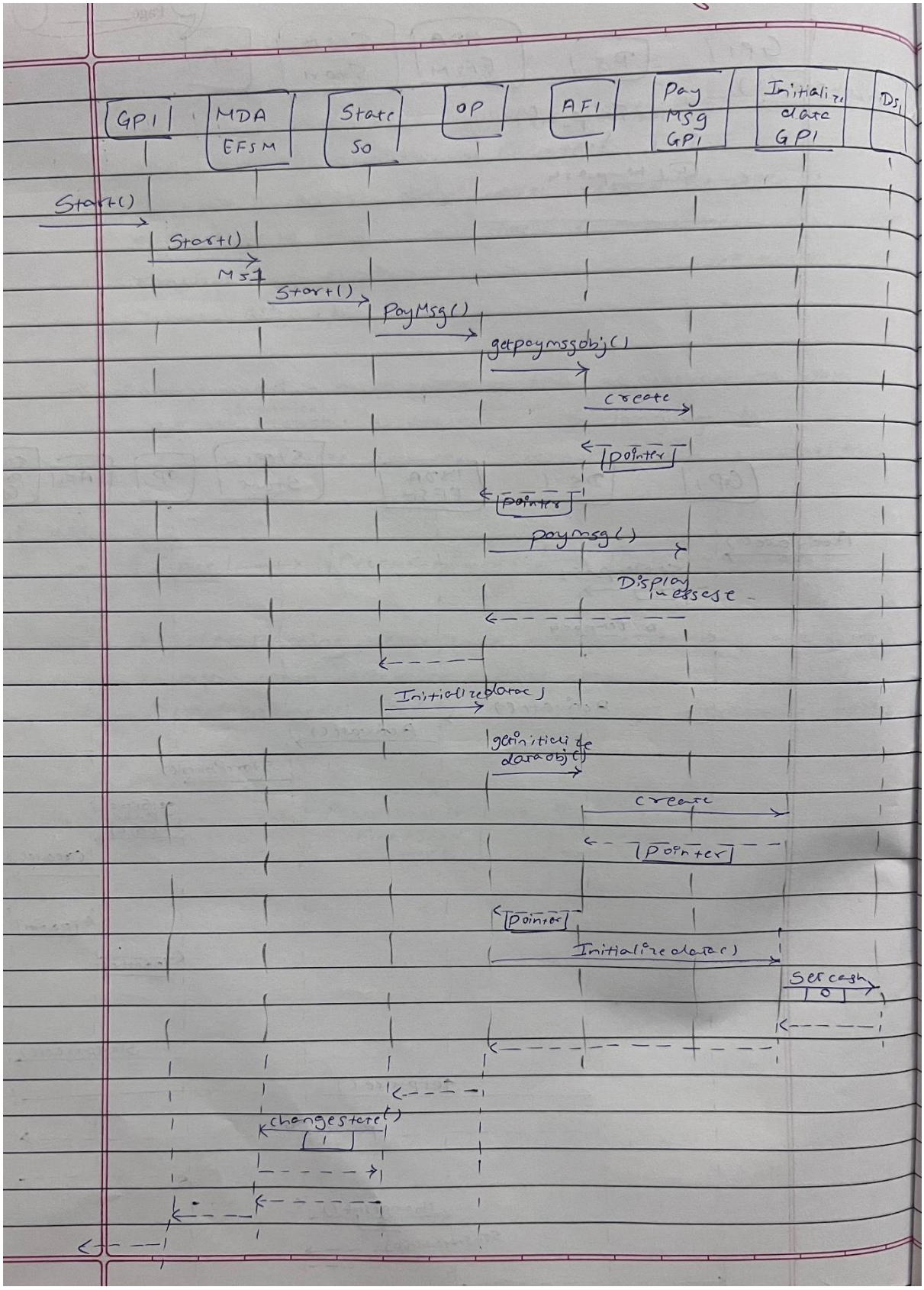
void setDataStore(datastore_abstract ds) – set pointer to the datastore object of GP1 or GP2 from the driver

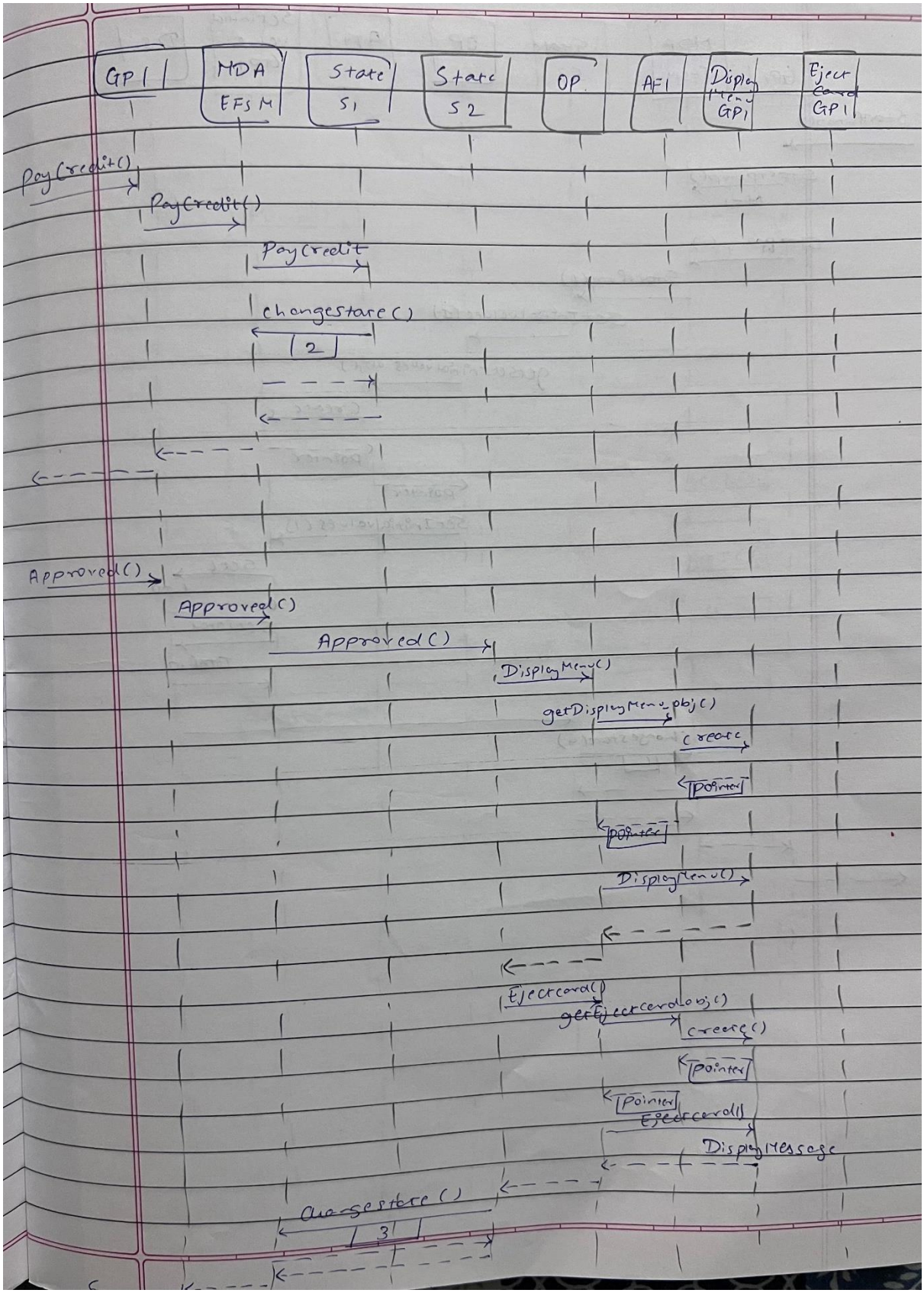
The methods get the pointers to the respective the action classes from the abstract factory and call the method to perform some specific action.

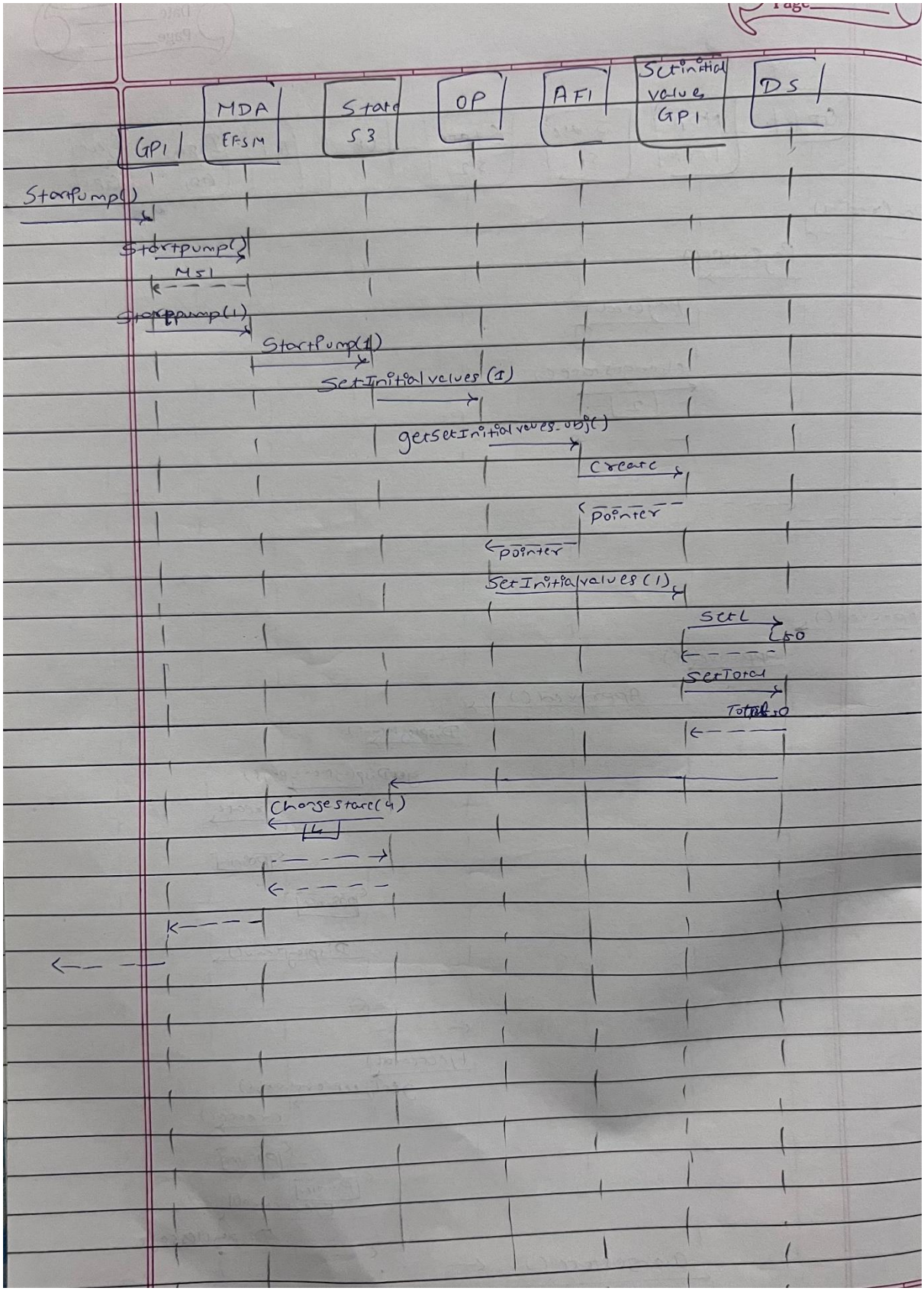
IV. Sequence Diagram

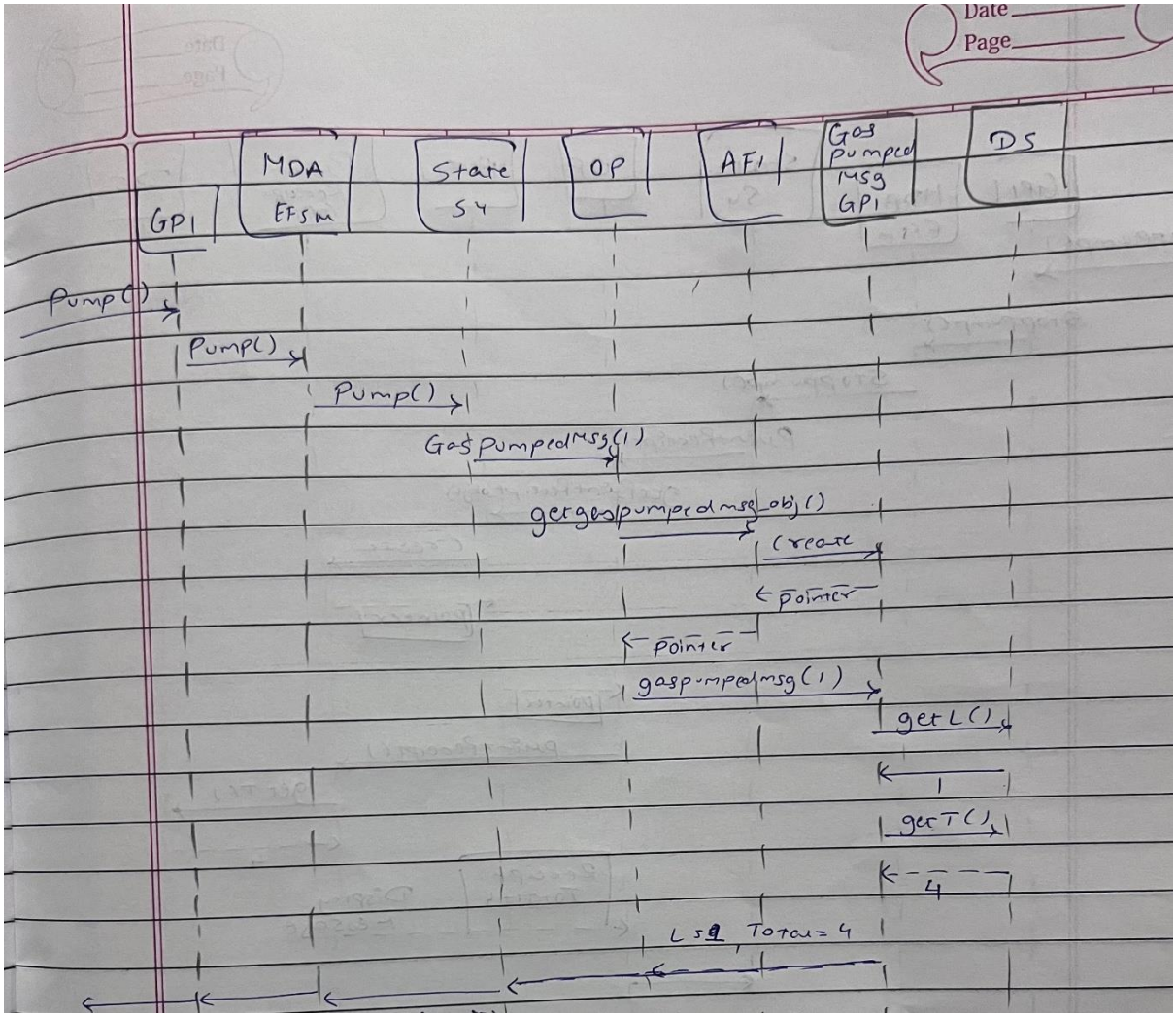
a. Activate(4), Start(), PayCredit(), Approved(), StartPump(), Pump(), StopPump()

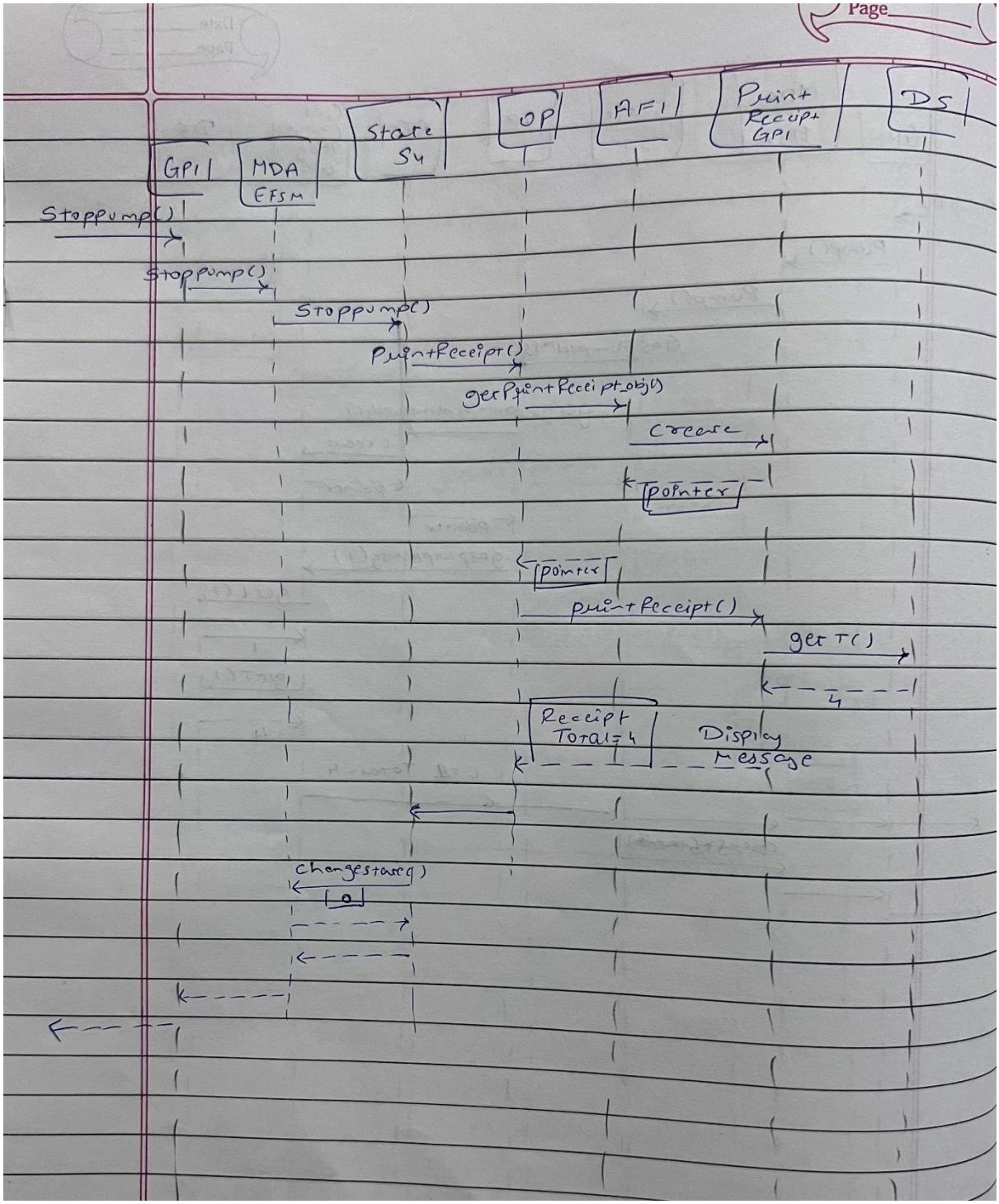



GPI
MDA
EFSM
State
So
OP
AFI
Pay
MSG
GPI
Initialize
data
GPI
DS
Start()
Start()
MSG
Start()
PayMsg()
getpaymsgobj()
create
pointer
pointer
paymsg()
Display message
Initialize data()
getinitialize data obj()
create
pointer
pointer
Initialize data()
Search
changestate()

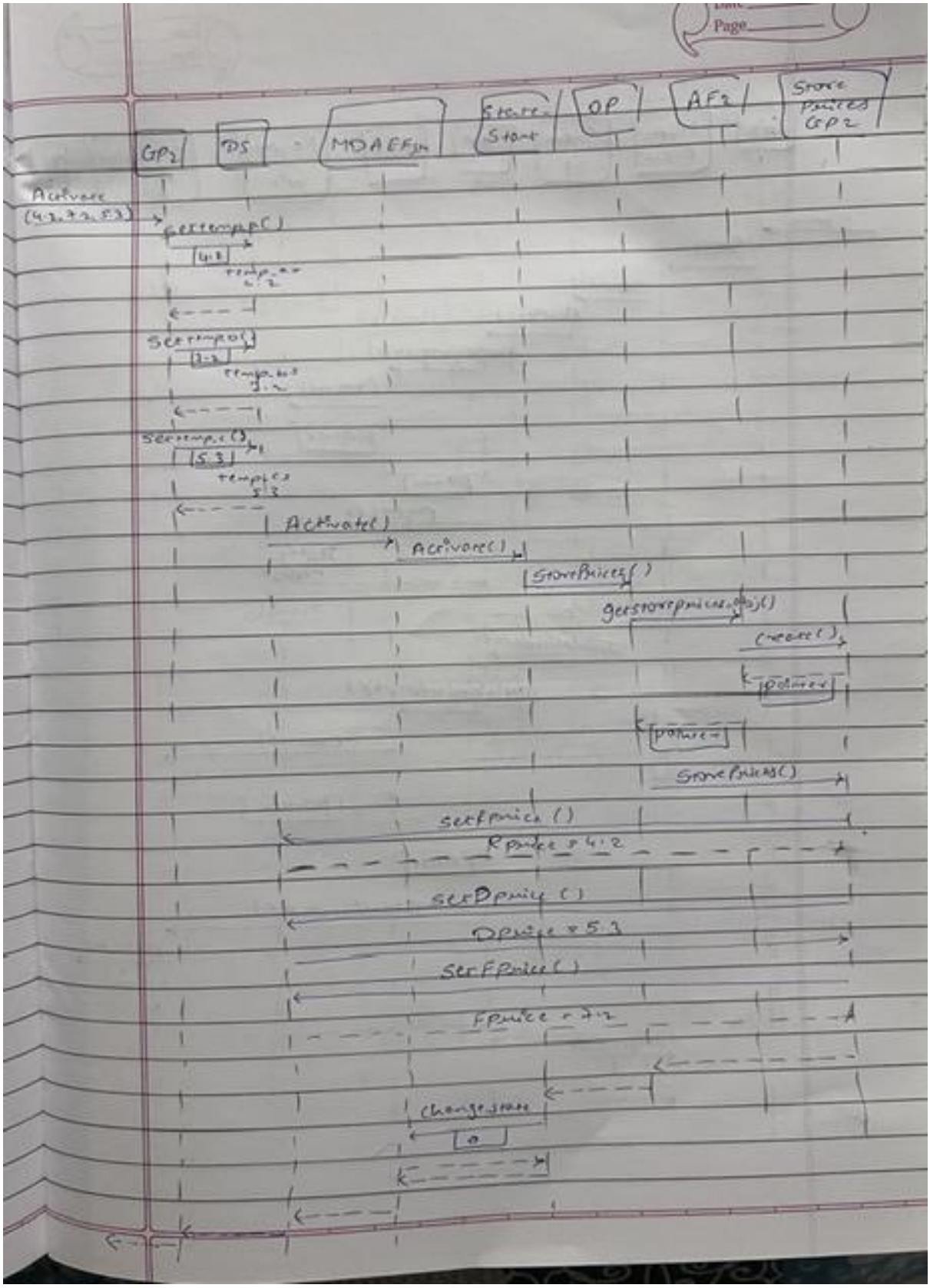

GP1
MDA
EFSM
State
S1
State
S2
OP
AF1
Display
GP1
Eject
GP1
PayCredit()
PayCredit()
PayCredit
changestate()
2
Approved()
Approved()
Approved()
DisplayMenu()
getDisplayMenuObj()
create
pointer
pointer
DisplayMenu()
EjectCard()
getEjectCardObj()
create()
pointer
Pointer
EjectCard()
DisplayMessage
changestate()
3

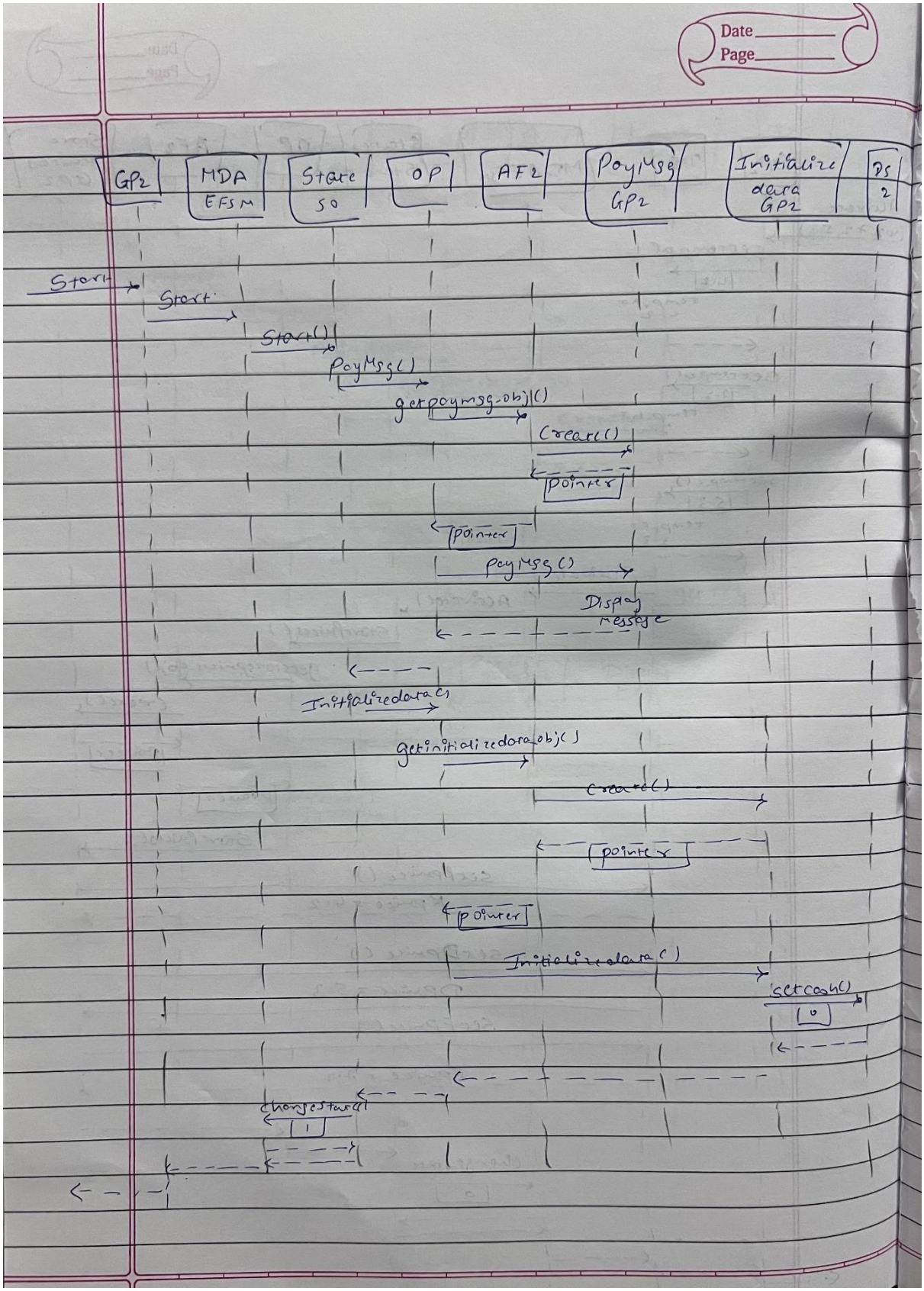

GPI
MDA
EFSM
State
S3
OP
AFI
SetInitial values
GPI
DS
StartPump()
Startpump()
MSI
StartPump(1)
SetInitialvalues (I)
getSetInitialvalues_obj()
Create
pointer
pointer
SetInitialvalues (1)
SetL
50
SetTorq
ChangeState(4)

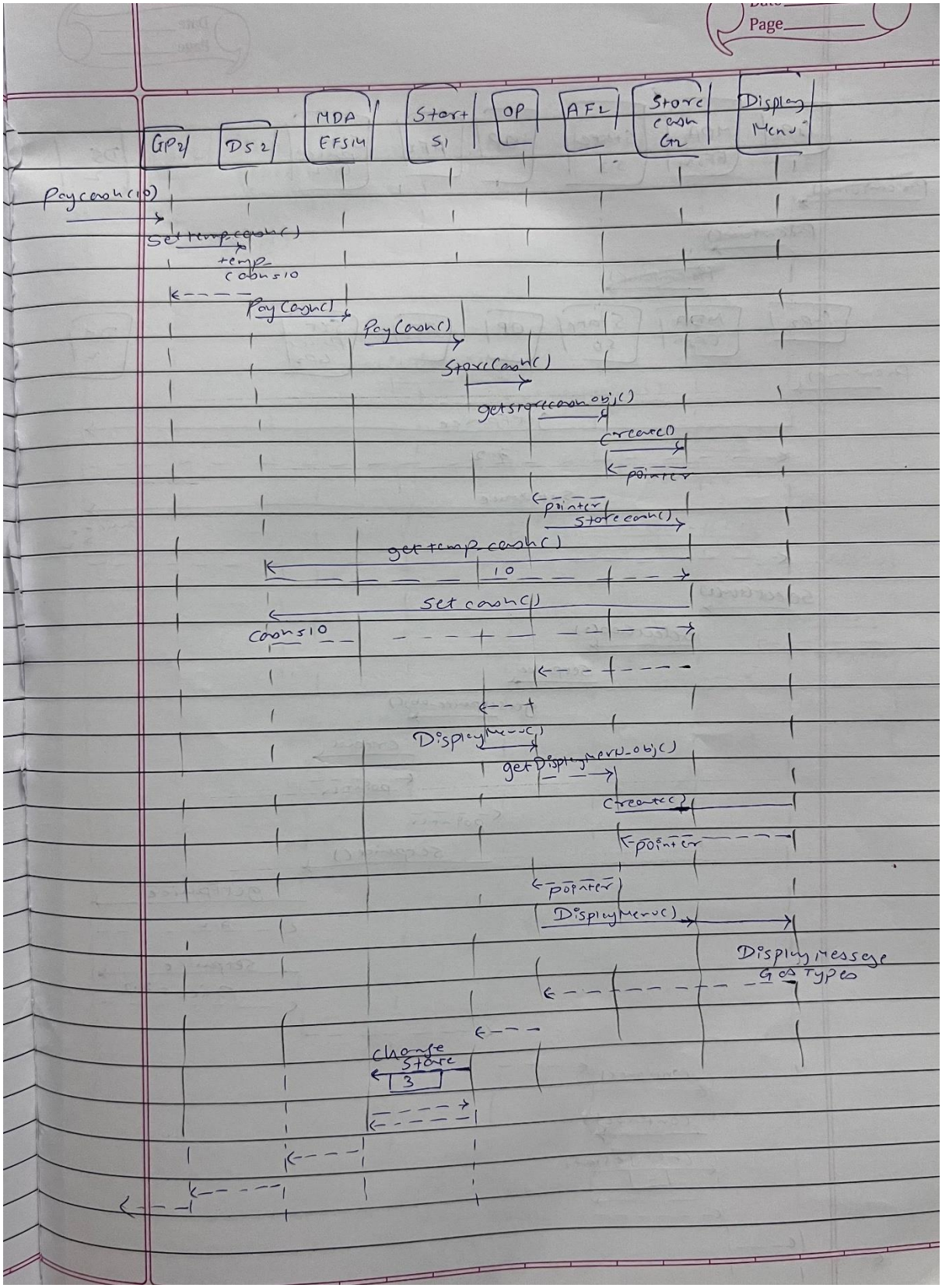

Date
Page
GPI
MDA
EFSM
State
S4
OP
AFI
Gas Pumped Msg GPI
DS
Pump()
Pump()
Pump()
GasPumpedMsg()
gergaspumpedmsg_obj()
create
pointer
pointer
gaspumpedmsg()
getL()
1
getT()
4
L=4 Total=4

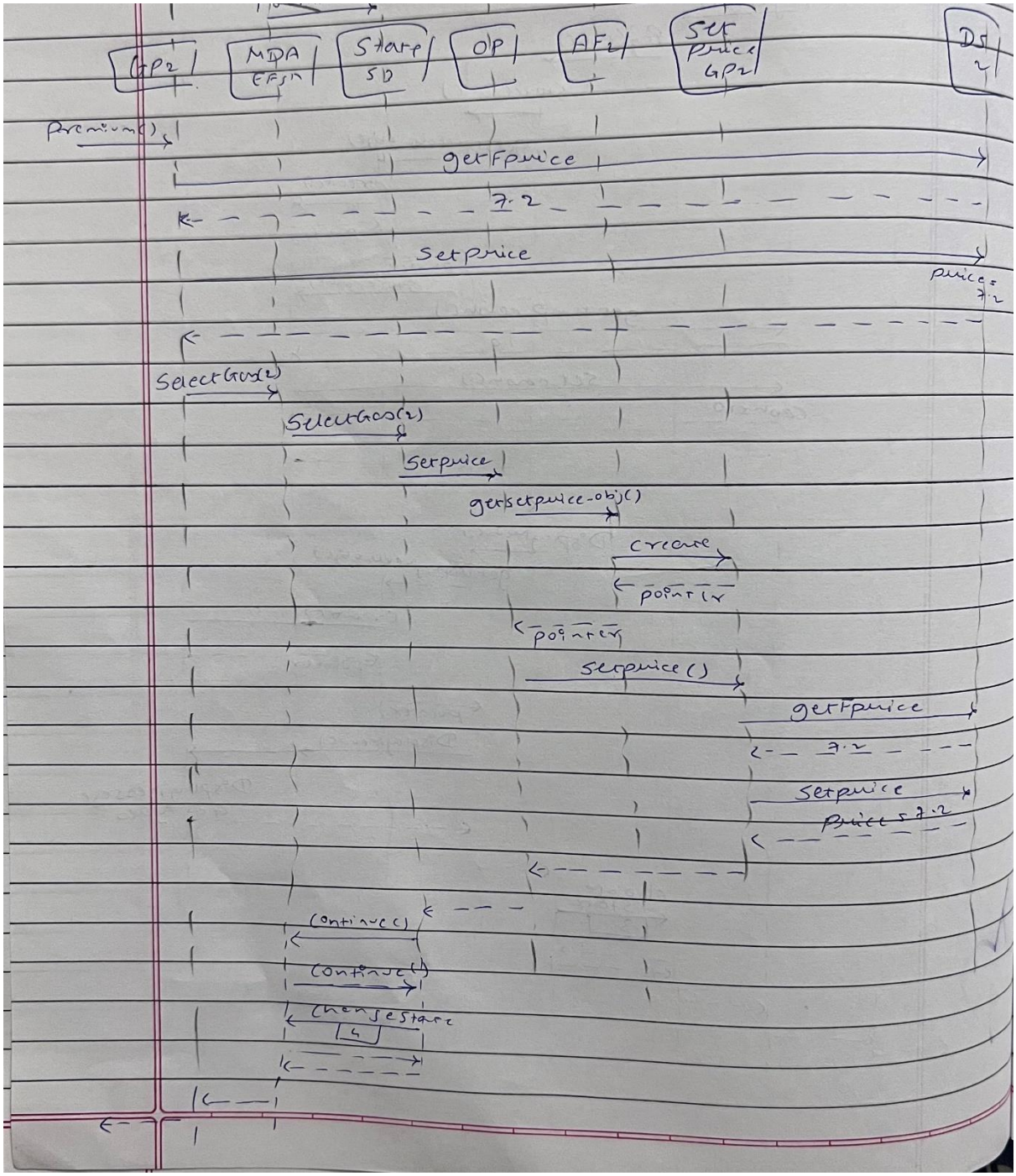

GPI
MDA EFSM
State S4
OP
AFI
Print Receipt GPI
DS
StopPump()
StopPump()
StopPump()
PrintReceipt()
getPrintReceipt_obj()
Create
pointer
pointer
printReceipt()
getT()
4
Receipt Total=4
Display Message
changeState()
0

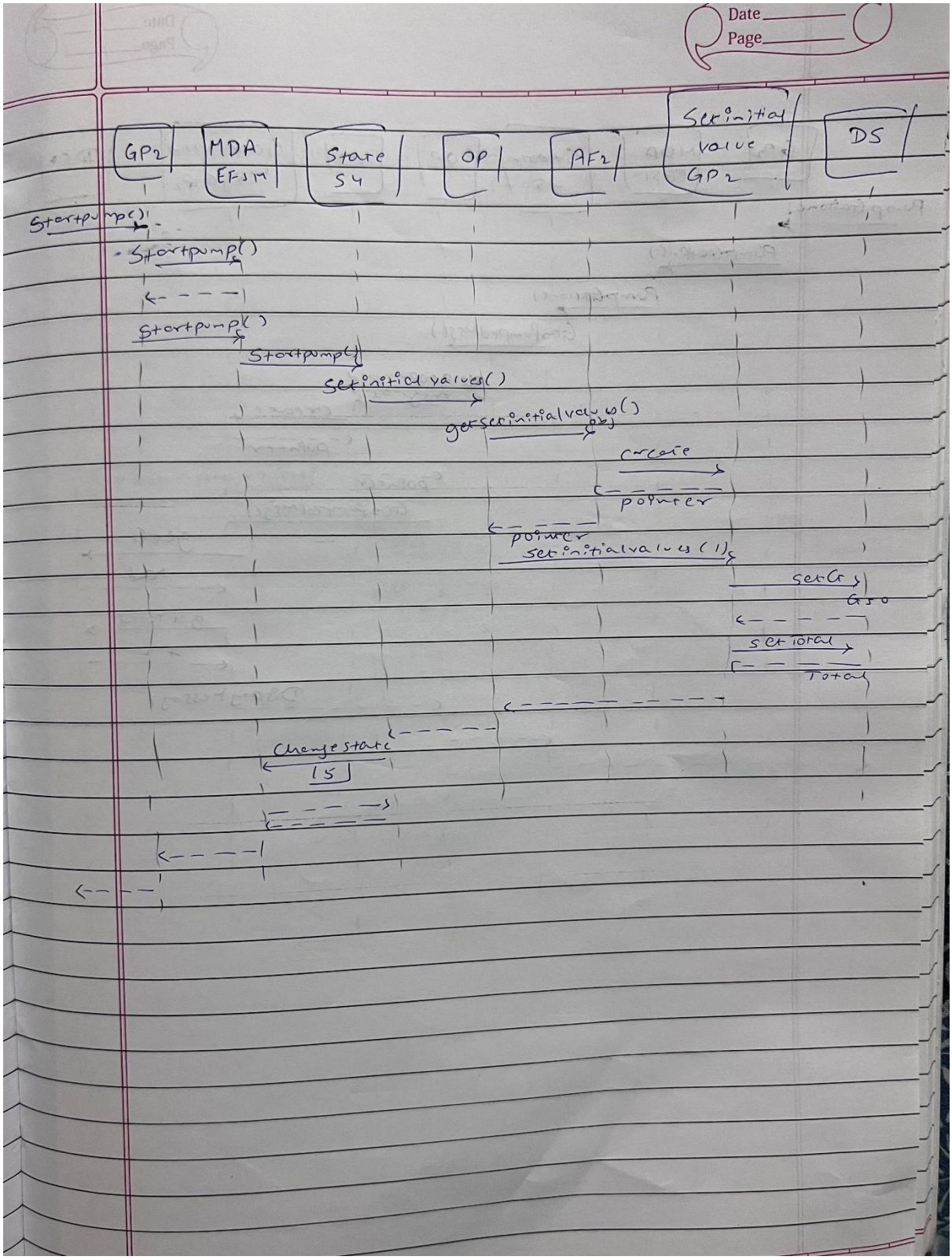
b. Activate(4.2, 7.2, 5.3), Start(), PayCash(10), Premium(), StartPump(), PumpGallon(), PumpGallon(), Receipt()

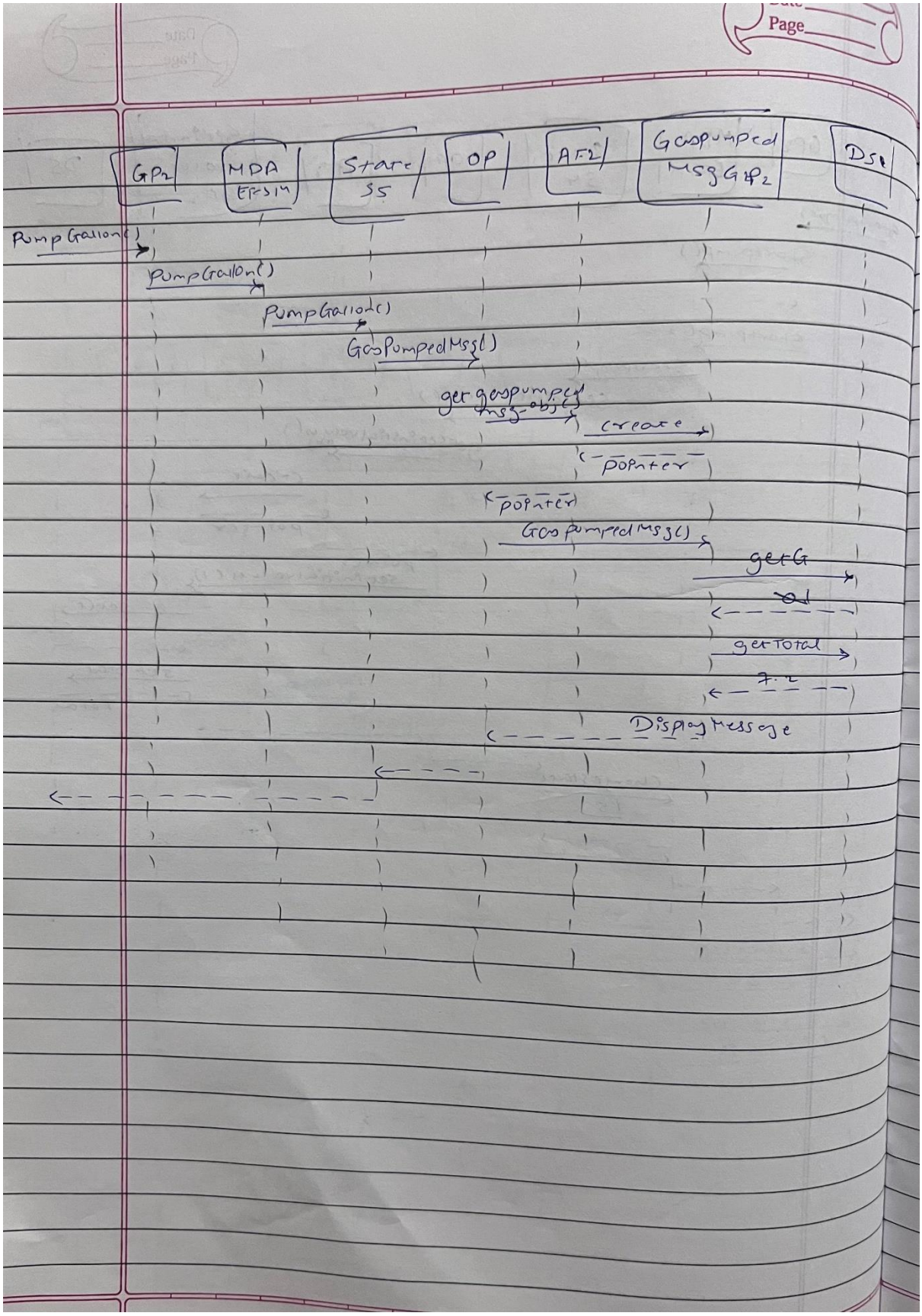


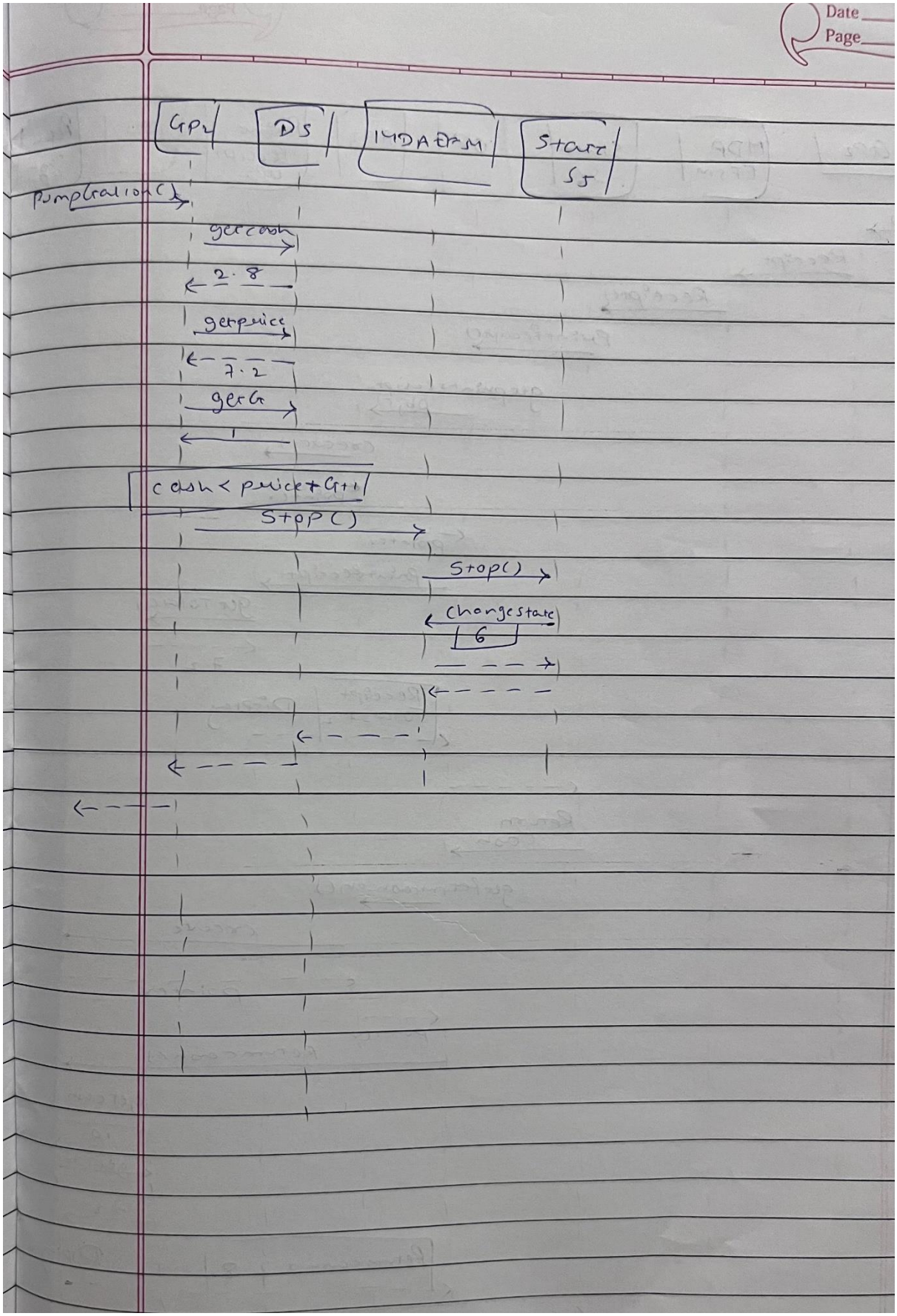

GP2
MDA EFSM
State SO
OP
AF2
PayMsg GP2
Initialize data GP2
DS 2
Start
Start
Start()
PayMsg()
getpaymsg_obj()
create()
pointer
pointer
PayMsg()
Display message
Initializedata()
getinitializedata_obj()
create()
pointer
pointer
Initializedata()
setcash()
0
changestate()
1

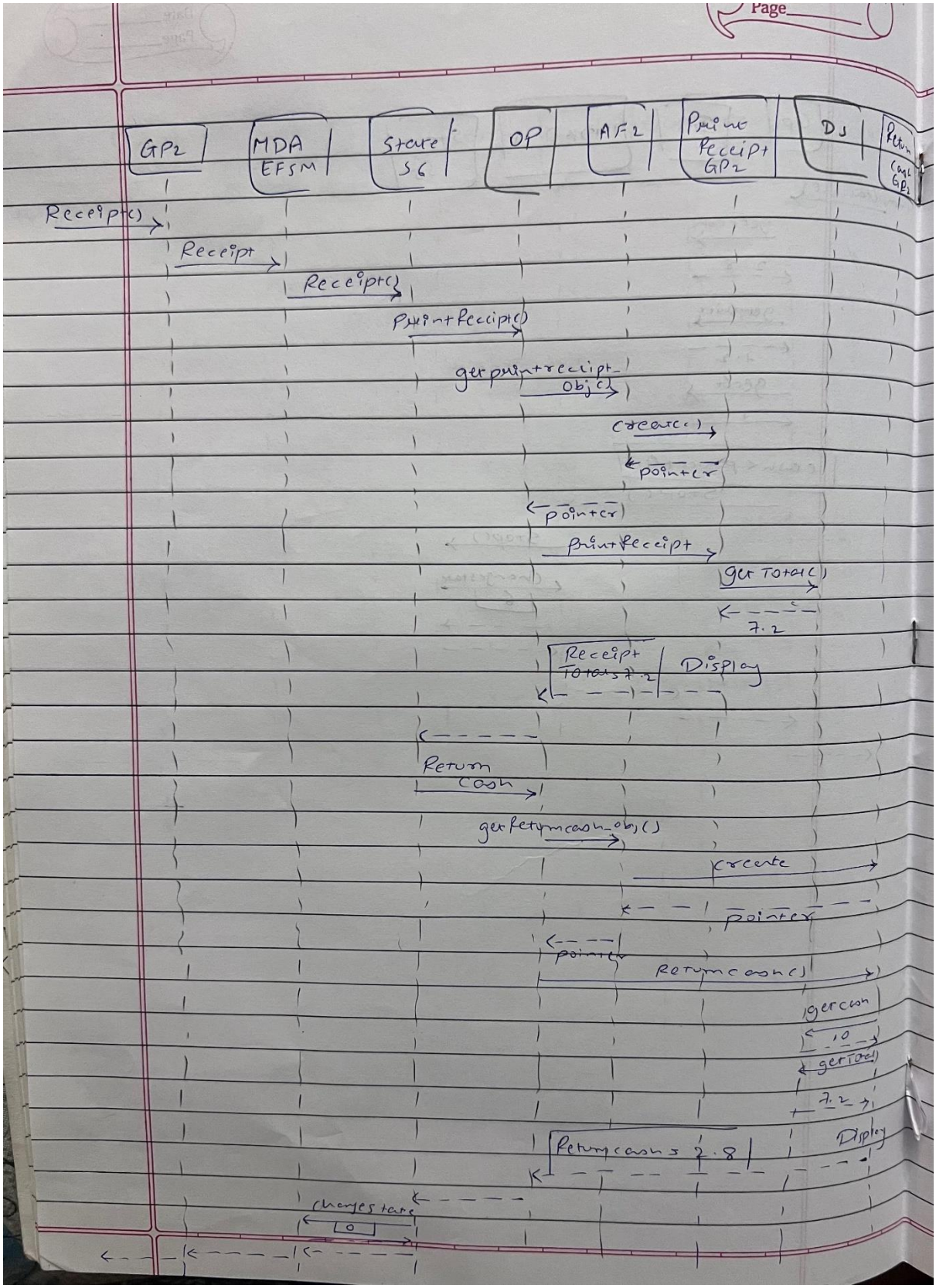

GP2
DS2
MDA
EFS14
Start
S1
OP
AFL
Store
cash
Gn
Display
Menu
Paycash(10)
Set temp cash()
temp
cash=10
Pay(cash)
Pay(cash)
Store(cash)
getstorecash obj()
create()
pointer
pointer
store cash()
get temp cash()
10
set cash()
cash=10
Display Menu()
get Display Menu obj()
create()
pointer
pointer
Display Menu()
Display message
Gas Types
change
store
3


GP2
MDA
Start
OP
AF2
Set Price GP2
premium()
getFprice
7.2
setprice
price = 7.2
SelectGas(2)
SelectGas(2)
setprice
getsetprice-obj()
create
pointer
pointer
setprice()
getFprice
7.2
setprice
Price 7.2
continue()
continue()
changestate
4


Date
Page
GP2
MDA
EFSM
State
S4
OP
AF2
Setinitial
value
GP2
DS
startpump()
startpump()
startpump()
startpump()
setinitial values()
getsetinitialvalues()
obj
create
pointer
pointer
setinitialvalues()
setG
setTotal
Total
changestate
15


GP2
MDA
EFSM
State
S5
OP
AF2
GasPumped
Msg GP2
DS1
PumpGallon()
PumpGallon()
PumpGallon()
GasPumpedMsg1()
get gaspumped
msg-obj()
create
pointer
pointer
GasPumpedMsg3()
getG
getTotal
Display Message


GPU
DS
MDATFM
Start
S5
pumpGration()
getCash
2.8
getPrice
7.2
getG
1
cash < price+G+1
StopP()
Stop()
changeState
6


GP2
MDA EFSM
State SG
OP
AF2
Print Receipt GP2
DJ
Return Cash GP2
Receipt()
Receipt
Receipt()
PrintReceipt()
getprintreceipt_obj()
create()
pointer
pointer
PrintReceipt
getTotal()
7.2
Receipt Totals 7.2
Display
Return Cash
getReturncash_obj()
create
pointer
pointer
ReturnCash()
getCash
10
getTotal()
7.2
Display
Returncash : 7.8
changes tare
0