EX.NO:2 PASSPORT MANAGEMENT SYSTEM

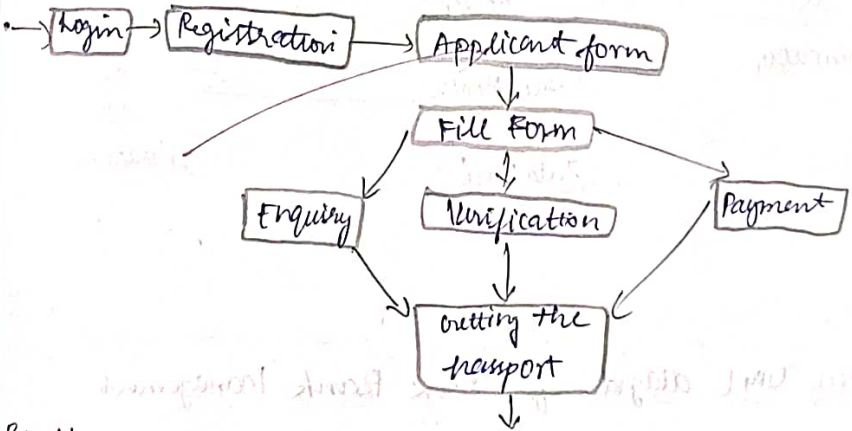
Aim:

To develop a system using UML for passport management.

Procedure Code:

```
#include <stdio.h>
#include <string.h>
typedef struct{
    char name[50];
    char address[100];
    char dob[20];
    int id;
}Applicant;
void submitApplicant(Applicant *applicant);
int verifyDetails(Applicant *applicant);
valid notifyApplicant(Applicent * applicant, int approval);
int main(){
  Applicant applicant;
  submit applicant(Applicant);
  int approval = verify Details(Applicant);
  if(approval){
    printf("Issuing passport for applicant ID:%d\n", applicant, Id);
  }
  void modify Applicant(Applicant *applicant, int approval)
  {
    if(approval)
      {
        printf("Notification: Dear %s your passport has been issued".}
    else{
      printf("Verification");
```

Diagram: Activity.



Result:

Thus, the UML diagram for passport management system is done successfully.

EX.NO:3 Book Bank Management System

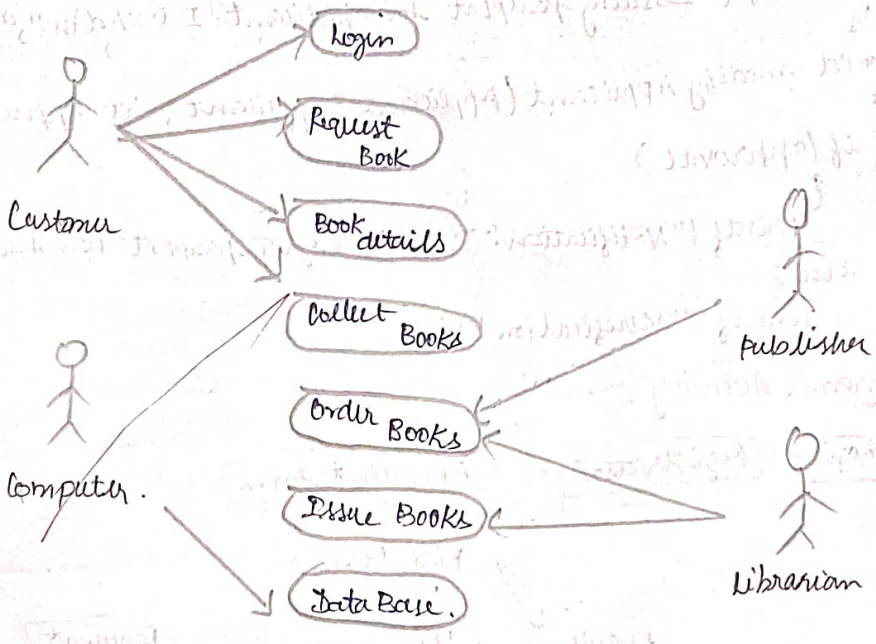
Aim:

To develop a system using UML for implementing the Book Bank process.

Code:

```
Bank_System = Bank_System()
customer1 = customer(1, "Customer", "Gmail.com")
agent1 = agent(100, "John")
bank_system.add_customer(customer)
bank_system.add_agent(agent1)
bank_system.record_interaction(interaction)
for interaction in customer.interaction{
    print("Interaction Id: Interaction.interaction_id),
    agent{Interaction.agent.name}
int main(){
  char studentDetails[] = "John Doe, Student ID = 12345;
 send details to central computer(studentDetails);
 return 0;
}
```

Diagram: Use case



Result:

Thus, the UML diagram for Book Bank Management System has been done successfully.

EX:No:4 Exam Registration Portal

Aim:

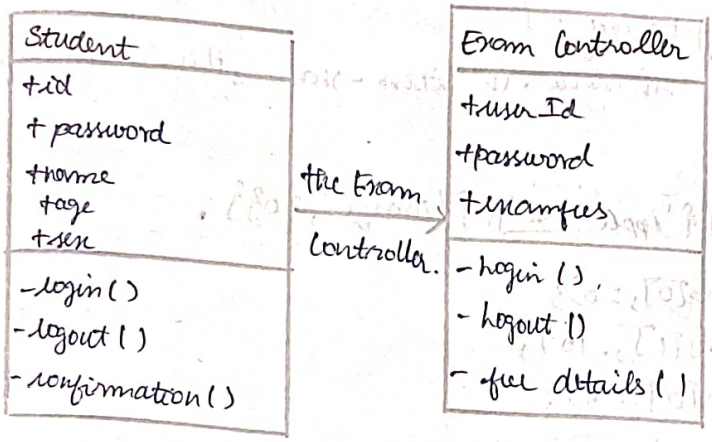
To develop a system using UML for implementing Exam Registration portal.

Code:

```
#include <stdio.h>
#include <string.h>
typedef struct {
    char name[50];
    int registered;
} Student;
int main() {
    Student student1 = {"Alice", 0};
    Student student2 = {"Bob", 0};
    register = exam(Student);
    return 0;
}
```

```
void issuetaken()
{printf("Office : Issue hallticket
                to candidate");
printf("Office: Hall ticket issued
                successfully");
}
int main()
{char candidatedetails[] = "Jane
                Doe, candidat
                ID:67890"
sendDetailsToControlComputer(
                candidate Details)
return 0;
}
```

Diagram: Class



Result:

The design for exam registration system is done successfully using the UML software.

Ex.No: 5 Stock Maintenance System

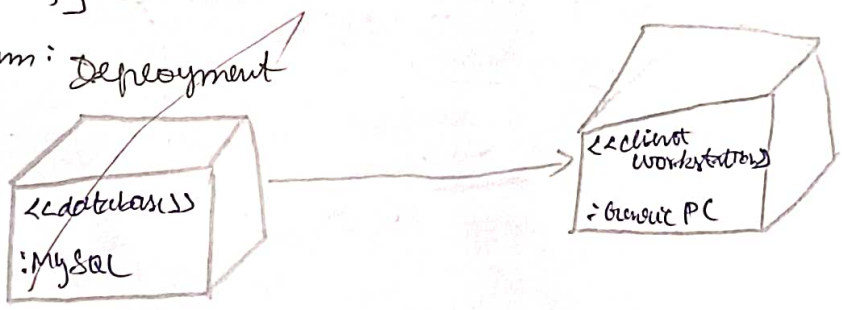
Aim:

To develop a system using UML for implementing Stock maintenance System.

Code:

```
#include <stdio.h>
#include <string.h>
typedef struct {
    char name[50];
    int registered;
} student;
void update_stock(student *s) {
    printf("%s %s\n", s->name, s->registered ? "already" : "");
    s->registered = 1;
}
int main() {
    student students[];
void update_stock(Item *item, int qty) {
    item->quantity += qty;
    printf("%s: %d in stock.\n", item->name, item->quantity);
}
int main() {
    Item items[] = {{"Apple", 50}, {"Orange", 30}};
    update_stock(&items[0], 20);
    update_stock(&items[1], -10);
    update_stock(&items[0], -5);
    return 0; }
```

Diagram: Deployment



Result:

Thus, the UML diagram for Stock Market System executed successfully.

EX.NO: 6 Online Course Reservation System

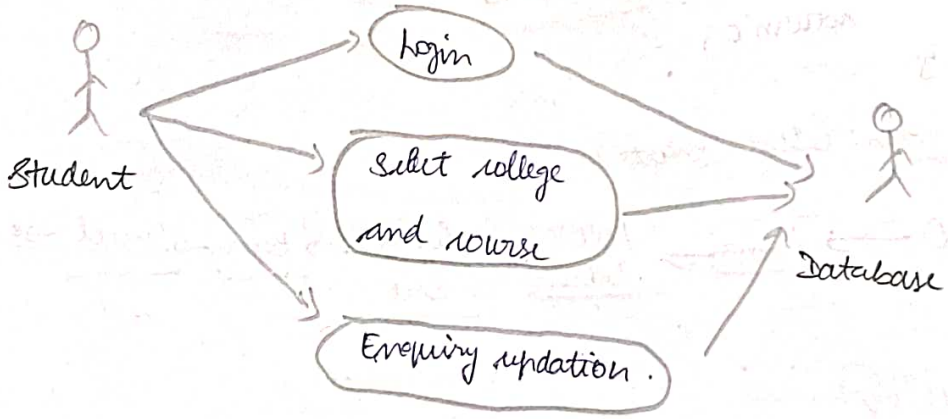
Aim:

To develop a system UML for implementing online course reservation system.

Code:

```
#include <stdio.h>
#include <string.h>
typedef struct {
  char name[50];
  int registered;
} Student;
void register_course (Student *s){
  printf("%s, %d registered.\n", s->name);
  s->registered = 1;
}
int main(){
  Student student[] = {{"Alice", 0}};
  register_course (&student[0]);
  return 0;
}
```

Diagram: Use case



Result:

Thus, the UML diagram for Online Course Reservation System successfully.

Ex No: 7 E-Ticketing System

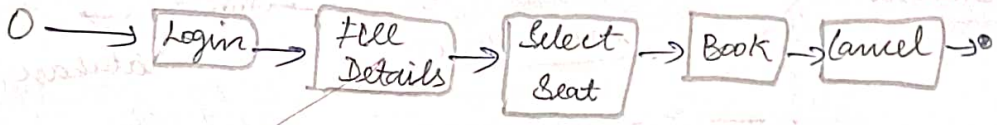
Aim:

To develop UML for implementing E-ticketing system.

Code:

```
#include <stdio.h>
typedef struct {
    int ticket_id;
    int booked; } Ticket;
void book_ticket(Ticket *t){
    if(!t->booked){
        t->booked = 1;
        printf("Ticket Booked");
    }
    else{
        printf("Already Booked");
    }
}
int main(){
    book_ticket(tickets);
    return 0;
}
```

Diagram: (State Chart)



Result:

Thus, the UML for implementing E-ticketing system has executed successfully.

Ex: No: 8 Credit Card Processing System

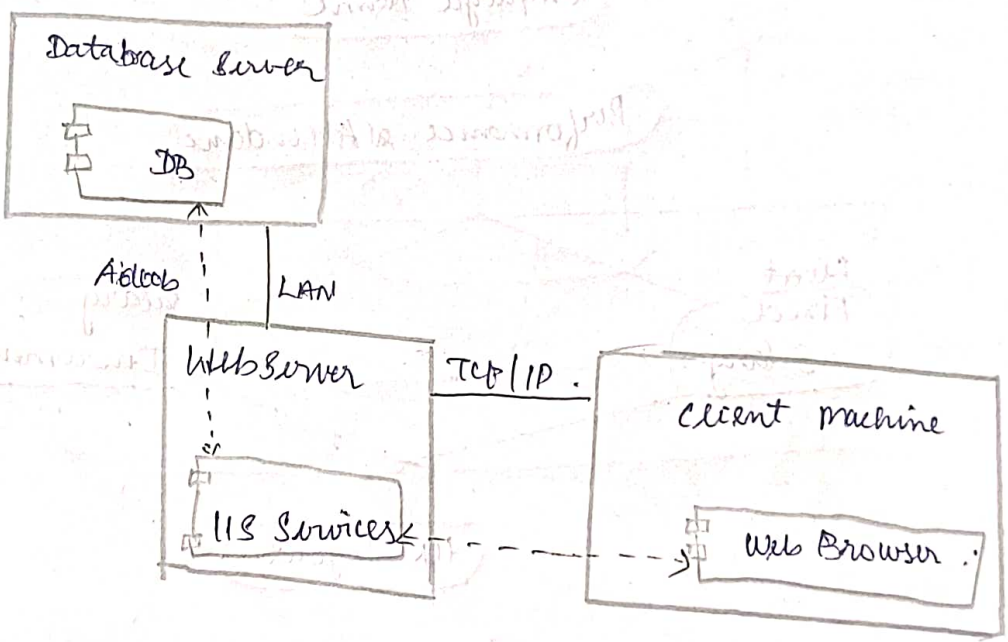
Aim:

To develop a system using UML for implementing credit card processing system.

Code:

```
#include <stdio.h>
typedef struct{
    char card_number[20];
    float balance;}
    Credit card;
void procedure.payment (Credit Card *card, float amount);
int main (){
    Credit card card = {"12.34567890", 100.0};
    process.payment (& card = 30.0);
    return 0;
}
```

Diagram: (Component)



Result:

Thus, the UML for implementing credit card processing system has executed successfully.

EX.NO:9 Software Personal Management System.

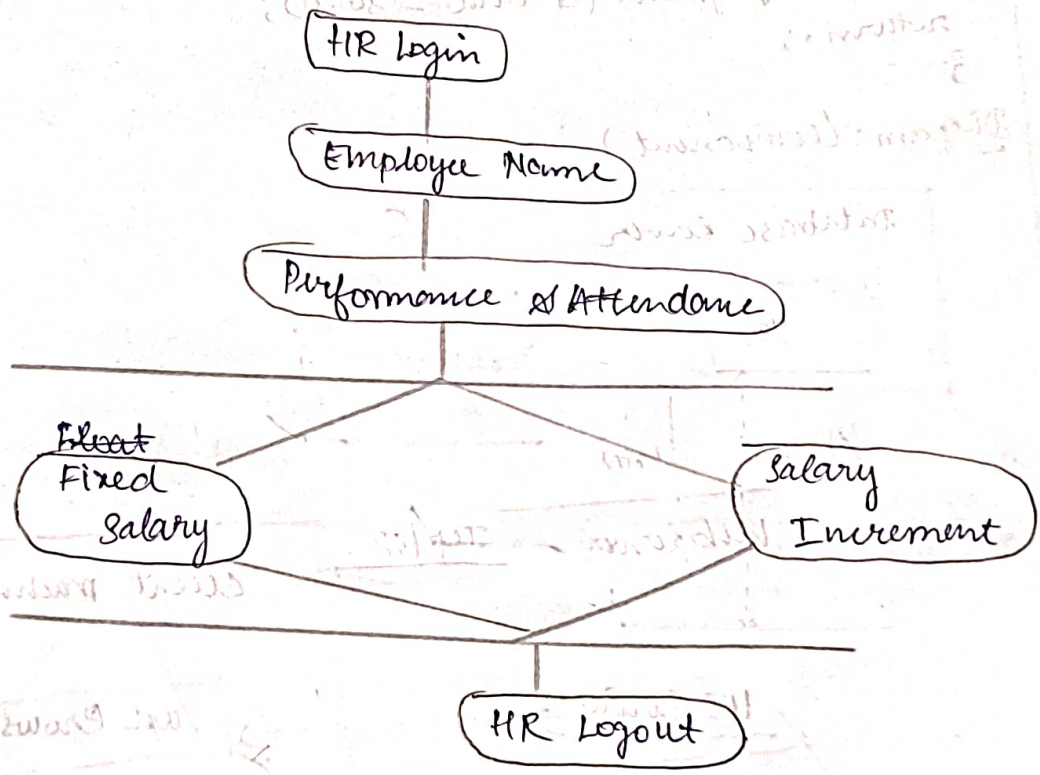
Aim:

To develop a system using UML for software personal management system.

Code:

```
#include <stdio.h>
int main(){
    personel staff[] = {{"Ram","develop"}};
    Updated_position (&staff[0]="Senior Developer");
    return 0;
}
```

Diagram: (Activity):



Result:

Thus, the UML diagram for Software Personal Management System.

Ex:No:10 E-Book Management System.

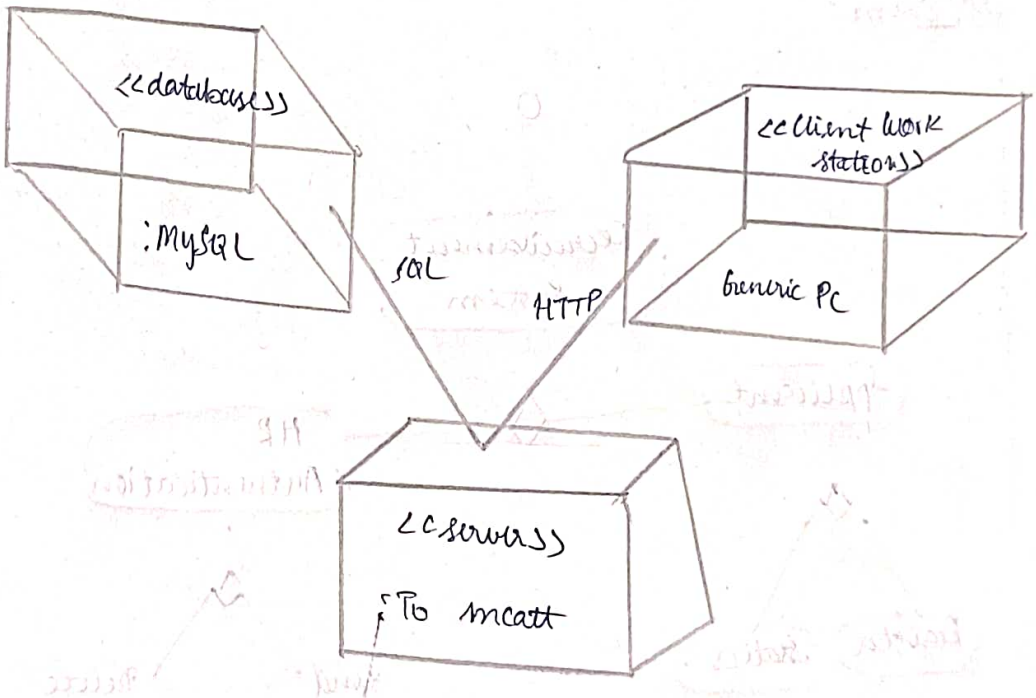
Aim:

To develop a UML for E-Book management system

Code:

```
#include <stdio.h>
typedef struct {
    char title[50];
    int available;
} EBook;
int main(){
    EBook books[] = {{"OOAD", 1}, {"AI", 1}};
    borrow_book(&books[0]);
    return 0;
}
```

Diagram:



Result:

Thus, the UML for E-Book management system has implemented successfully.

EX.NO:11 Recruitment System.

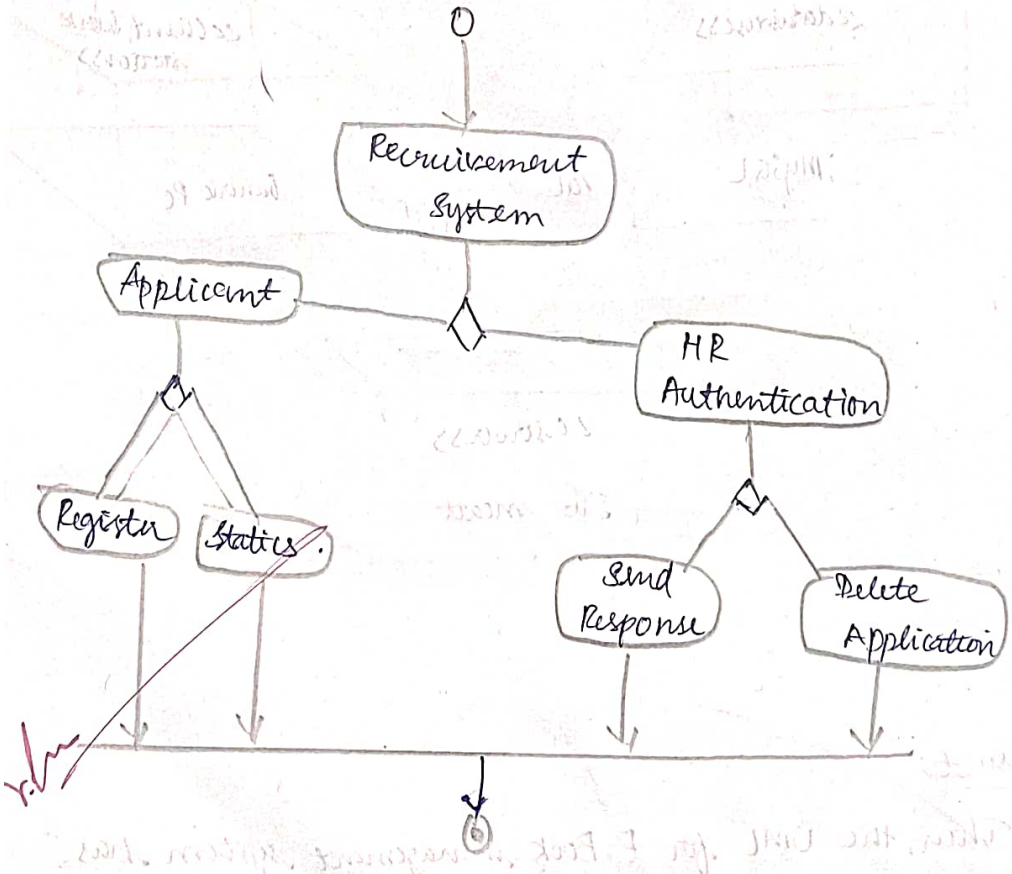
Aim:

To develop a system using UML of ~~software personal management~~ recruitment system.

Code:

```
#include <stdio.h>
typedef struct {
    char name [50];
    int hired;
} Candidate;
int main() {
    Candidate candidates [] = {{"Ram", 0}};
    process_candidates();
    return 0;
}
```

Diagram:



Result:

Thus, the program of UML executed diagram and verified.