

**The University of Texas at Dallas
Department of Computer Science**

CS 6360.003 Database Design

Online Airport System

Final Report

Version: 1.0

Instructor - Jalal Omer

05/02/2022

Jahnavi Subramaniam	jxs200098	jxs200098@utdallas.edu
Jayen Boopathy	jxb200032	jxb200032@utdallas.edu
Archana Bijoor	axb210005	axb210005@utdallas.edu
Lavanya Gopal	lxg200017	lxg200017@utdallas.edu
Suchith Natraj Javali	sxj200024	sxj200024@utdallas.edu

TABLE OF CONTENTS

1. COVER PAGE	
2. TABLE OF CONTENTS	
3. INTRODUCTION.....	4
3.1 Purpose.....	4
3.2 Overview.....	4
4. SYSTEM REQUIREMENTS.....	5
4.1 Context Diagram.....	5
4.2 Interface Requirements of the System (or Subsystem).....	6
4.3 Functional Requirements.....	14
4.3.1 Login Functional Requirements.....	14
4.3.2 Browsing Functional Requirements.....	15
4.3.3 Administrator Functional Requirements	15
4.4 Non Functional Requirements.....	16
5. CONCEPTUAL DESIGN OF THE DATABASE	17
5.1 ER Diagram.....	17
5.2 List of Business Rules and Integrity Constraints.....	18
5.2.1 Business Rules.....	18
5.2.2 Integrity Constraints.....	18
6. LOGICAL DATABASE SCHEMA.....	19
6.1 Relational Database Schema.....	19

6.2 SQL Statements and Database Construction.....	20
6.3 Expected Database Operations.....	34
6.4 Views.....	38
7. FUNCTIONAL DEPENDENCIES AND DATABASE NORMALIZATION.....	41
7.1 Identifying Functional dependencies.....	41
7.2 Normalizing the Relations in the Schema.....	41
7.3 SQL Statements for Constructing Normalized Table.....	44
8.THE DATABASE SYSTEM.....	47
9. USER APPLICATION INTERFACE	52
10. CONCLUSION AND FUTURE WORK	59
11. REFERENCES	59

3. INTRODUCTION

3.1 Purpose

Main objective of this project is to build the Online Airport system to organize information about Airplanes stationed and maintained at the airport. This document describes the requirements, functionalities provided for the users by the system. The intended audience for this document are the users of the Online Airport System like technicians, employees, traffic controller and the Administrator.

3.2 Overview

The Online Airport system provides information about every airplane's details like registration number, model, capacity. It also gives insight on technicians, traffic controllers and other employees working at the airport. Additionally, it also maintains the records regarding Airplane's test and quality.

The Online Airport System(OAS) allows users (Employees of the Airport) to create and use their account through it. It provides access to the information about the airplanes based on their designation which includes, Technicians who can login and update personal as well as the test and quality details of the aircrafts they are assigned to, Traffic controllers who can update annual medical examination reports and aircraft flight status.

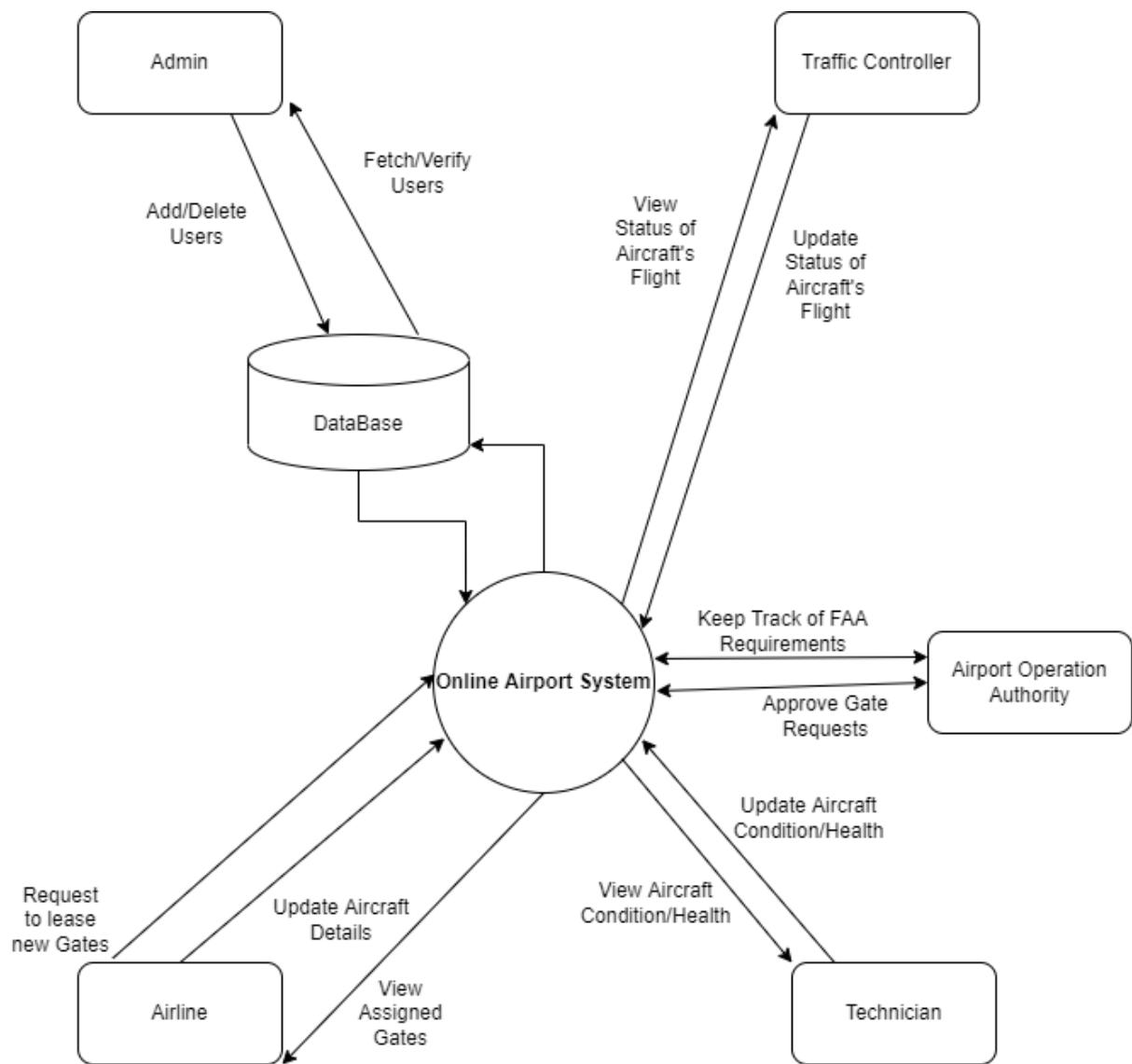
The Airport Operations Authority keeps track of the information regarding the tests conducted on airplanes to check the airplane's airworthiness as per Federal Aviation Administration(FAA) guidelines and approves gate requests from Airlines who can view, update aircraft information and request to lease gates. Finally, The OAS allows the admin to manage the employees, airline and test details.

4. SYSTEM REQUIREMENTS

4.1. Context Diagram

The Context Diagram shows the system under consideration as a single high-level process and then shows the relationship that the system has with other agents interacting with the system.

Below is the context diagram of the Online Airport System :-



4.2 Interface Requirements of the system (or subsystem)

The index page for the application is login page (a). Upon giving the LoginID and password the user can log in.

To create a new employee login, there is an option to sign up, which lands the user on page (b).

a. Login Page

The screenshot shows a login form titled "ONLINE AIRPORT SYSTEM". The form is contained within a large rectangular border. At the top center, it says "ONLINE AIRPORT SYSTEM". Below that, in the center, is the word "LOGIN". To the left of "LOGIN ID", there is a label "LOGIN ID" and to its right is a rectangular input field. Below "LOGIN ID", there is a label "PASSWORD" and to its right is another rectangular input field. At the bottom left of the form, there is a link "Don't have an account? [SIGN UP](#)".

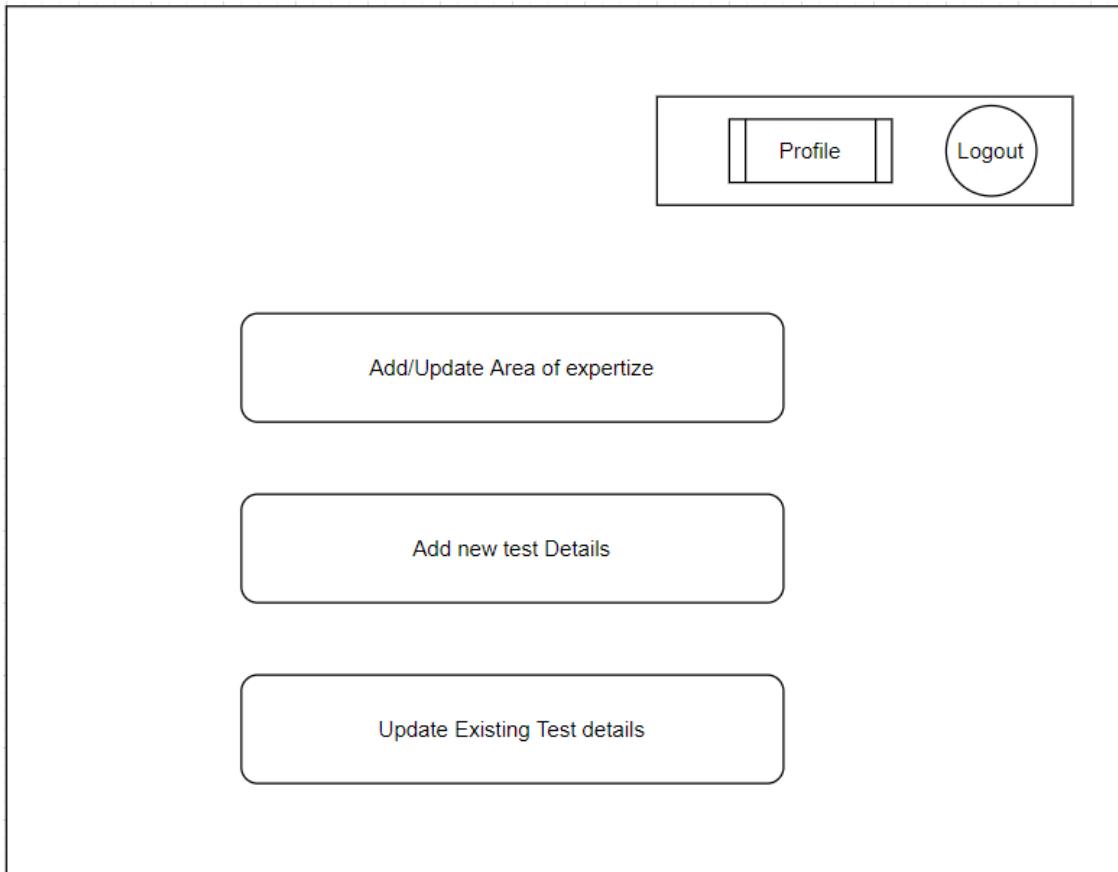
During signup, the user should enter all the mandatory fields(mentioned with an asterisk (*) symbol) as shown in figure 4.2, and upon clicking on the “submit” button the user will be able to sign up.

b. Signup Page

WELCOME TO ONLINE AIRPORT SYSTEM		
FIRST NAME *	MIDDLE NAME *	LAST NAME *
<input type="text"/>	<input type="text"/>	<input type="text"/>
EMPLOYEE ID*	DATE OF BIRTH (MM-DD-YYYY)	PASSWORD *
<input type="text"/>	<input type="text"/>	<input type="text"/>
CONFIRM PASSWORD *	ADDRESS LINE 1	ADDRESS LINE 2
<input type="text"/>	<input type="text"/>	<input type="text"/>
STATE	CITY	PINCODE
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="SUBMIT"/>		

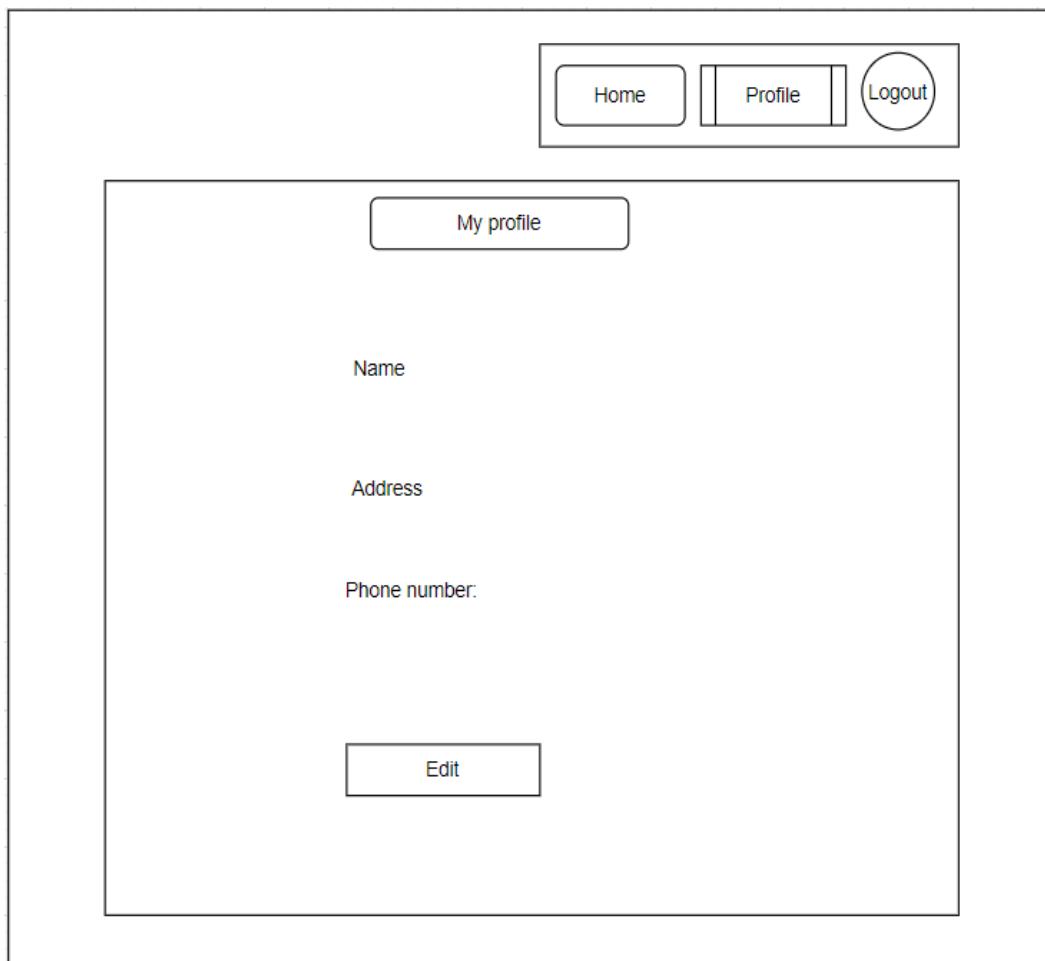
Technician main page is shown in figure (c), he can perform the operations listed by clicking the respective buttons (for ex: technician can add the test details of a new test by clicking on the “ Add new test details” button and he will be landed on the page shown on figure (e))

c. Technician Main Page



All the employees including the technician can view their profile details by clicking on the “profile” on the navigation bar. There is an option to edit the details like and phone number by clicking on the edit button as shown in figure (d).

d. Technician profile page



A wireframe diagram of a technician profile page. At the top, there is a horizontal navigation bar with three items: "Home" (in a rectangular button), "Profile" (in a rectangular button with a vertical line to its left), and "Logout" (in a circular button). Below this is a large rectangular content area. Inside the content area, at the top center, is a button labeled "My profile". Below this button are three text labels: "Name", "Address", and "Phone number:". At the bottom center of the content area is another button labeled "Edit".

e. Technician - add test details page

The screenshot shows a web-based application interface for adding test details. At the top right, there are three buttons: 'Home' (blue), 'Profile' (blue), and 'Logout' (blue). Below these, a sub-menu bar contains a single item: 'Add test details' (blue). The main content area contains five input fields, each with a label and a corresponding text input box. The labels are: 'Test ID *', 'Aircraft Reg_no *', 'Date of test *', 'Number Of Hours *', and 'Score *'. At the bottom center of the form is a blue rectangular button labeled 'Add'.

Test ID *	<input type="text"/>
Aircraft Reg_no *	<input type="text"/>
Date of test *	<input type="text"/>
Number Of Hours *	<input type="text"/>
Score *	<input type="text"/>

The traffic controller can add his annual medical exam details on the page shown in figure (f) and submit them by clicking on the “ submit “ button.

f. Traffic Controller - Annual medical exam Page

ANNUAL MEDICAL EXAMINATION

EMPLOYEE FIRST NAME

EMPLOYEE LAST NAME

EMPLOYEE ID

TEST DATE CONDUCTED
(MM-DD-YYYY)

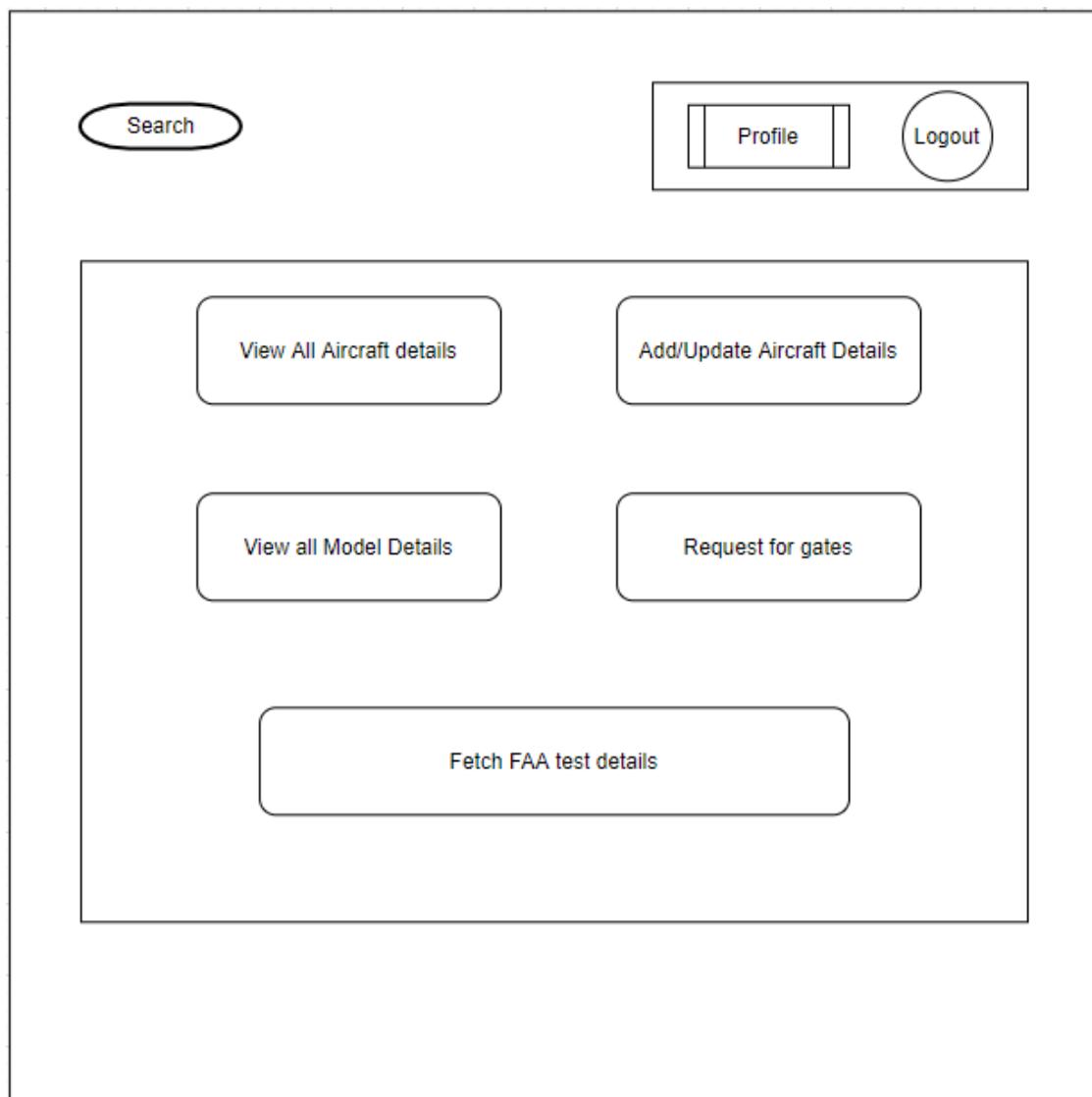
TEST SCORE
(0 - 100)

LOCATION

TEST ID

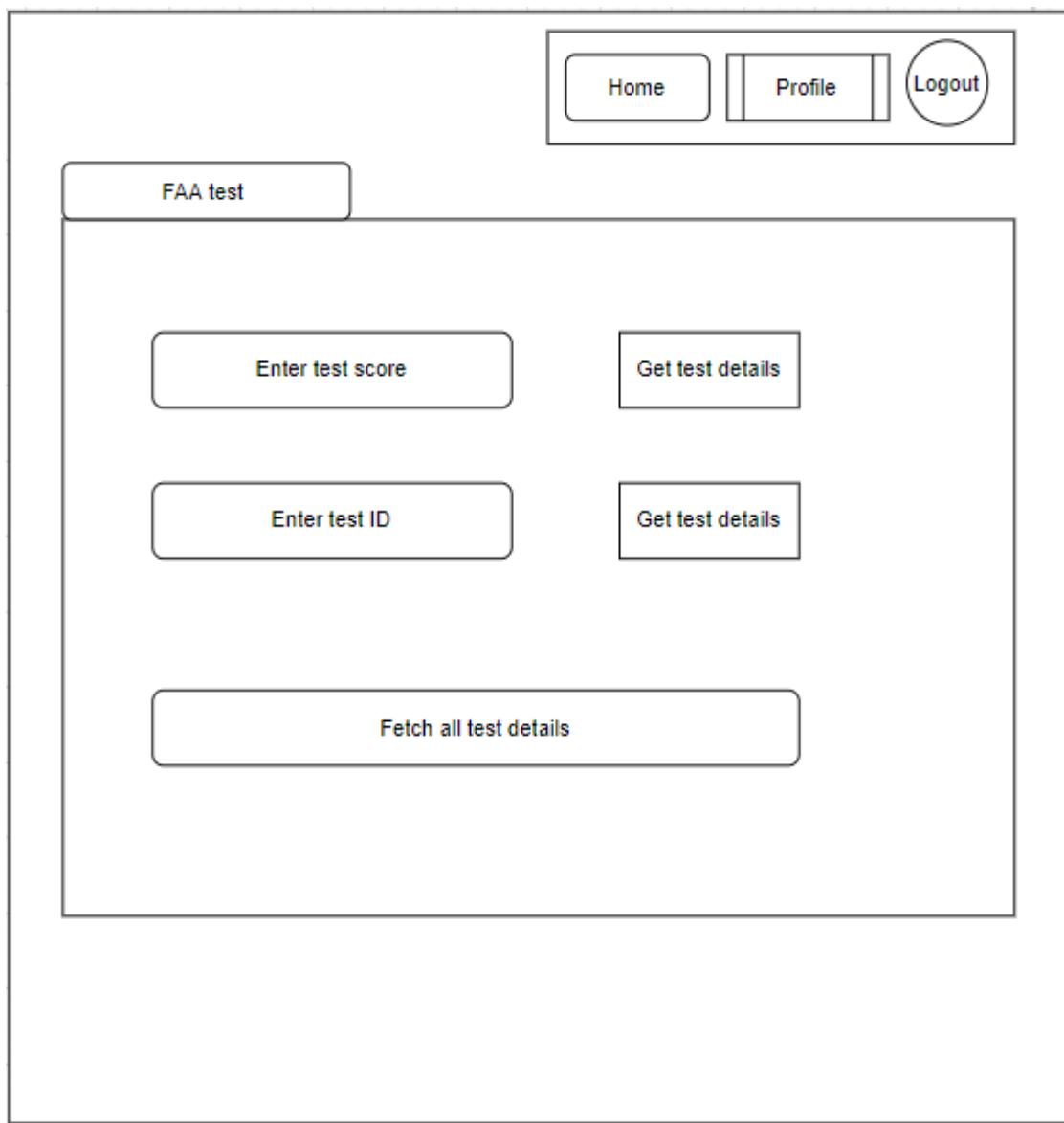
AOA's main page is as shown in figure (g), it has the options for various functions that AOA can do, by clicking on the respective button they can perform the operations.

g. Airport Operations Authority - Main page



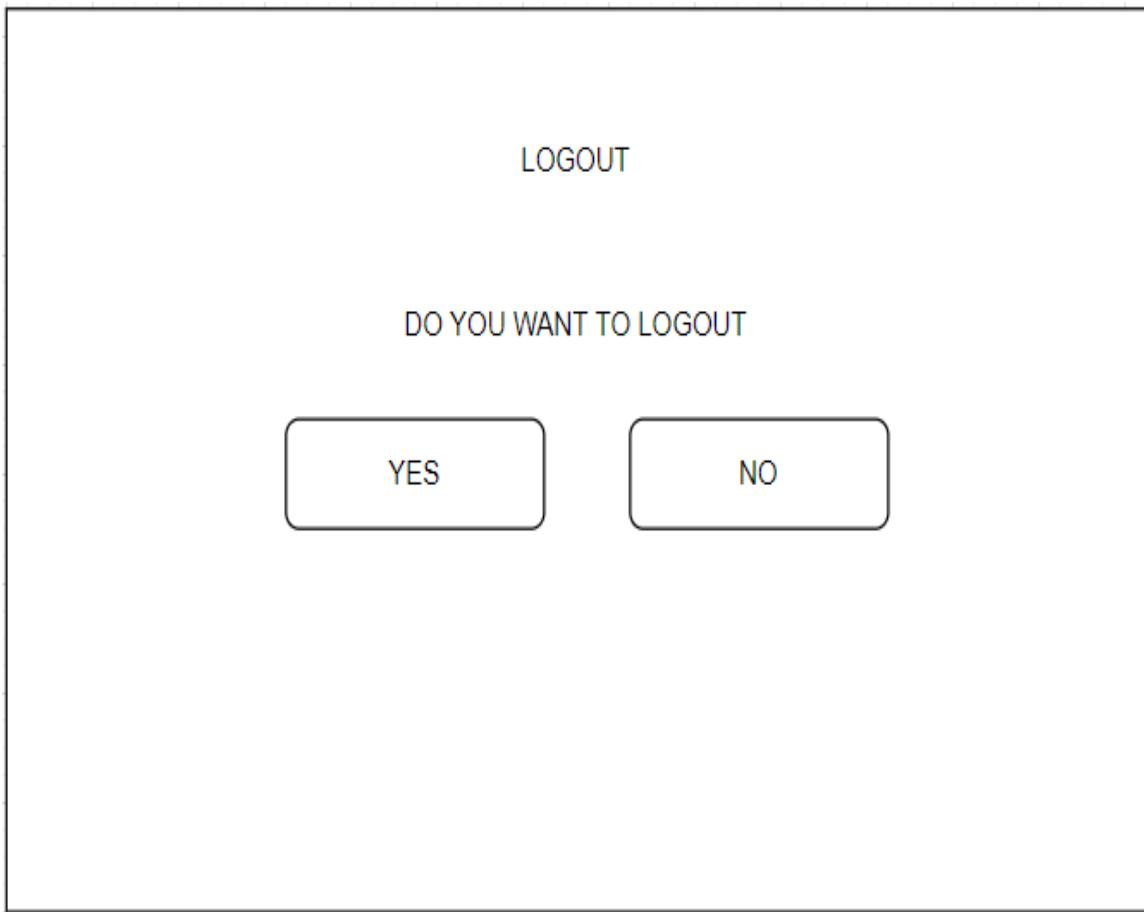
As said earlier, AOA can perform the actions related to the FAA test as shown below in (h). AOA can enter the test score and click on the “ get test details “ button to get the details of the aircraft with a score greater or equal to the entered quantity. They can fetch the details of the specific test detail by giving the test id or can fetch all the test details by clicking on the “ Fetch all test details “ button.

h. Airport Operations Authority - FAA test details page



All the users have the logout button in the navigation bar, once it's clicked the below dialog button appears as in figure (i), and if the user chooses “yes”, then he will be logged out, if the user clicks on the “no” button then he will be in his home page and no logout occurs.

i. Logout Page



4.3 Functional Requirements

4.3.1 Login Functional Requirements:

The basic features that are offered by Online Airport System as part of login functionality for OAS users belong to this category. The login functional requirements that are available for users are as listed below.

1. The system will allow the user to log in.
2. The system will verify the username and password.
3. The system will not allow the user to log in with an invalid username or password.
4. The system will be able to remember usernames and passwords.
5. The system will allow users to create accounts.
6. The system will enable users to log out of their accounts

4.3.2 Browsing Functional Requirements:

The functional requirements that are needed to make it accessible from the browser, for the users who access the Online Airport System through the browser are listed below.

1. Technician should be able to add/update his area of expertise.
2. Technicians are allowed to update the aircraft test details he worked on.
3. Traffic controller can update about his annual medical examination details.
4. Employees can access his profile details.
5. Employee can update his profile details (eg: Address, phone number)
6. The authority will have access to all the current airplane's details and model details.
7. The system gives access to the Airport Operation Authority to fetch all the tests done and its details.
8. The system allows the Airport Operation Authority to fetch details of the airplanes which have a particular maximum score in a test to verify if an aircraft is airworthy.
9. The system allows the Airport Operation Authority to fetch the details of any particular test, given he has the test_number.
10. The system allows the airlines to add/update the details of its aircrafts.
11. The system allows each airline to request for gates and gives the airport the Authority to do so.

4.3.3 Administrator Functional requirements:

The administrator is responsible for creating and maintaining the database, and also has the authority and access to the database to modify as needed. Below are the administrator functional requirements for the Online Airport System.

1. The admin can add/delete the users.
2. The admin can fetch or verify the user details
3. The admin can check/update the aircraft details.
4. The admin can add/update the model details.
5. The admin can update the salary of the users(technician, ground staff etc.)
6. The admin can fetch the details of which technician tested any particular aircraft.

7. The admin should maintain/verify the record of the annual medical examination of the Traffic controller.
8. The admin should assign every employee(technician and traffic controller) to a union.
9. The admin can assign/update the gates to which the aircraft should belong to.
10. The admin can fetch/verify the FAA tests done on aircrafts.

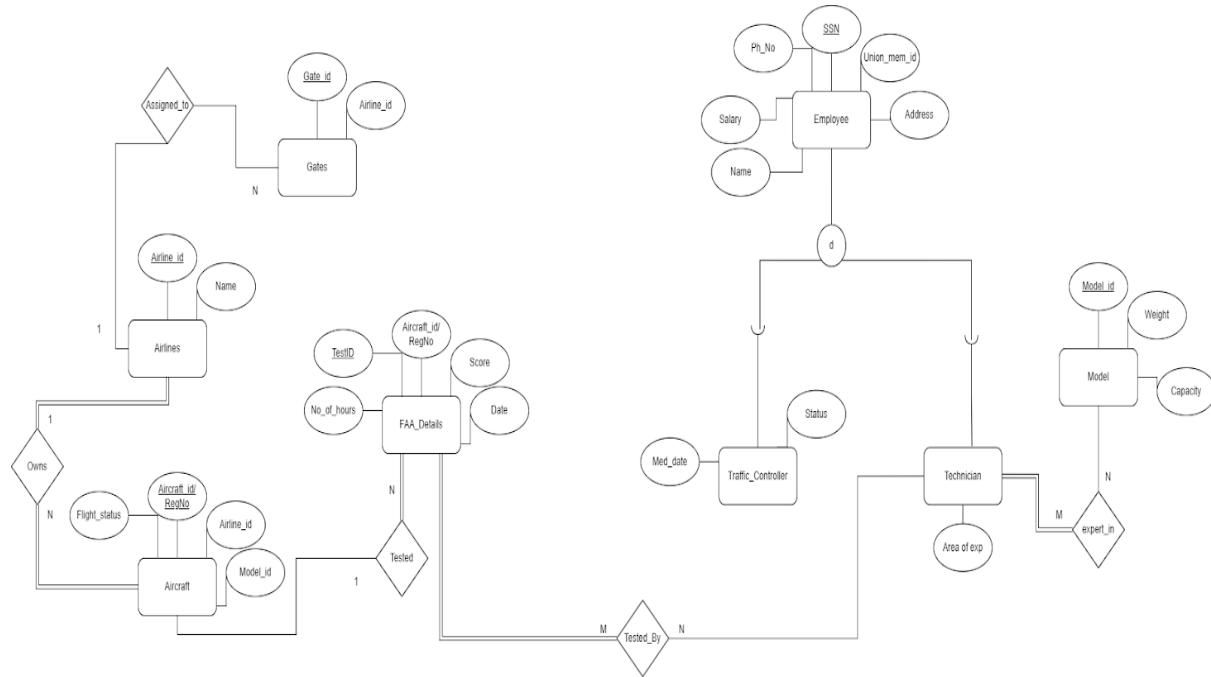
4.4 Non-Functional Requirements

The non-functional requirements define the basic standards that the system maintains to reach the market needs. Also known as the system qualities, they draw constraints and restrictions for the system attributes. The functional requirements will help to maintain performance, usability, scalability, and other aspects that ensure the effectiveness of the entire system. In any project, the non-functional requirements are as critical as the functional requirements, and not being able to attain any of these requirements might result in a system that fails to satisfy its customers.

1. The system will have a promising User Interface which will be easy to use.
2. The system will ensure to be error free.
3. The system will be scalable.
4. The system will have appreciable maintenance by segregating only the active users.
5. The system will keep its data secured and ensure access control.

5. CONCEPTUAL DESIGN OF THE DATABASE

5. 1 ER Diagram



Assumptions:

1. There may be other employees apart from technicians and traffic controllers.
2. A predefined number of gates are leased by the airlines from the airport.
3. Different models have specific weights and capacities.
4. A test can be conducted by multiple technicians depending on their expertise

5.2 List of Business Rules and Integrity Constraints

5.2.1 Business Rules:

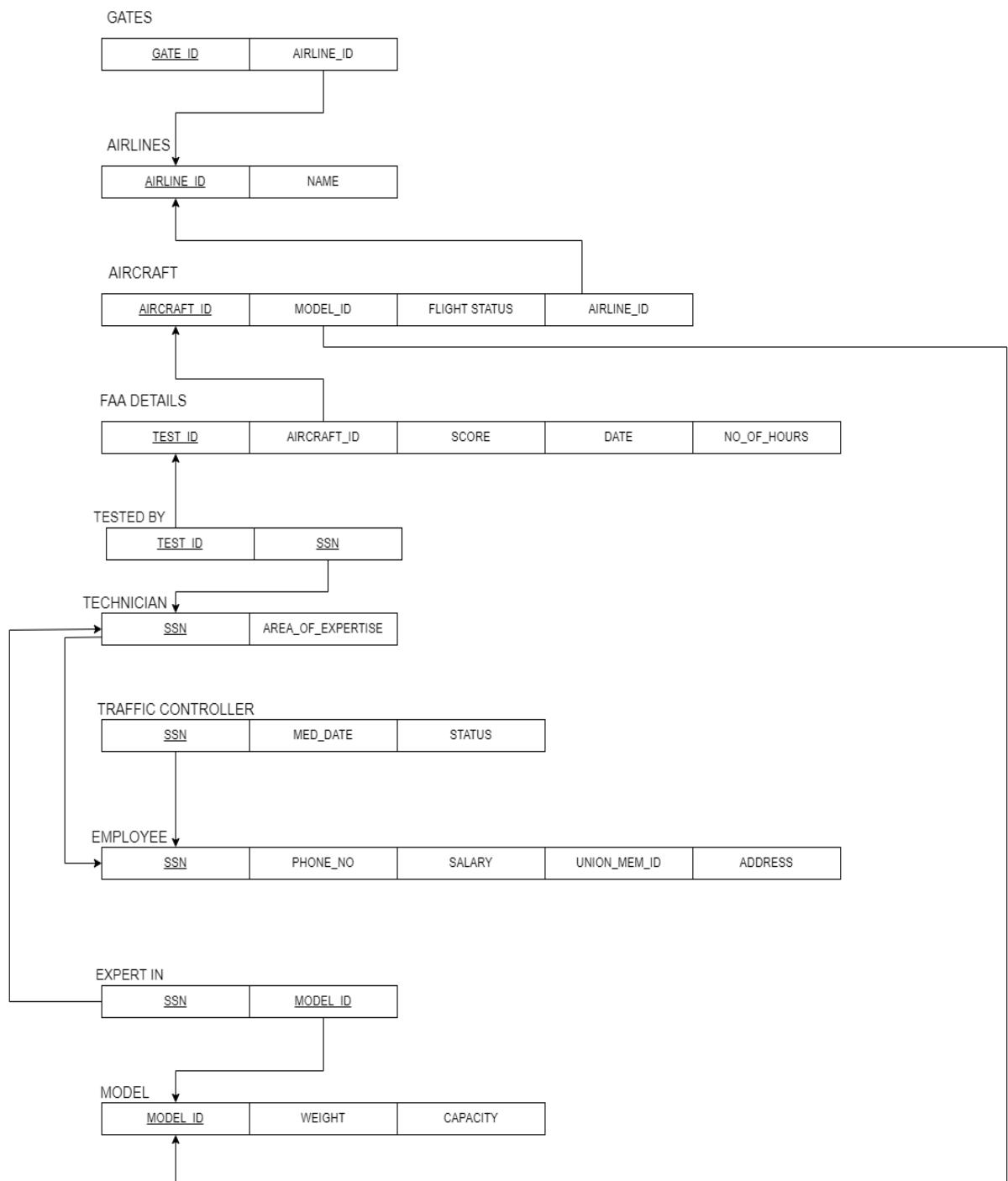
1. Every new Employee should be a part of the Union and be assigned a Membership ID.
2. The FAA Test Score of an Aircraft must fall in the range [0-100].
3. No airline can be assigned more than 10 gates by the Airport Authority.
4. Traffic controllers enter an inactive state if they haven't had a medical examination in the past 365 days.

5.2.2 Integrity Constraints:

	OnUpdate	OnDelete
Gates - Airline Id	Cascade	Set Null
Airlines- Airline Id	Cascade	Set Default
Aircraft- Model Id	Restrict	Restrict
FAA_Details- Aircraft Id	Cascade	Set Null
Tested By- Ssn	Cascade	Set Default
Tested By- TestId	Cascade	Restrict
Technician- Ssn	Cascade	Cascade
Traffic Controller-Ssn	Cascade	Cascade
Expert In- Ssn	Cascade	Set Default
Expert In- Model Id	Cascade	Cascade

6. LOGICAL DATABASE SCHEMA

6.1 Relational Database Schema



6.2. SQL Statements and Database Construction

I. Database creation

```
1 •  create database airport_system;
2 •  use airport_system;
```

II. Table creation

```
4 •  create table Airlines(airline_id int default 1111, name varchar(20), primary key (airline_id));
5
6 •  create table Gates(gate_id int, airline_id int, primary key (gate_id), foreign key(airline_id) references Airlines(airline_id) on update cascade on delete set null);
7
8 •  create table Model ( model_id int, weight int, capacity int, primary key(model_id));
9
10 •  create table employee ( ssn int(9) default 999999999, salary decimal(12), union_mem_id int, primary key(ssn));
11
12 •  create table Address( ssn int(9), address_line1 varchar(20), address_line2 varchar(20), city
13   varchar(20), state varchar(20), pincode varchar(7), primary key(ssn, address_line1), foreign key(ssn) references Employee(ssn) on update cascade on delete cascade );
14
15 •  create table Phone_Number ( ssn int(9), ph_no int(11), primary key(ssn, ph_no), foreign key ( ssn ) references employee(ssn) on update cascade on delete cascade);
16
17 •  create table Aircraft( aircraft_id int, model_id int , flight_status varchar(10), airline_id int, primary key(aircraft_id),
18   foreign key (model_id) references Model(model_id) on update restrict on delete restrict, foreign key (airline_id) references Airlines(airline_id) on update cascade on delete set null);
19
20 •  create table Traffic_controller(ssn int(9), med_date date, status varchar(20), primary key(ssn), foreign key (ssn) references Employee(ssn) on update cascade on delete cascade);
21
22 •  create table Technician ( ssn int(9), area_of_expertise varchar(20), primary key (ssn), foreign key ( ssن ) references Employee(ssn) on update cascade on delete cascade);
23
24 •  create table FAA_Details( test_id int, aircraft_id int, score decimal(3), date_date, no_of_hours int , primary key(test_id), foreign key (aircraft_id) references Aircraft(aircraft_id)
25   on update cascade on delete set null);
26
27 •  create table Tested_by ( test_id int, ssn int(9), primary key(test_id, ssn), foreign key(test_id) references FAA_Details(test_id) on update cascade on delete restrict, foreign key (ssn)
28   references Technician(ssn) on update cascade on delete cascade);
29
30 •  create table expert_in(ssn int(9), model_id int, primary key(ssn, model_id), foreign key(ssn) references Technician(ssn) on update cascade , foreign key(model_id) references Model(model_id)
31   on update cascade on delete cascade);
32
```

III. Output:

Action	Time	Action	Message	Duration / Fetch
174	00:07:26	create table Airlines(airline_id int default 1111, name varchar(20), primary key (airline_id))	0 row(s) affected	0.063 sec
175	00:07:38	create table Gates(gate_id int, airline_id int, primary key (gate_id), foreign key(airline_id) references Airlines(airline_id) on update cascade on delete set null)	0 row(s) affected	0.047 sec
176	00:07:40	create table Model (model_id int, weight int, capacity int, primary key(model_id))	0 row(s) affected	0.031 sec
177	00:07:42	create table employee (ssn int(9) default 999999999, salary decimal(12), union_mem_id int, primary key(ssn))	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.	0.031 sec
178	00:07:45	create table Address(ssn int(9), address_line1 varchar(20), address_line2 varchar(20), city varchar(20), state v...	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.	0.032 sec
179	00:07:47	create table Phone_Number (ssn int(9), ph_no int(11), primary key(ssn, ph_no), foreign key (ssn) references employee(ssn) on update cascade on delete cascade);	0 row(s) affected, 2 warning(s): 1681 Integer display width is deprecated and will be removed in a future release...	0.031 sec
180	00:07:52	create table Aircraft(aircraft_id int, model_id int , flight_status varchar(10), airline_id int, primary key(aircraft_id), ...)	0 row(s) affected	0.078 sec
181	00:07:58	create table Traffic_controller(ssn int(9), med_date date, status varchar(20), primary key(ssn), foreign key (ssn) r...	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.	0.031 sec
182	00:08:00	create table Technician (ssn int(9), area_of_expertise varchar(20), primary key (ssn), foreign key (ssن) references Employee(ssn) on update cascade on delete cascade);	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.	0.031 sec
183	00:08:01	create table FAA_Details(test_id int, aircraft_id int, score decimal(3), date_date, no_of_hours int , primary key(te...	0 row(s) affected	0.031 sec
184	00:08:04	create table Tested_by (test_id int, ssn int(9), primary key(test_id, ssn), foreign key(test_id) references FAA_Details(test_id) on update cascade on delete restrict, foreign key (ssn) references Technician(ssn) on update cascade on delete cascade);	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.	0.031 sec
185	00:08:05	create table expert_in(ssn int(9), model_id int, primary key(ssn, model_id), foreign key(ssn) references Technician(ssn) on update cascade on delete cascade);	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.	0.032 sec

IV. List of tables created:

The screenshot shows the MySQL Workbench interface. The top tab bar has 'SQL_queries_HW2' and 'Airport_system*' selected. Below the tabs is a toolbar with various icons. The main area contains the following SQL code:

```
1
2 •  create database airport_system;
3 •  use airport_system;
4 •  show tables;
```

Below the code, a horizontal line separates it from the results grid. The results grid has 'Result Grid' and 'Filter Rows:' buttons. The table 'Tables_in_airport_system' is listed, containing the following table names:

Tables_in_airport_system
address
aircraft
airlines
employee
expert_in
faa_details
gates
model
phone_number
technician
tested_by
traffic_controller

V. Table Description:

The screenshot shows the MySQL Workbench interface with the SQL query 'desc Airlines;' entered in the query editor. Below the query, a horizontal line separates it from the results grid. The results grid has 'Result Grid', 'Filter Rows:', 'Export:', and 'Wrap Cell Content:' buttons. The table 'Airlines' is described with the following columns:

Field	Type	Null	Key	Default	Extra
airline_id	int	NO	PRI	1111	
name	varchar(20)	YES		NULL	

15 • desc Gates;

16

	Field	Type	Null	Key	Default	Extra
▶	gate_id	int	NO	PRI	NULL	
	airline_id	int	YES	MUL	NULL	

31 • desc Model;

32

	Field	Type	Null	Key	Default	Extra
▶	model_id	int	NO	PRI	NULL	
	weight	int	YES		NULL	
	capacity	int	YES		NULL	

40 • desc address;

41

	Field	Type	Null	Key	Default	Extra
▶	ssn	int	NO	PRI	NULL	
	address_line1	varchar(20)	NO	PRI	NULL	
	address_line2	varchar(20)	YES		NULL	
	city	varchar(20)	YES		NULL	
	state	varchar(20)	YES		NULL	
	pincode	varchar(7)	YES		NULL	

```
43 • desc Phone_Number;
```

	Field	Type	Null	Key	Default	Extra
▶	ssn	int	NO	PRI	NULL	
	ph_no	int	NO	PRI	NULL	

```
50 • desc Aircraft;
```

```
51
```

	Field	Type	Null	Key	Default	Extra
▶	aircraft_id	int	NO	PRI	NULL	
	model_id	int	YES	MUL	NULL	
	flight_status	varchar(10)	YES		NULL	
	airline_id	int	YES	MUL	NULL	

```
55 • desc Traffic_controller;
```

```
56
```

	Field	Type	Null	Key	Default	Extra
▶	ssn	int	NO	PRI	NULL	
	med_date	date	YES		NULL	
	status	varchar(20)	YES		NULL	

```
58 • desc Technician;
```

```
59
```

	Field	Type	Null	Key	Default	Extra
▶	ssn	int	NO	PRI	NULL	
	area_of_expertise	varchar(20)	YES		NULL	

```
61 •     desc FAA_Details;
```

```
62
```

	Field	Type	Null	Key	Default	Extra
▶	test_id	int	NO	PRI	NULL	
	aircraft_id	int	YES	MUL	NULL	
	score	decimal(3,0)	YES		NULL	
	date	date	YES		NULL	
	no_of_hours	int	YES		NULL	

```
67 •     desc Tested_by;
```

```
68
```

	Field	Type	Null	Key	Default	Extra
▶	test_id	int	NO	PRI	NULL	
	ssn	int	NO	PRI	NULL	

```
70 •     desc expert_in;
```

```
71
```

	Field	Type	Null	Key	Default	Extra
▶	ssn	int	NO	PRI	NULL	
	model_id	int	NO	PRI	NULL	

VI. Data Population:

A. Airlines table:

```
insert into airlines(airline_id, name) values(222,"Airindia");
insert into Airlines values(333,"Airaisa");
insert into Airlines values(444,"AmericanAirlines");
insert into Airlines values(555,"UnitedAirlines");
insert into Airlines values(666,"SoutheasternAirlines");
insert into Airlines values(777,"BritishAirways");
insert into Airlines values(888,"Qatar");
insert into Airlines values(999,"DeltaAirlines");
```

	airline_id	name
▶	222	Airindia
	333	Airaisa
	444	AmericanAirlines
	555	UnitedAirlines
	666	SoutheasternAirlines
	777	BritishAirways
	888	Qatar
	999	DeltaAirlines
	NULL	NULL

B. Gates Table:

```
insert into Gates values(1111,222);
insert into Gates values(1112,222);
insert into Gates values(1113,333);
insert into Gates values(1114,333);
insert into Gates values(1115,333);
insert into Gates values(1116,444);
insert into Gates values(1117,444);
insert into Gates values(1118,555);
insert into Gates values(1119,666);
insert into Gates values(1120,777);
insert into Gates values(1121,888);
insert into Gates values(1122,999);
insert into Gates values(1123,999);
```

Result Grid		Filter R
	gate_id	airline_id
▶	1111	222
	1112	222
	1113	333
	1114	333
	1115	333
	1116	444
	1117	444
	1118	555
	1119	666
	1120	777
	1121	888
	1122	999
	1123	999
*	NULL	NULL

C. Model table:

```
insert into Model values(1,20,50);
insert into Model values(2,44,60);
insert into Model values(3,34,70);
insert into Model values(4,34,70);
insert into Model values(5,15,50);
insert into Model values(6,65,60);
insert into Model values(7,34,60);
insert into Model values(8,43,70);
insert into Model values(9,45,40);
insert into Model values(10,45,50);
```

Result Grid | Filter Rows:

	model_id	weight	capacity
▶	1	20	50
	2	44	60
	3	34	70
	4	34	70
	5	15	50
	6	65	60
	7	34	60
	8	43	70
	9	45	40
	10	45	50
*	NULL	NULL	NULL

D. Employee table:

```
insert into employee(ssn,salary, union_mem_id) values(987698761,55000,67671);
insert into employee(ssn,salary, union_mem_id) values(987698762,65000,67672);
insert into employee(ssn,salary, union_mem_id) values(987698763,77000,67673);
insert into employee(ssn,salary, union_mem_id) values(987698764,78000,67674);
insert into employee(ssn,salary, union_mem_id) values(987698765,89000,67675);
insert into employee(ssn,salary, union_mem_id) values(987698766,90000,67676);
insert into employee(ssn,salary, union_mem_id) values(987698767,94000,67677);
insert into employee(ssn,salary, union_mem_id) values(987698768,45000,67678);
insert into employee(ssn,salary, union_mem_id) values(987698769,56000,67679);
insert into employee(ssn,salary, union_mem_id) values(987698770,78000,67680);
insert into employee(ssn,salary, union_mem_id) values(987698771,89000,67681);
```

Result Grid | Filter Rows:

	ssn	salary	union_mem_id
▶	987698761	55000	67671
	987698762	65000	67672
	987698763	77000	67673
	987698764	78000	67674
	987698765	89000	67675
	987698766	90000	67676
	987698767	94000	67677
	987698768	45000	67678
	987698769	56000	67679
	987698770	78000	67680
	987698771	89000	67681
*	NULL	NULL	NULL

E. Address Table:

```
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698761,"Frankford Rd","Apt 3434","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698762,"Richardson Rd","Apt 898","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698763,"Frankford Rd","Apt 787","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698764,"Richardson Rd","Apt 444","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698765,"Richardson Rd","Apt 909","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698766,"Madison Rd","Apt 123","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698767,"Madison Rd","Apt 1717","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698768,"Madison Rd","Apt 5656","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698769,"Orchids Rd","Apt 109","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698770,"Melrose Rd","Apt 6767","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698771,"Melrose Rd","Apt 333","Dallas","Texas","75252");
```

The screenshot shows a MySQL Workbench result grid with the following columns: ssn, address_line1, address_line2, city, state, and pincode. The data is as follows:

	ssn	address_line1	address_line2	city	state	pincode
▶	987698761	Frankford Rd	Apt 3434	Dallas	Texas	75252
	987698762	Richardson Rd	Apt 898	Dallas	Texas	75252
	987698763	Frankford Rd	Apt 787	Dallas	Texas	75252
	987698764	Richardson Rd	Apt 444	Dallas	Texas	75252
	987698765	Richardson Rd	Apt 909	Dallas	Texas	75252
	987698766	Madison Rd	Apt 123	Dallas	Texas	75252
	987698767	Madison Rd	Apt 1717	Dallas	Texas	75252
	987698768	Madison Rd	Apt 5656	Dallas	Texas	75252
	987698769	Orchids Rd	Apt 109	Dallas	Texas	75252
	987698770	Melrose Rd	Apt 6767	Dallas	Texas	75252
	987698771	Melrose Rd	Apt 333	Dallas	Texas	75252
*	NULL	NULL	NULL	NULL	NULL	NULL

F. Phone_number table:

```
insert into phone_number (ssn, ph_no) values(987698761,567567898);
insert into phone_number (ssn, ph_no) values(987698761,567567891);
insert into phone_number (ssn, ph_no) values(987698762,675675788);
insert into phone_number (ssn, ph_no) values(987698763,564678898);
insert into phone_number (ssn, ph_no) values(987698764,987987678);
insert into phone_number (ssn, ph_no) values(987698765,234234543);
insert into phone_number (ssn, ph_no) values(987698766,678678564);
insert into phone_number (ssn, ph_no) values(987698767,890897567);
insert into phone_number (ssn, ph_no) values(987698768,890109272);
insert into phone_number (ssn, ph_no) values(987698769,679091234);
insert into phone_number (ssn, ph_no) values(987698770,786756478);
insert into phone_number (ssn, ph_no) values(987698771,897890121);
insert into phone_number (ssn, ph_no) values(987698771,897098348);
insert into phone_number (ssn, ph_no) values(987698771,345543679);
```

Result Grid | Filter Rows:

	ssn	ph_no
▶	987698761	567567891
	987698761	567567898
	987698762	675675788
	987698763	564678898
	987698764	987987678
	987698765	234234543
	987698766	678678564
	987698767	890897567
	987698768	890109272
	987698769	679091234
	987698770	786756478
	987698771	345543679
	987698771	897098348
	987698771	897890121
✳	NUL	NUL

G. Aircraft table:

```

insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(2228883,1,"inAir",222);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(2228884,2,"landed",222);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(222888,3,"undertest",222);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(3338882,1,"inAir",333);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(3338883,4,"inAir",333);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(2228881,5,"inAir",222);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(2228882,6,"landed",222);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(3338881,7,"landed",333);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(4448881,8,"landed",444);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(5558881,9,"landed",555);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(6668881,1,"inAir",666);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(7778881,2,"landed",777);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(8888881,3,"inAir",888);
insert into aircraft(aircraft_id, model_id,flight_status,airline_id) values(9998881,4,"landed",999);

```

Result Grid | Filter Rows: Ed

	aircraft_id	model_id	flight_status	airline_id
▶	222888	3	undertest	222
	2228881	5	inAir	222
	2228882	6	landed	222
	2228883	1	inAir	222
	2228884	2	landed	222
	3338881	7	landed	333
	3338882	1	inAir	333
	3338883	4	inAir	333
	4448881	8	landed	444
	5558881	9	landed	555
	6668881	1	inAir	666
	7778881	2	landed	777
	8888881	3	inAir	888
	9998881	4	landed	999
✳	NUL	NUL	NUL	NUL

H. Traffic_controller table:

```
insert into traffic_controller (ssn, med_date, status) values (987698761,11/2/2021,"Done");
insert into traffic_controller (ssn, med_date, status) values (987698762,1/2/2022,"Done");
insert into traffic_controller (ssn, med_date, status) values (987698763,3/2/2022,"Done");
```

	ssn	med_date	status
▶	987698761	0000-00-00	Done
	987698762	0000-00-00	Done
	987698763	0000-00-00	Done
✳	NULL	NULL	NULL

I. Technician table:

```
insert into Technician (ssn, area_of_expertise) values (987698764, "hydraulics");
insert into Technician (ssn, area_of_expertise) values (987698765, "engine");
insert into Technician (ssn, area_of_expertise) values (987698766, "propeller");
insert into Technician (ssn, area_of_expertise) values (987698767, "wings");
insert into Technician (ssn, area_of_expertise) values (987698768, "engine");
insert into Technician (ssn, area_of_expertise) values (987698769, "hydraulics");
```

	ssn	area_of_expertise
▶	987698764	hydraulics
	987698765	engine
	987698766	propeller
	987698767	wings
	987698768	engine
	987698769	hydraulics
✳	NULL	NULL

J. FAA_Details table:

```

insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7771,2228884,78,1/2/2021,9);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7772,2228884,78,10/1/2022,8);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7773,2228884,78,9/4/2019,8);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7774,2228883,78,7/3/2021,6);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7775,2228883,78,1/5/2020,7);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7776,2228883,78,1/7/2022,8);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7777,2228883,56,8/8/2019,9);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7778,2228883,88,8/9/2018,12);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7779,2228882,98,7/8/2022,6);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7780,2228881,78,6/7/2022,8);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7781,2228881,89,3/4/2021,9);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7782,2228882,87,1/2/2021,9);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7783,3338881,82,6/1/2022,8);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7784,4448881,34,5/3/2021,7);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7785,4448881,12,4/2/2022,12);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7786,5558881,78,3/5/2022,12);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7787,6668881,89,8/6/2022,11);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7788,7778881,89,7/7/2022,10);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7789,8888881,99,6/8/2022,6);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7790,9998881,90,5/9/2022,5);
insert into FAA_Details (test_id,aircraft_id,score,date,no_of_hours) values (7791,9998881,89,4/9/2021,4);

```

	test_id	aircraft_id	score	date	no_of_hours
▶	7771	2228884	78	0000-00-00	9
	7772	2228884	78	0000-00-00	8
	7773	2228884	78	0000-00-00	8
	7774	2228883	78	0000-00-00	6
	7775	2228883	78	0000-00-00	7
	7776	2228883	78	0000-00-00	8
	7777	2228883	56	0000-00-00	9
	7778	2228883	88	0000-00-00	12
	7779	2228882	98	0000-00-00	6
	7780	2228881	78	0000-00-00	8
	7781	2228881	89	0000-00-00	9
	7782	2228882	87	0000-00-00	9
	7783	3338881	82	0000-00-00	8
	7784	4448881	34	0000-00-00	7
	7785	4448881	12	0000-00-00	12
	7786	5558881	78	0000-00-00	12
	7787	6668881	89	0000-00-00	11
	7788	7778881	89	0000-00-00	10
	7789	8888881	99	0000-00-00	6
	7790	9998881	90	0000-00-00	5
*	7791	9998881	89	0000-00-00	4
*	NULL	NULL	NULL	NULL	NULL

K. Tested_by table:

```
insert into tested_by(test_id,ssn) values(7771,987698764);
insert into tested_by(test_id,ssn) values(7772,987698765);
insert into tested_by(test_id,ssn) values(7773,987698766);
insert into tested_by(test_id,ssn) values(7774,987698767);
insert into tested_by(test_id,ssn) values(7775,987698768);
insert into tested_by(test_id,ssn) values(7776,987698769);
insert into tested_by(test_id,ssn) values(7777,987698764);
insert into tested_by(test_id,ssn) values(7778,987698765);
insert into tested_by(test_id,ssn) values(7779,987698766);
insert into tested_by(test_id,ssn) values(7780,987698766);
insert into tested_by(test_id,ssn) values(7781,987698767);
insert into tested_by(test_id,ssn) values(7782,987698767);
insert into tested_by(test_id,ssn) values(7783,987698764);
insert into tested_by(test_id,ssn) values(7784,987698764);
insert into tested_by(test_id,ssn) values(7785,987698765);
insert into tested_by(test_id,ssn) values(7786,987698765);
insert into tested_by(test_id,ssn) values(7787,987698766);
insert into tested_by(test_id,ssn) values(7788,987698766);
insert into tested_by(test_id,ssn) values(7789,987698767);
insert into tested_by(test_id,ssn) values(7790,987698768);
insert into tested_by(test_id,ssn) values(7791,987698769);
```

Result Grid		
	test_id	ssn
▶	7771	987698764
	7777	987698764
	7783	987698764
	7784	987698764
	7772	987698765
	7778	987698765
	7785	987698765
	7786	987698765
	7773	987698766
	7779	987698766
	7780	987698766
	7787	987698766
	7788	987698766
	7774	987698767
	7781	987698767
	7782	987698767
	7789	987698767
	7775	987698768
	7790	987698768
	7776	987698769
	7791	987698769
*	NULL	NULL

L. Expert_in table:

```
insert into expert_in (ssn,model_id) values (987698764,1);
insert into expert_in (ssn,model_id) values (987698764,2);
insert into expert_in (ssn,model_id) values (987698764,3);
insert into expert_in (ssn,model_id) values (987698765,8);
insert into expert_in (ssn,model_id) values (987698766,1);
insert into expert_in (ssn,model_id) values (987698766,2);
insert into expert_in (ssn,model_id) values (987698767,7);
insert into expert_in (ssn,model_id) values (987698767,6);
insert into expert_in (ssn,model_id) values (987698768,5);
insert into expert_in (ssn,model_id) values (987698769,4);
```

Result Grid		Filter Rows:
	ssn	model_id
▶	987698764	1
	987698766	1
	987698764	2
	987698766	2
	987698764	3
	987698769	4
	987698768	5
	987698767	6
	987698767	7
	987698765	8
*	NULL	NULL

6.3 Expected Database Operations

6.3.1 Queries:

- Select all the aircraft details

```
354      -- TC view all aircrafts details
355 •   select * from aircraft;
356
357
```

	aircraft_id	model_id	flight_status	airline_id
▶	222888	3	undertest	222
	2228881	5	inAir	222
	2228882	6	landed	222
	2228883	1	inAir	222
	2228884	2	landed	222
	3338881	7	landed	333
	3338882	1	inAir	333
	3338883	4	inAir	333
	4448881	8	landed	444
	5558881	9	landed	555
	6668881	1	inAir	666
	7778881	2	landed	777
	8888881	3	inAir	888
	9998881	4	landed	999
*	NULL	NULL	NULL	NULL

- Select the details of the specific aircraft detail

```
359      -- TC view specific aircraft detail by giving aircraft id
360 •   select * from Aircraft where aircraft_id = "2228884";
```

	aircraft_id	model_id	flight_status	airline_id
▶	2228884	2	landed	222
*	NULL	NULL	NULL	NULL

c. Update the specific aircraft's flight_status.

```
367 -- TC update the aircraft status by giving specific aircraft id  
368 • update aircraft set flight_status = "undertest" where aircraft_id = "2228884";
```

The screenshot shows the MySQL Workbench interface. At the top, there is a command line with the following text:

```
761 14:49:55 update aircraft set flight_status = "undertest" where aircraft_id = "2228884"
```

Below the command line, the status bar indicates:

1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.032 sec

Under the status bar, there is a result grid titled "Result Grid". It contains the following data:

aircraft_id	model_id	flight_status	airline_id
2228884	2	undertest	222

d. SQL Query to check if the gates are assigned to the particular airline id

```
1 -- SQL Query to check if the gates are assigned to the particular airline id  
2  
3 • SELECT gate_id  
4 FROM Airlines, Gates  
5 WHERE Airlines.airline_id = Gates.airline_id;
```

The screenshot shows the MySQL Workbench interface. At the top, there is a command line with the following text:

```
100% 77:1 |  
Result Grid Filter Rows: Search Export:
```

Below the command line, the status bar indicates:

100% 77:1 |

Under the status bar, there is a result grid titled "Result Grid". It contains the following data:

gate_id
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123

e. Updating the aircraft details after the flight is landed

- **UPDATE** Aircraft
 set flight_status = 'landed'
 where aircraft_id = '3338881';

Result Grid | Filter Rows: Search | Edit: Export/Import: |

	aircraft_id	model_id	flight_status	airline_id
▶	222888	3	undertest	222
	2228881	5	inAir	222
	2228882	6	landed	222
	2228883	1	inAir	222
	2228884	2	landed	222
	3338881	7	landed	333
	3338882	1	inAir	333
	3338883	4	inAir	333
	4448881	8	landed	444
	5558881	9	landed	555
	6668881	1	inAir	666
	7778881	2	landed	777
	8888881	3	inAir	888
	9998881	4	landed	999
	HULL	HULL	HULL	HULL

6.3.2 TECHNICIAN:

a. Updating expertise

```

390 • select * from expert_in where ssn = 987698769;
391 • Update expert_in
392 Set ssn = 987698769, model_id = 6
393 Where ssn = 987698769;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Read Only
| ssn | model_id |
| 987698769 | 4 |

expert_in 14 x
Output
Action Output
# Time Action Message Duration / Fetch
35 19:03:28 Select * from FAA_Details as f where myTechid in any (select distinct ssn from tested_by where test_id=4,test_id) Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se... 0.000 sec
36 19:10:11 select * from expert_in where ssn = 987698769 LIMIT 0, 1000 1 row(s) returned 0.015 sec / 0.000 sec
Read Only

```

```

390 • select * from expert_in where ssn = 987698769;
391 • Update expert_in
392 Set ssn = 987698769, model_id = 6
393 Where ssn = 987698769;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Read Only
| ssn | model_id |
| 987698769 | 6 |

Read Only

```

```

expert_in 15 x
Output
Action Output
# Time Action Message Duration / Fetch
37 19:10:25 Update expert_in Set ssn = 987698769, model_id = 6 Where ssn = 987698769 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.015 sec
38 19:10:29 select *from expert_in where ssn = 987698769 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
Read Only

```

b. Update faa_details

```

395 • select * from faa_details where test_id = 7791;
396 • Update faa_details
397   Set aircraft_id = 9998881, score = 90 , date =2/2/2022, no_of_hours =7
398   Where test_id = 7791;
399

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid | Read Only

test_id	aircraft_id	score	date	no_of_hours
7791	9998881	89	0000-00-00	4

faa_details 16 x

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
38	19:10:29	select * from expert_in where ssn = 987698769 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
39	19:14:45	select * from faa_details where test_id = 7791 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

```

395 • select * from faa_details where test_id = 7791;
396 • Update faa_details
397   Set aircraft_id = 9998881, score = 90 , date =2/2/2022, no_of_hours =7
398   Where test_id = 7791;
399

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid | Read Only

test_id	aircraft_id	score	date	no_of_hours
7791	9998881	90	0000-00-00	7

faa_details 17 x

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
40	19:15:36	Update faa_details Set aircraft_id = 9998881, score = 90 , date =>2/2/2022, no_of_hours =>7 Where test_id = 7... Where test_id = 7791 LIMIT 0, 1000	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
41	19:15:39	select * from faa_details where test_id = 7791 LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

c. Fetch the faa_test details of the test done by specific technician

```

404 • Select *
405   from FAA_Details as f
406   where f.test_id in(select distinct test_id from tested_by where ssn = 987698769 );

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid | Form Editor | Read Only

test_id	aircraft_id	score	date	no_of_hours
776	2228883	78	0000-00-00	8
7791	9998881	90	0000-00-00	7

FAA_Details 37 x

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
64	19:23:49	select * from tested_by where ssn = 987698769 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
65	19:23:52	Select * from FAA_Details as f where f.test_id in(select distinct test_id from tested_by where ssn = 987698769) ...	2 row(s) returned	0.000 sec / 0.000 sec

6.4 Views

a. View to get all technician profile details

The screenshot shows the MySQL Workbench interface with the 'SQL_queries_Hv2' database selected. In the Navigator pane, under the 'airport_system' schema, a 'Views' folder contains the 'technician_profile_view'. The DDL pane displays the SQL code for creating the view:

```

CREATE VIEW technician_profile_view AS
select technician.ssn, address.line1, address.line2, city, ph_no
from address, phone_number, technician
where address.ssn = phone_number.ssn and address.ssn = technician.ssn ;

```

The screenshot shows the 'Review SQL Script' dialog in MySQL Workbench. It displays the same SQL code as above. The 'Apply' button is visible at the bottom right.

The screenshot shows the MySQL Workbench interface after applying the changes. The 'Output' pane at the bottom shows a message: '7 15:08:59 Apply changes to technician_profile_view' and 'Changes applied'. The DDL pane now shows the modified view code, including the 'DEFINER' clause and a more detailed 'SELECT' statement:

```

CREATE ALGORITHM = UNDEFINED
DEFINER = 'root'@'localhost'
SQL SECURITY DEFINER
VIEW `airport_system`.`technician_profile_view` AS
SELECT
    `airport_system`.`technician`.`ssn` AS `ssn`,
    `airport_system`.`address`.`address_line1` AS `address_line1`,
    `airport_system`.`address`.`address_line2` AS `address_line2`,
    `airport_system`.`address`.`city` AS `city`,
    `airport_system`.`phone_number`.`ph_no` AS `ph_no`
FROM
    ((`airport_system`.`address`
    JOIN `airport_system`.`phone_number`)
    JOIN `airport_system`.`technician`)
WHERE
    ((`airport_system`.`address`.`ssn` = `airport_system`.`phone_number`.`ssn`)
    AND (`airport_system`.`address`.`ssn` = `airport_system`.`technician`.`ssn`))

```

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator SQL_queries_Hw2 Airport_system technician_profile_view - View
SCHEMAS
Q Filter objects
▼ airport_system
  □ Tables
    address
    aircraft
    airlines
    employee
    expert_in
    faa_details
    gender
    model
    phone_number
    technician
    vendor_by
    traffic_controller
  □ Views
    technician_profile_v
      ran
      address_line1
      address_line2
      city
      ph_no
  □ Stored Procedures
    getAllAirlines
Administration Schemas Information
Table: technician
Columns:
  ssn          int PK
  area_of_expertise  varchar(255)
Object Info Session
profile_view 1 x
.profile_view 1 x
Output
Action Output
# Time Action
1 10:15:10:50 Apply changes to technician_profile_view
2 11:15:10:50 Apply changes to technician_profile_view
3 12 15:11:26 select * from technician_profile_view LIMIT 0, 1000
Message
No changes detected
No changes detected
6 row(s) returned
Duration / Fetch
0.031 sec / 0.000 sec
  
```

b. View to get traffic controller status detail

```

SQL_queries_Hw2 Airport_system fetch_tc_status - View fetch_aircraft_details fetch_faa_details
Name: fetch_tc_status
The name of the view is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:
1 • create view fetch_TC_status
2   as select ssn, status
3     from traffic_controller;
View
Apply Revert
  
```

c. View to get the aircraft details

```

SQL_queries_Hw2 Airport_system fetch_tc_status - View fetch_aircraft_details fetch_faa_details
1 • create view fetch_aircraft_details
2   as select aircraft_id, flight_status, airline_name
3     from aircraft, airlines
4       where aircraft.airline_id = airlines.airline_id;
  
```

d. View to get the FAA_test details

```

SQL_queries_Hw2 Airport_system fetch_tc_status - View fetch_aircraft_details fetch_faa_details
1 • create view fetch_FAA_details
2   as select aircraft_id, score
3     from faa_details
  
```

e. View to get the list of landed flights

```
10 • CREATE VIEW Landed_aircraft AS
11   SELECT *
12   FROM Aircraft
13   WHERE flight_status = 'landed';
14
15 •   SELECT *
16   FROM Landed_aircraft;
17
```

The screenshot shows a MySQL Workbench interface. At the top, there is a code editor window with the following SQL code:

```
10 • CREATE VIEW Landed_aircraft AS
11   SELECT *
12   FROM Aircraft
13   WHERE flight_status = 'landed';
14
15 •   SELECT *
16   FROM Landed_aircraft;
17
```

Below the code editor is a results grid titled "Result Grid". The grid has four columns: "aircraft_id", "model_id", "flight_status", and "airline_id". The data is as follows:

aircraft_id	model_id	flight_status	airline_id
2228882	6	landed	222
2228884	2	landed	222
3338881	7	landed	333
4448881	8	landed	444
5558881	9	landed	555
7778881	2	landed	777
9998881	4	landed	999

7. FUNCTIONAL DEPENDENCIES AND DATABASE NORMALIZATION

7.1 Identifying Functional dependencies

There are no Functional dependencies observed in the relations of the schema since all of the non-primary key columns are completely functionally dependent upon the primary key itself.

7.2 Normalizing the Relations in the Schema

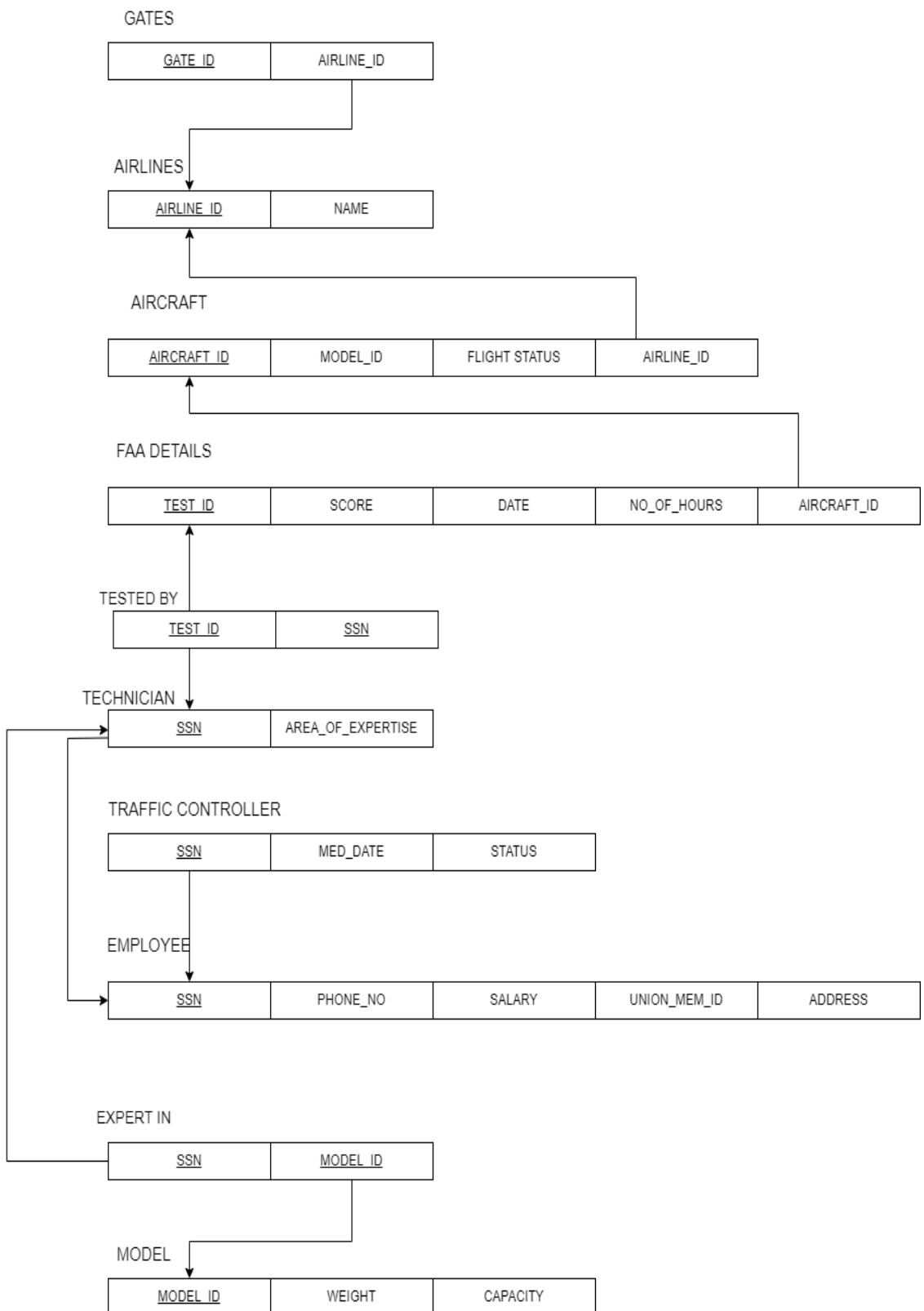
It is observed that every relation in the schema is in the Third Normal Form(3NF) except the EMPLOYEE table which consists of two attributes violating 1NF:

1. PHONE_NO → Multivalued Attribute
2. ADDRESS → Composite Attribute

Therefore the EMPLOYEE table is Decomposed into EMPLOYEE, PHONE_NUMBER and ADDRESS Tables.

And upon further Inspection every relation in the schema is observed to be in their Third Normal Form(3NF).

7.2.1 Schema before Normalization



7.2.2 Schema of all the relations in their Third Normal Form(3NF)

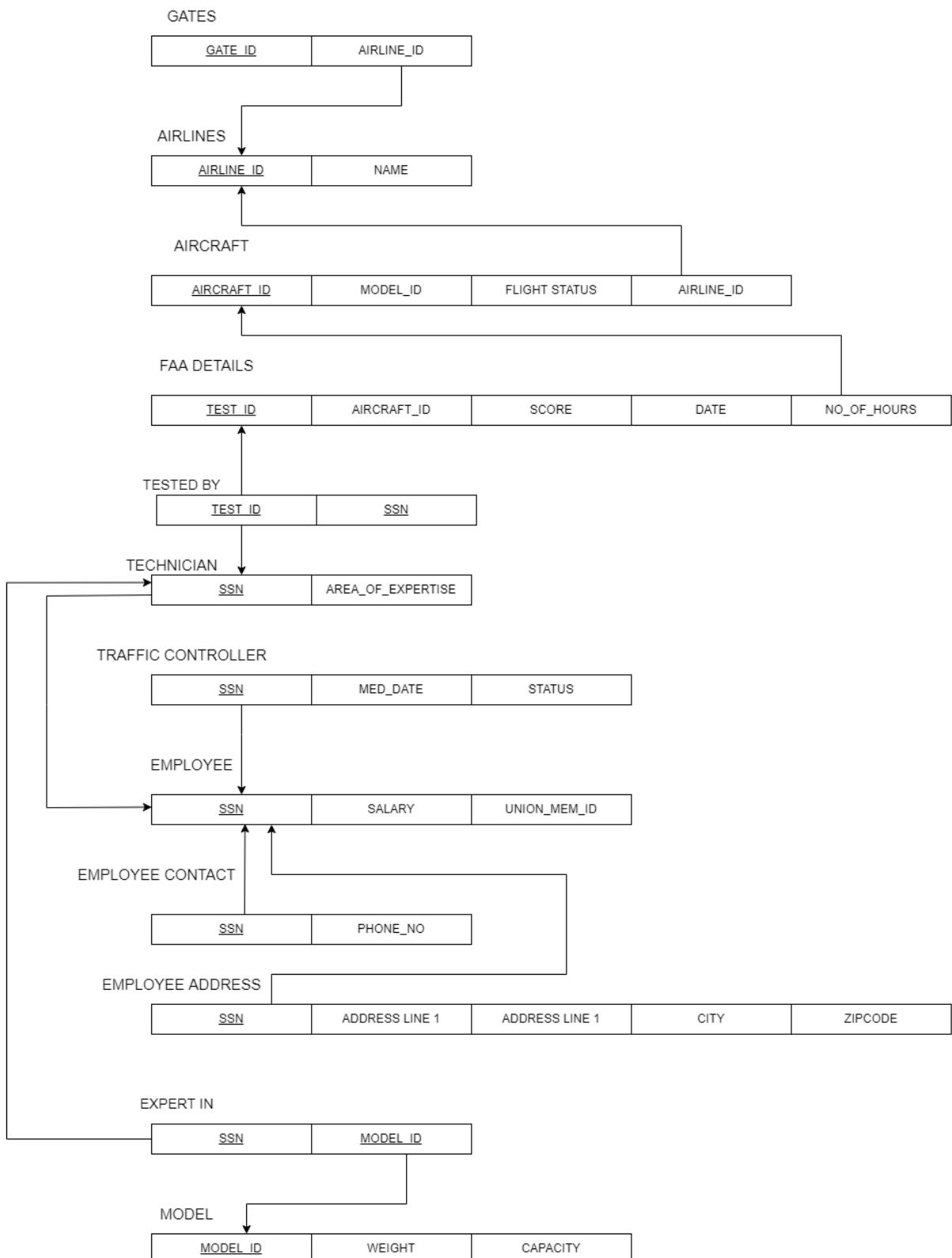


Fig : Functional Dependencies of the Normalized Relations of the DB System

7.3 SQL Statements for Constructing Normalized Table

7.3.1 Creating the Normalized Table ADDRESS and PHONE_NUMBER

- `create table Address(ssn int(9), address_line1 varchar(20), address_line2 varchar(20), city varchar(20), state varchar(20), pincode varchar(7), primary key(ssn, address_line1), foreign key(ssn) references Employee(ssn) on update cascade on delete cascade);`
 - `desc Address;`

Field	Type	Null	Key	Default	Extra
► ssn	int	NO	PRI	NULL	
address_line1	varchar(20)	NO	PRI	NULL	
address_line2	varchar(20)	YES		NULL	
city	varchar(20)	YES		NULL	
state	varchar(20)	YES		NULL	
pincode	varchar(7)	YES		NULL	

```
create table Phone_Number ( ssn int(9), ph_no int(11), primary key(ssn, ph_no), foreign key ( ssn) references employee(ssn) on update cascade on delete cascade);
desc Phone Number;
```

Field	Type	Null	Key	Default	Extra
▶ ssn	int	NO	PRI	NULL	
ph_no	int	NO	PRI	NULL	

7.3.2 Inserting Values to the Normalised Table ADDRESS and PHONE NUMBER

```
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698761,"Frankford Rd","Apt 3434","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698762,"Richardson Rd","Apt 898","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698763,"Frankford Rd","Apt 787","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698764,"Richardson Rd","Apt 444","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698765,"Richardson Rd","Apt 909","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698766,"Madison Rd","Apt 123","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698767,"Madison Rd","Apt 1717","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698768,"Madison Rd","Apt 5656","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698769,"Orchids Rd","Apt 109","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698770,"Melrose Rd","Apt 6767","Dallas","Texas","75252");
insert into Address(ssn, address_line1,address_line2,city,state,pincode) values (987698771,"Melrose Rd","Apt 333","Dallas","Texas","75252");
```

```
select * from Address;
```

ssn	address_line1	address_line2	city	state	pincode	
▶ 987698762	Richarson Rd	Apt 898	Dallas	Texas	75252	
987698763	Frankford Rd	Apt 787	Dallas	Texas	75252	
987698764	Richarson Rd	Apt 444	Dallas	Texas	75252	
987698765	Richarson Rd	Apt 909	Dallas	Texas	75252	
987698766	Madison Rd	Apt 123	Dallas	Texas	75252	
987698767	Madison Rd	Apt 1717	Dallas	Texas	75252	
987698768	Madison Rd	Apt 5656	Dallas	Texas	75252	
987698769	Orchids Rd	Apt 109	Dallas	Texas	75252	
987698770	Melrose Rd	Apt 6767	Dallas	Texas	75252	
987698771	Melrose Rd	Apt 333	Dallas	Texas	75252	
NULL	NULL	NULL	NULL	NULL	NULL	

```
insert into phone_number (ssn, ph_no) values(987698761,567567898);
insert into phone_number (ssn, ph_no) values(987698761,567567891);
insert into phone_number (ssn, ph_no) values(987698762,675675788);
insert into phone_number (ssn, ph_no) values(987698763,564678898);
insert into phone_number (ssn, ph_no) values(987698764,987987678);
insert into phone_number (ssn, ph_no) values(987698765,234234543);
insert into phone_number (ssn, ph_no) values(987698766,678678564);
insert into phone_number (ssn, ph_no) values(987698767,890897567);
insert into phone_number (ssn, ph_no) values(987698768,890109272);
insert into phone_number (ssn, ph_no) values(987698769,679091234);
insert into phone_number (ssn, ph_no) values(987698770,786756478);
insert into phone_number (ssn, ph_no) values(987698771,897890121);
insert into phone_number (ssn, ph_no) values(987698771,897098348);
insert into phone_number (ssn, ph_no) values(987698771,345543679);
select * from Phone_Number;
```

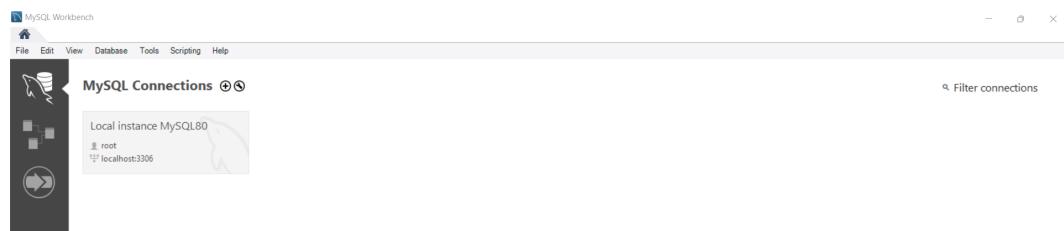
ssn	ph_no
► 987698762	675675788
987698763	564678898
987698764	987987678
987698765	234234543
987698766	678678564
987698767	890897567
987698768	890109272
987698769	679091234
987698770	786756478
987698771	345543679
987698771	897098348
987698771	897890121
NULL	NULL

8. THE DATABASE SYSTEM

We have used Mysql workbench for mysql database and Eclipse IDE for running the Java Springboot application. Below sections shows the detailed description about how to install and access our application.

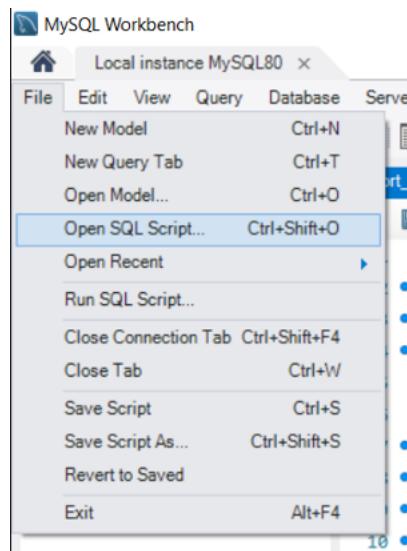
Steps to setup the mysql workbench:

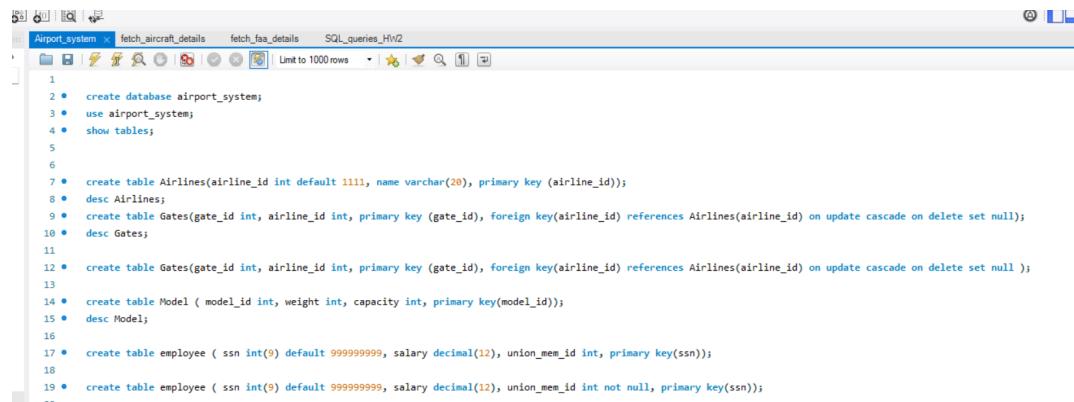
1. Download the latest version of the mysql workbench from the mysl official website and double click the downloaded file and install.
2. Give the credentials for your mysql workbench during installation and save it for later access to the workbench databases.
3. Upon installation, open the mysql application installed.



Open script file and run the command to create the database and tables

Use the commands in the .sql file provided in the project folder for all the commands to create the table and populate the data. The sample output is as shown in the section 6 (6.1, 6.2 and 6.3)

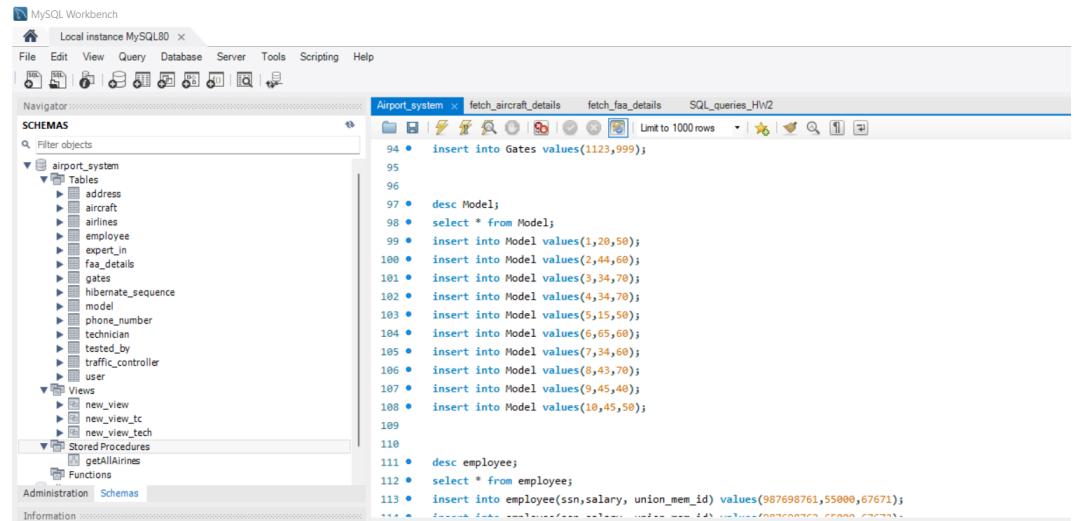




```

1
2 •  create database airport_system;
3 •  use airport_system;
4 •  show tables;
5
6
7 •  create table Airlines(airline_id int default 1111, name varchar(20), primary key (airline_id);
8 •  desc Airlines;
9 •  create table Gates(gate_id int, airline_id int, primary key (gate_id), foreign key(airline_id) references Airlines(airline_id) on update cascade on delete set null);
10 •  desc Gates;
11
12 •  create table Gates(gate_id int, airline_id int, primary key (gate_id), foreign key(airline_id) references Airlines(airline_id) on update cascade on delete set null );
13
14 •  create table Model ( model_id int, weight int, capacity int, primary key(model_id));
15 •  desc Model;
16
17 •  create table employee ( ssn int(9) default 999999999, salary decimal(12), union_mem_id int, primary key(ssn));
18
19 •  create table employee ( ssn int(9) default 999999999, salary decimal(12), union_mem_id int not null, primary key(ssn));
--
```

On the left side of the mysql workbench we can see the database created and the list of tables and views created as shown in the screenshot below. Now confirm that the mysql server status is running and start the Java application after updating the configurations as per the next steps below.

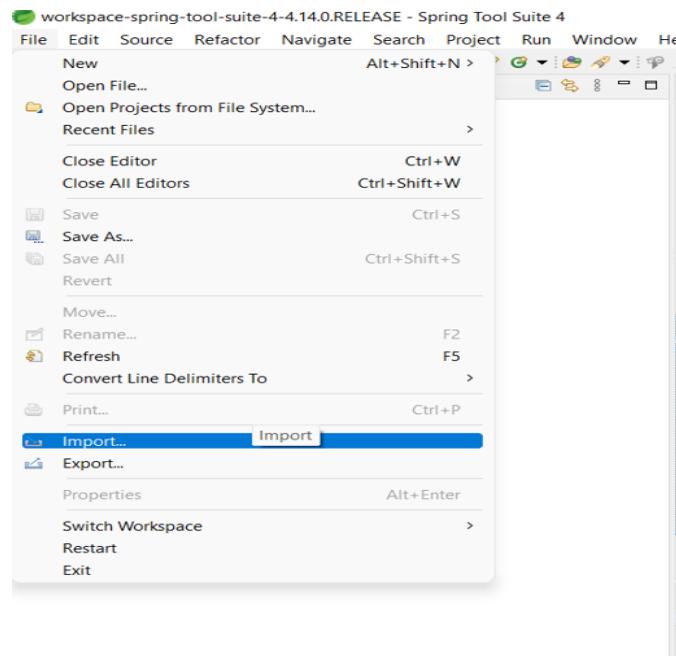


```

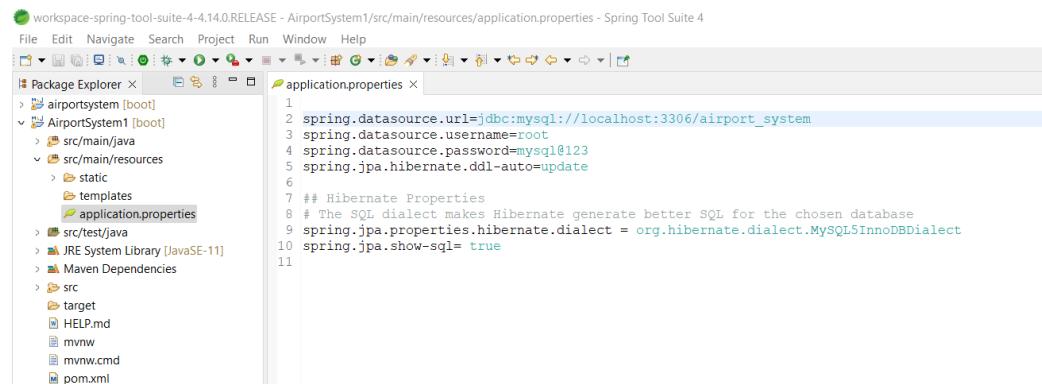
94 •  insert into Gates values(1123,999);
95
96
97 •  desc Model;
98 •  select * from Model;
99 •  insert into Model values(1,20,50);
100 • insert into Model values(2,44,60);
101 • insert into Model values(3,34,70);
102 • insert into Model values(4,34,70);
103 • insert into Model values(5,15,50);
104 • insert into Model values(6,65,60);
105 • insert into Model values(7,34,60);
106 • insert into Model values(8,43,70);
107 • insert into Model values(9,45,40);
108 • insert into Model values(10,45,50);
109
110
111 •  desc employee;
112 •  select * from employee;
113 •  insert into employee(ssn,salary, union_mem_id) values(987698761,55000,67671);
--
```

Steps to run the java application:

1. Install the Spring tool suite IDE from the official STS website and run the application.
2. Extract the AirportSystem1.zip file from the /project folder.
3. In the STS side click on the File > Import and choose the directory of the extracted folder from step 2 and open it.



- Upon opening the folder, in the location `src/main/resources`, the file named "application.properties" has the property fields that contains the database credentials and database url, where we need to provide the url of the mysql server running, and the credentials which was saved while installing the mysql workbench in previous step. Here, `airport_system` refers to the database created in mysql. These are the very important details and must be proper to establish the connection with the mysql database.



- `src/main/java` folder contains the package called "com.example.demo" which contains all the java classes used for implementing the business logic of the application and also the connectivity from database to the frontend. Below screenshot shows the list of java classes of the application.

The screenshot shows the Spring Tool Suite interface. The Package Explorer on the left lists the project structure under the package com.example.demo. The main area displays the code for `AirportSystem1Application.java`:

```

1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class AirportSystem1Application {
7
8     public static void main(String[] args) {
9         SpringApplication.run(AirportSystem1Application.class, args);
10    }
11 }
12
13 }
14

```

- src/main/resources contains all the frontend related code in the form of html pages as shown in the screenshot below.

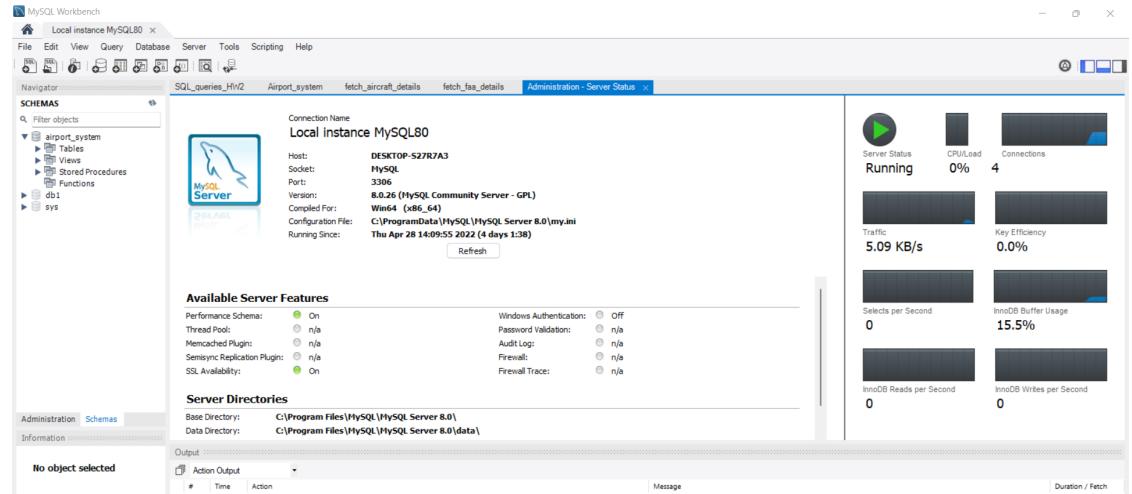
The screenshot shows the Spring Tool Suite interface. The Package Explorer on the left lists the project structure, including the static resources folder containing `Home.html`. The main area displays the code for `Home.html`:

```

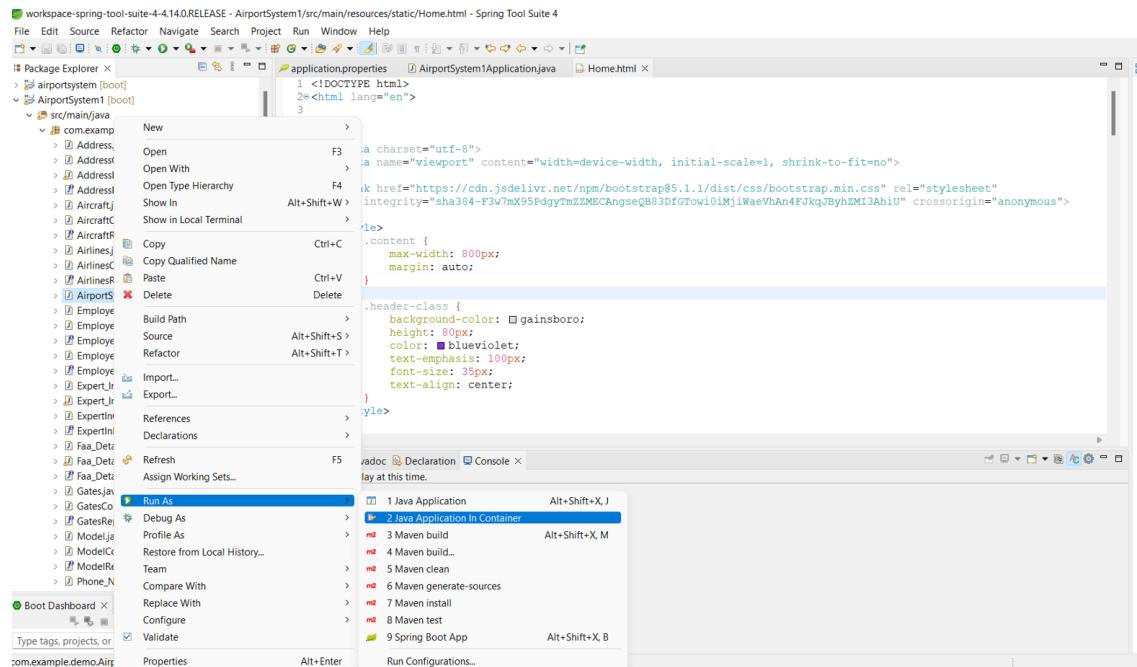
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7M7fKNGVjA8qVJ+QkXyEzOYUxXnZGZIu0lJWZJLZDd5h0Zo" crossorigin="anonymous">
<style>
    .content {
        max-width: 800px;
        margin: auto;
    }
    .header-class {
        background-color: #gainsboro;
        height: 80px;
        color: #blueviolet;
        text-emphasis: 10px;
        font-size: 35px;
        text-align: center;
    }
</style>
</head>

```

- Make sure mysql server status is running and the airport system database is created and its table and data is populated as said earlier.



- Click on the java file named as “AirportSystem1Application.java” under the location “/AirportSystem1/src/main/java/com/example/demo/AirportSystem1Application.java” and choose to run as “Java Application” as shown below to start the application.



Once the application is started the all the functionalities will be available to the users through the Front-end web pages, which are under the static folder.

Users can open the browser and use the url below to login or signup :

<http://<domain>:8080/Home.html> or <http://<domain>:8080/signup.html>

Replace <domain> with the “localhost” while the application is running on your own system and you are accessing the pages from the same system. Else use the ip address of the system where the application is running.

Then the user will be able to login or register as per the demo screenshots shown in section 10.

Later users will be redirected to appropriate pages and will be able to access all the functionalities of the application.

9. USER APPLICATION INTERFACE

The following is a description of the User interfaces, the functions that are offered by The Airport System to the users and how the functions are implemented in SQL.

The functions offered by the system are as depicted by the context diagram.

User - Technician

The technician's main page offers options to:

1. Update their expertise in models of aircrafts. Which is translated into a simple update operation in MySQL.
2. Add Test results of aircrafts they mend. Which is translated into an insert operation in MySQL.

The screenshot shows a web browser window with the URL `localhost:8080/TechnicalMainPage`. The title bar reads "ONLINE AIRPORT SYSTEM". The main content area is titled "TECHNICIAN MAIN PAGE" and contains two sections: "Add/Update Expertise" and "Add/Update Test details".

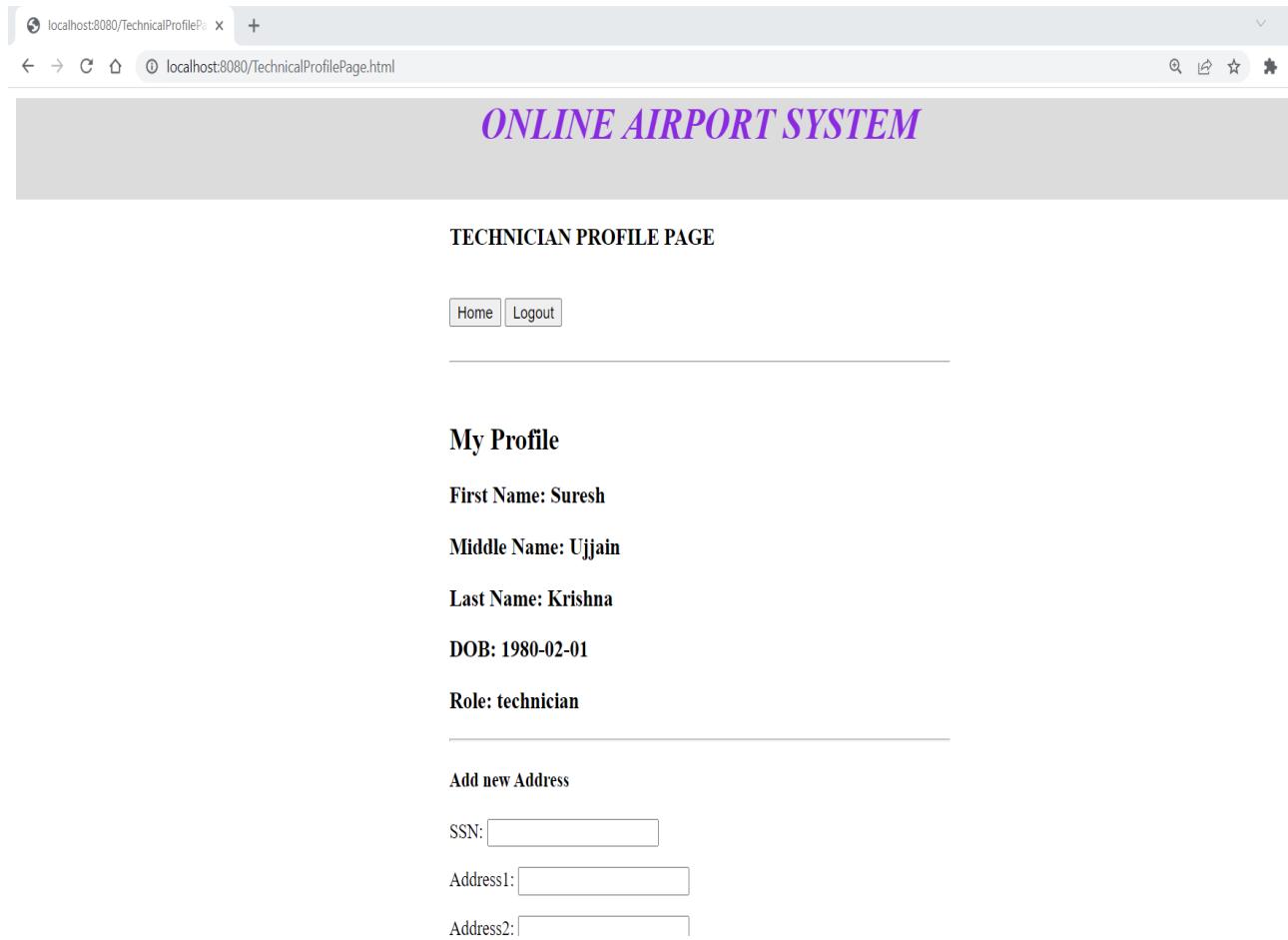
Add/Update Expertise

Employee SSN:
Area of expertise:

Add/Update Test details

Test ID:
Aircraft ID:
Score:
Date: mm/dd/yyyy
No of hours:

The technician profile page, displays and maintains up to date profile information of the Technicians.



A screenshot of a web browser displaying the 'ONLINE AIRPORT SYSTEM' technician profile page. The page title is 'TECHNICIAN PROFILE PAGE'. It shows a profile section with fields for First Name (Suresh), Middle Name (Ujjain), Last Name (Krishna), DOB (1980-02-01), and Role (technician). Below this is a form for adding a new address, with fields for SSN, Address1, and Address2.

ONLINE AIRPORT SYSTEM

TECHNICIAN PROFILE PAGE

Home Logout

My Profile

First Name: Suresh

Middle Name: Ujjain

Last Name: Krishna

DOB: 1980-02-01

Role: technician

Add new Address

SSN:

Address1:

Address2:

localhost:8080/TechnicalProfilePage.html

Middle Name: Ujjain
Last Name: Krishna
DOB: 1980-02-01
Role: technician

Add new Address

SSN:
Address1:
Address2:
City:
State:
Pincode:

Add new Phone number

SSN:
Phone Number:

localhost:8080/TechnicalProfilePage.html

Middle Name: Ujjain
Last Name: Krishna
DOB: 1980-02-01
Role: technician

Add new Address

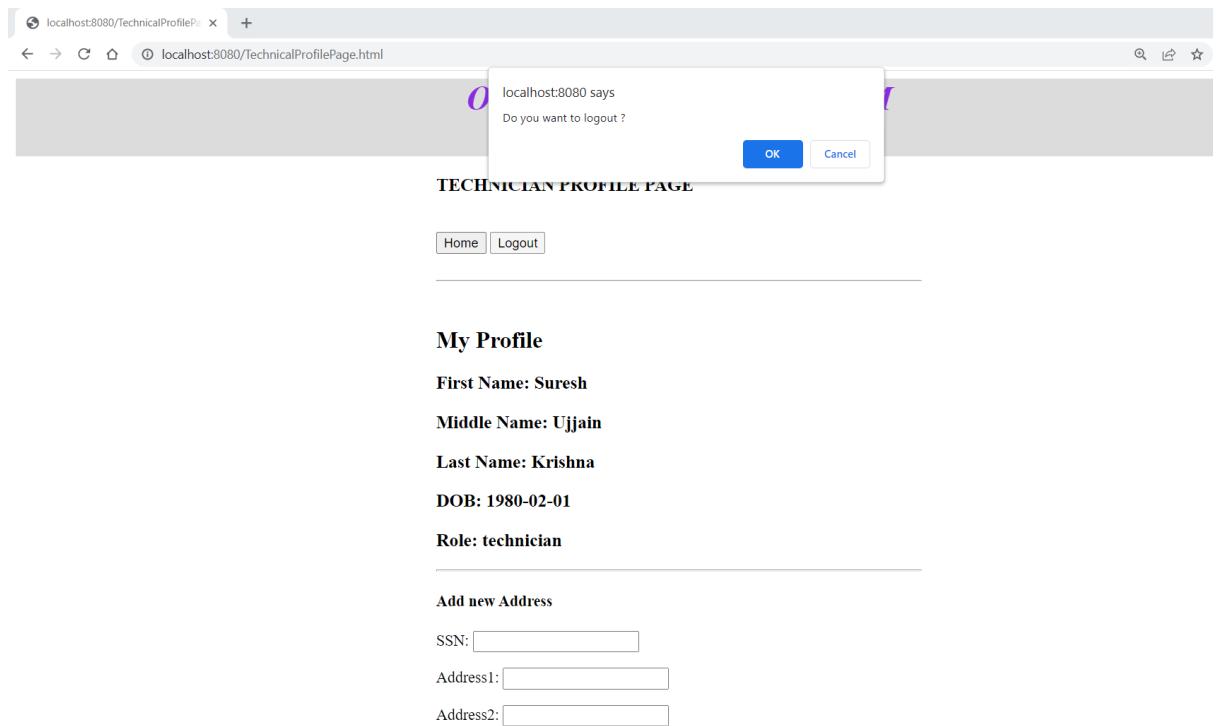
SSN: 987698764
Address1: Frankford Rd
Address2: Apt5656
City: Dallas
State: TX
Pincode: 75252

Add new Phone number

SSN: 987698764
Phone Number: 8989898

81°F Cloudy 2:20 PM 5/2/2022

Technician Logout page



User - Traffic Controller

The Traffic Controller's main page offers options to:

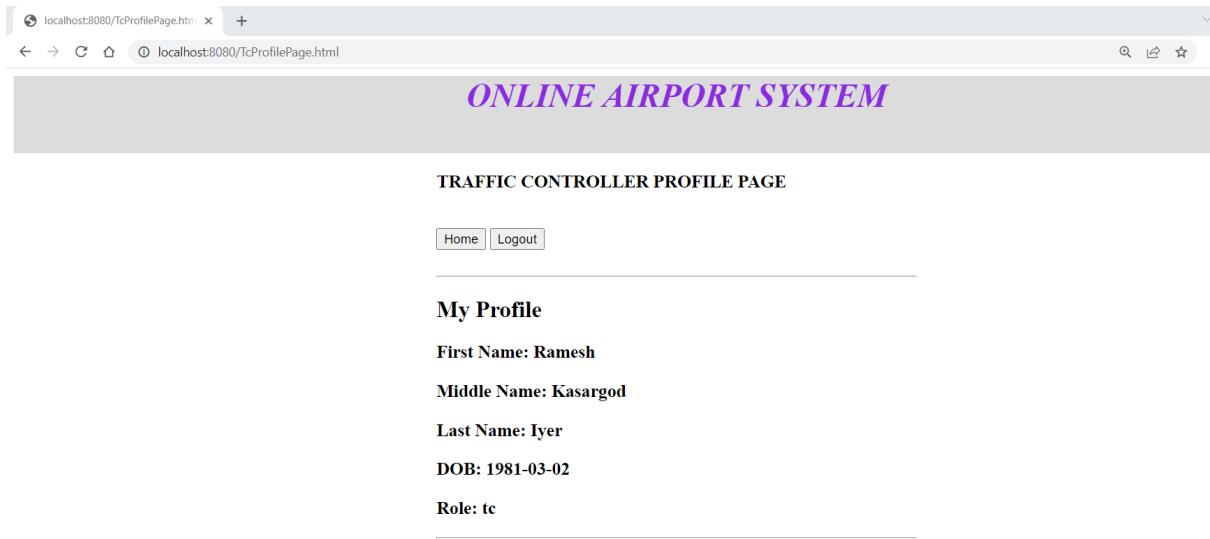
1. update his/her Most recent Medical Examinations details, which determine their status.
Which is translated into a simple update operation in MySQL.

A screenshot of a web browser window showing the "ONLINE AIRPORT SYSTEM" header. The page title is "TRAFFIC CONTROLLER MAIN PAGE". At the bottom, there are "Profile" and "Logout" buttons.

Add/Update Medical Exam details

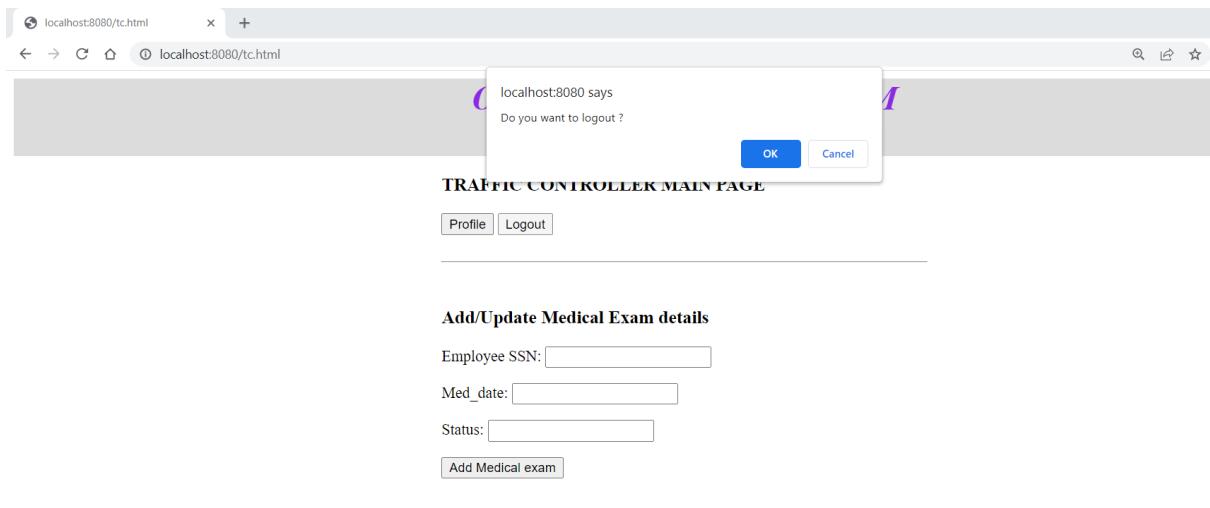
Employee SSN: 987698761
Med_date: 2022-03-03
Status: Done

The Traffic Controller profile page, displays and maintains up to date profile information of the Traffic Controllers..



A screenshot of a web browser window showing the 'TRAFFIC CONTROLLER PROFILE PAGE'. The title bar reads 'localhost:8080/TcProfilePage.htm'. The main content area has a header 'ONLINE AIRPORT SYSTEM' and a sub-header 'TRAFFIC CONTROLLER PROFILE PAGE'. Below this, there are two buttons: 'Home' and 'Logout'. A horizontal line separates this from the 'My Profile' section. Under 'My Profile', the following details are listed:
First Name: Ramesh
Middle Name: Kasargod
Last Name: Iyer
DOB: 1981-03-02
Role: tc

Traffic Controller Logout page



A screenshot of a web browser window showing a confirmation dialog box. The dialog box contains the message 'localhost:8080 says Do you want to logout ?' with 'OK' and 'Cancel' buttons. Behind the dialog, the 'TRAFFIC CONTROLLER MAIN PAGE' is visible, featuring 'Profile' and 'Logout' buttons. A horizontal line separates this from the 'Add/Update Medical Exam details' section. Under 'Add/Update Medical Exam details', there are four input fields: 'Employee SSN:' (with a placeholder box), 'Med_date:' (with a placeholder box), 'Status:' (with a placeholder box), and a button 'Add Medical exam'.

User - Airport Operations Authority

The airport Authority mailpage offers options like:

1. View all aircraft details, which translates to a select all operation in SQL.

2. Update Aircraft details, which translates to a update operation in SQL.
3. View all model Details of aircrafts.
4. Add and assign gates to airlines, which translates to insert operation in SQL.
5. View all airlines.
6. Get Test details based on test_ID, translates to a conditional select in SQL.
7. Get Test details of aircrafts with a test score greater than the given value, translates to a conditional select in SQL.

ONLINE AIRPORT SYSTEM

Airport Operations Authority - Main page

[Home](#) [Logout](#)

View all Aircrafts

[View all Aircraft details](#)

Add/Update Aircraft details

Aircraft ID:

Model Id:

Flight Status:

Airline ID:

[Add Aircraft Details](#) [Update Aircraft details](#)

Airport Operations Authority - Main page

[Home](#) [Logout](#)

View all Aircrafts

[View all Aircraft details](#)

aircraft_id	model_id	flight_status	airline_id
7777	7	repair	888
66666	6	inair	888
222888	2	repair	222
656565	7	repair	888
2228881	5	inAir	222
2228882	6	landed	222
2228883	1	inAir	222
2228884	2	landed	222
3338881	7	landed	333
3338882	1	inAir	333
3338883	4	inAir	333
4448881	8	landed	444
5558881	9	landed	555
6668881	1	inAir	666
7778881	2	landed	777
8888881	3	inAir	888
9998881	4	landed	999

localhost:8080/aoa.html

Add/Update Aircraft details

Aircraft ID::	9998881
Model Id:	5
Flight Status:	inair
Airline ID:	999

View all Model details

Model ID Weight Capacity

undefined 20	50
undefined 44	60
undefined 34	70
undefined 34	70
undefined 15	50
undefined 65	60
undefined 34	60
undefined 43	70
undefined 45	40
undefined 45	50
undefined 55	55
undefined 5	55

localhost:8080/aoa.html

Add Gates

Airline ID::	888
Gate Id:	2222

Get all Airlines

Airline ID Gate ID

222	1111
222	1112
222	1177
333	1113
333	1114
333	1115
444	1116
444	1117
555	1118
666	1119
777	1120
888	1121
888	2222
999	1122
999	1123
999	1124

79°F
Mostly cloudy



2:45 PM 5/2/2022

Test ID	Aircraft ID	Score	Date	Number_of_Hrs
555	1118			
666	1119			
777	1120			
888	1121			
888	2222			
999	1122			
999	1123			
999	1124			

Get specific test detail				
Test ID: <input type="text" value="5555"/>				
<input type="button" value="Get Faa test"/>				
Test ID Aircraft ID Score Date Number_of_Hrs				
5555 2228884 5 2021-02-01 7				

Get all test details above the score entered				
Enter Test Score:				
<input type="text" value="90"/>				
<input type="button" value="Get test details >= score"/>				
Test ID Aircraft ID Score Date Number_of_Hrs				
7779 2228882 98 2022-08-07 6				
7789 8888881 99 2022-08-06 6				
7790 9998881 90 2022-09-05 5				

10. CONCLUSIONS AND FUTURE WORK

Our goal to design an Online Airport System is successfully completed which contains all the information related to the airport. This includes information about all the airplanes that are stationed and maintained at the airport. System serves various types of users like technicians, employees, traffic controllers and the Administrator.

We started the design with a context diagram and then gathered functional and non-functional requirements. This was followed up with the conceptual design of the database along with logical database schema and database normalization. After designing the system, we started the implementation using SQL queries for the backend. To render the data and for UI, we used HTML, CSS, JavaScript for smooth interaction for the users. Connection of the backend data and frontend was done using Java Springboot.

The future scope includes extending the system to help users access the system via mobile and involve more functionality like tracking and security.

11. REFERENCES

Fundamentals of Database Systems, 7th Edition, R. Elmasri and S. B. Navathe, AddisonWesley