A Database Management System Course Project Report on

Airport Management System

Undergone at

National Institute Of Technology Karnataka

Under the guidance of

Dr. M. Venkatesan and

Ms. Marwa Mohiddin

Submitted By

Mehnaz Yunus (16CO124) and

Sharanya Kamath (16CO140)

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science & Engineering

Department of Computer Science and Engineering

National Institute of Technology Karnataka

July-December, 2018

To whom it may concern

This is to certify that Mehnaz Yunus and Sharanya Kamath have worked on the project titled *Airport Management System* under my guidance at the Computer Science and Engineering Department, National Institute of Technology Karnataka, Surathkal, as part of Database Management System Course Project during July 2018- December 2018.

Signature

DECLARATION

I certify that the report on 'Title' which is being submitted as record of my Database Management System Course Project is a bonafide report of the work carried out by me. The material contained in this report has not been submitted to any university or Institution for the award of any degree.

Signature
Mehnaz Yunus (16CO124)
Sharanya Kamath (16CO140)

Abstract

Airport Management System is a web application, developed to maintain the details of passengers, security personnel and flight staff in an airport. It maintains the information about the personal details of the passengers and their bookings. The passengers have the ability to generate and download their flight ticket in a pdf format. The details of the security personnel and the flight staff for each flight is also stored. They can log in and perform their respective functionalities such as clearing and checking in passengers and generating the flight report.

This system is an application developed in python3.5. It is user-friendly and very intuitive. It is fast and can perform many operations which are necessary for an airport. It is simple to understand and can even be used by people who are not even familiar with the workings of an airport.

This software package has been developed using HTML, CSS, Bootstrap and Javascript at Front
End and Django at Back End with Microsoft SQL Server database. This version of the software has a multi-user approach. For further enhancement or development of the package, user’s feedback will be considered.

CONTENTS

List of Tables and Figures

Introduction

1.1 Purpose:

Airport Management system is an application that enables users to view and book flights. The application also provides facilities of a staff and security system through which staff can check-in and clear passengers for take-off. This application is useful for both the passenger and staff side of an airport system.

The application is developed using Django in Python.

1.2 Objective :

In this world of growing technologies everything has been computerized. With large amount of work opportunities the human workforce has increased. Thus there is a need of a system which can handle the data of such a large number of passengers, security and flight staff in an airport. This project simplifies the task of booking and managing flights because of its user friendly nature.

Requirement Analysis and Specifications

The aim of the system is to develop an "Airport Management System" software, which automates the process to create and store passenger, security and staff details . The system is supposed to be used in an airport where multi-user functionality is required. Security clearance, check-in, ticket generation can be done through this system, thus saving manual labour and increasing convenience.

2.1 Software Requirements

DBMS : MySQL Server version: 5.7.23

Backend : Django version 2.1.3

Development tool : PyCharm

Operating System : Ubuntu 16.04/18.04

2.2 Hardware Requirements

Processor: Intel(R) Xeon(R)

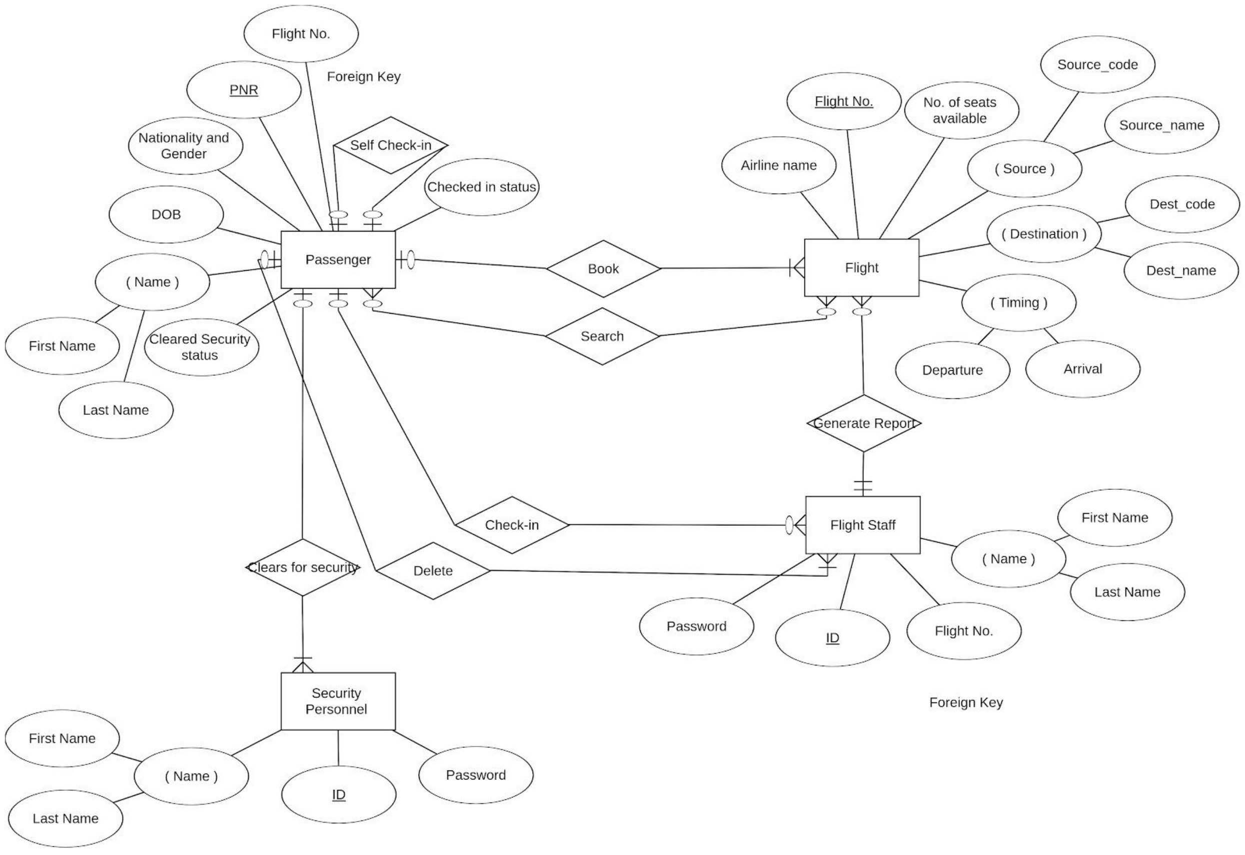
CPU: E5-2680 v4 @ 2.40GHz

RAM: 8GB.

Hard Disk space : 500GB

Architecture : x86-64

3.1. ER Diagram



This ER diagram represents the model of Airport management System. The entity-relationship diagram of shows all the visual instruments of database tables and relationship between Passenger, Flight, Security Personnel, Flight Staff etc. It uses structured data and defines the relationship between structured data groups of Airport management System functionalities. The main entities of Airport Management System are Flight, Passenger, Security Personnel and Flight Staff.

Flight Entity:

Attribute Name	**Meaning**	**Value**
Flight No.	Uniquely identifies the flight.	Integer (default=1007)
Airline Name	Specifies which airline the flight belongs to.	Varchar (50)

No. of seats available	The numbers of unbooked seats in the flight.	Integer (default=0)
Source Name	Name of place from which the flight starts.	Varchar (50)
Source Code	A 3 letter abbreviation of the source.	Varchar (3)
Destination Name	Name of the place to which the flight will go.	Varchar (50)
Destination Code	A 3 letter abbreviation of the destination. This appears in the PNR of the passenger.	Varchar (3)
Departure	Date and time of departure of flight from source.	DateTime
Arrival	Date and time of arrival of flight to destination.	DateTime

Passenger Entity

Attribute Name	**Meaning**	**Value**
PNR	Uniquely identifies the passenger. It contains the flight no. and destination code as well.	Varchar (10)
First Name	First name of passenger.	Varchar (50)
Last Name	Last name of passenger.	Varchar (50)
Date of Birth	Date of birth of passenger.	Date

Nationality	The nationality of the passenger.	Varchar (50)
Gender	Specifies whether the passenger is male (M) or female (F).	Varchar (1)
Flight no.	Identifies the flights which the passenger has booked.	Foreign key to flight no. in Flight table.
Checked in Status	Specifies whether the passenger has checked in.	Boolean (default=False)
Cleared Security Status	Specifies whether passenger has been cleared by security, and who has cleared the passenger.	Integer (default=0)

Security Personnel Entity

Attribute Name	**Meaning**	**Value**
ID	Uniquely identifies the security personnel.	Integer
First Name	First name of security personnel.	Varchar (50)
Last Name	Last name of security personnel.	Varchar (50)
Password	Password for secure purposes.	Varchar (255)

Flight Staff Entity

Attribute Name	Meaning	Value
ID	Uniquely identifies the flight staff person.	Integer
First Name	First name of staff person.	Varchar (50)
Last Name	Last name of staff person.	Varchar (50)
Password	Password for secure purposes.	Varchar (255)
Flight no.	Identifies the flight for which the staff person works.	Foreign key to flight no. in Flight table.

Description of Airport Management System :

The entity is a concept or object in which the piece of information can be stored. There are three types of relationship between entities. They are as follows:

● One to One(1-1): This relationship specifies that one instance of an entity is associated with another instance of an entity.

● One to Many(1-N): This relationship specifies that one instance of an entity is associated with zero or many other instances of another entity.

● Many to Many(M-N): This relationship specifies that one instance of an entity is associated with zero or many other instances of another entity.

write description about the entity relationship with respect to your project.

Entity 1	Entity 2	Relationship Name	Relationship Type	Description
Passenger	Flight	Book	1-N	One passenger can

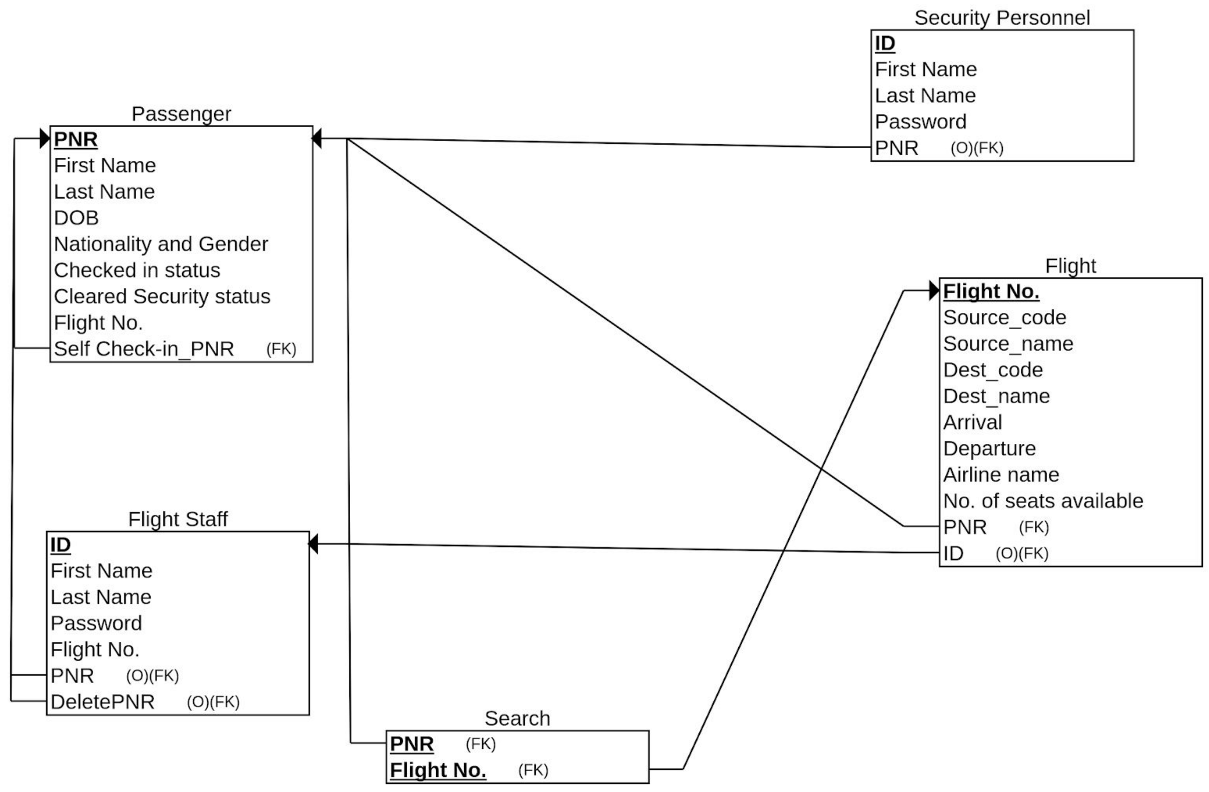
				make only one flight booking. One flight can have many passengers.
Passenger	Flight	Search	M-N	One passenger can search many flights. One flight can be searched by many passengers.
Flight	Flight Staff	Generate Report	1-N	One flight staff can generate one flight's report. One flights report can be generated by many flight staffs.
Passenger	Flight Staff	Delete	N-1	One passenger can be deleted by one flight staff. One flight staff can delete many passengers.
Passenger	Security	Clear for security	N-1	One passenger can be cleared by one security personnel. One security personnel can clear many passengers.
Passenger	Passenger	Self Check-in	1-1	One passenger can perform self check-in only for themselves.
Passenger	Flight Staff	Check-in	N-1	One passenger can be checked in by one flight staff. One flight staff can check-in many passengers.

3.2 Relational Database design

In this section we give a description of Mapping of ER diagram with Relation schema along with schema diagram of Airport Management System.

A relational database schema is the tables, columns and relationships that make up a relational database. There are two steps to creating a relational database schema: creating the

logical schema and creating the physical schema. The logical schema depicts the structure of the database, showing the tables, columns and relationships with other tables in the database and can be created with modeling tools or spreadsheet and drawing software. The physical schema is created by actually generating the tables, columns and relationships in the relational database management software (RDBMS). Most modeling tools can automate the creation of the physical
schema from the logical schema, but it can also be done by manually.



The application Airport Management System consists of the entities described below. Some entities are related other entities with the help of primary key - foreign key pair. The foreign key is used in establishing relation with the other table. Hence it is called relational database system.

1. Passenger

Attributes: pnr, first_name, last_name, dob, nationality, gender, checked_in_status, cleared_security_status

Foreign Key: flight_no

2. Flight

Attributes: flight_no, airline_name, no_of_seats, source, source_code, destination, destination_code, arrival_time, departure_time

Foreign Key: -

3. Flight Staff

Attributes: id, first_name, last_name, password

Foreign Key: flight_no

4. Security

Attributes: id, first_name, last_name, password

Foreign Key: -

3.3 Constraints in Relation Schema:

- **Key Constraints**

Entity	**Primary Key**	**Foreign Key**
Passenger	PNR	flight_no
Flight	flight_no	-
Flight Staff	id	flight_no
Security	id	-

- **Cardinality Ratio**

Entity 1	**Entity 2**	**Relationship Name**	**Cardinality Ratio**	**Description**
Passenger	Flight	Book	1-N	One passenger can make only one flight booking. One flight can

				have many passengers.
Passenger	Flight	Search	M-N	One passenger can search many flights. One flight can be searched by many passengers.
Flight	Flight Staff	Generate Report	1-N	One flight staff can generate one flight's report. One flights report can be generated by many flight staffs.
Passenger	Flight Staff	Delete	N-1	One passenger can be deleted by one flight staff. One flight staff can delete many passengers.
Passenger	Security	Clear for security	N-1	One passenger can be cleared by one security personnel. One security personnel can clear many passengers.
Passenger	Passenger	Self Check-in	1-1	One passenger can perform self check-in only for themselves.
Passenger	Flight Staff	Check-in	N-1	One passenger can be checked in by one flight staff. One flight staff can check-in many passengers.

● **Participation constraints**

Entity 1	**Entity 2**	**Relationship Name**	**Participation Constraint**	**Description**
Passenger	Flight	Book	1 : 0	Each passenger must book a flight to become a passenger. But there can be flights which have no passengers.
Passenger	Flight	Search	0 : 0	Not every passenger searches for a flight. Not every flight gets searched by a passenger.
Flight	Flight Staff	Generate Report	1 : 0	A report is generated for every flight by one or more of its flight staff. Not every flight staff generates a report.
Passenger	Flight Staff	Delete	1 : 0	Each passenger should be deleted after the flight takes off. Not every flight staff deletes a passenger.
Passenger	Security	Clear for security	1 : 0	Each passenger gets cleared for security. Not every security personnel clears a passenger.
Passenger	Passenger	Self Check-in	0 : 0	Not every passenger performs self check-in for himself.
Passenger	Flight Staff	Check-in	0 : 0	Not every passenger is checked-in by the flight staff i.e. he can perform self check-in. Not every flight staff checks-in passengers.

● **Structural constraints**

Passenger Entity

Attribute Name	**Meaning**	**Value**
first_name	First name of passenger	Varchar (50)
last_name	Last name of passenger	Varchar (50)
dob	Date of birth of passenger	Date
nationality	Nationality of passenger	Varchar (50)
gender	Gender of passenger	Varchar (1), either ‘M’ or ‘F’
checked_in_status	Tells whether the passenger has been checked-in	Boolean
cleared_security_status	Tells whether the passenger has cleared security	Integer, 0 if not cleared. id of security if cleared.

Flight Entity

Attribute Name	**Meaning**	**Value**
airline_name	Name of the airline to which the flight belongs	Varchar (50)
no_of_seats	Number of vacant seats on the flight	Integer (default=0)
source	Name of place from which the flight starts.	Varchar (50)
source_code	A 3 letter abbreviation of the source.	Varchar (3)
destination	Name of the place to which the flight will go.	Varchar (50)

destination_code	A 3 letter abbreviation of the destination. This appears in the PNR of the passenger.	Varchar (3)
arrival_time	Date and time of departure of flight from source.	DateTime
departure_time	Date and time of arrival of flight to destination.	DateTime

Security Entity

Attribute Name	**Meaning**	**Value**
ID	Uniquely identifies the security personnel.	Integer
First Name	First name of security personnel.	Varchar (50)
Last Name	Last name of security personnel.	Varchar (50)
Password	Password for secure purposes.	Varchar (255)

Flight Staff Entity

Attribute Name	**Meaning**	**Value**
ID	Uniquely identifies the flight staff person.	Integer
First Name	First name of staff person.	Varchar (50)

Last Name	Last name of staff person.	Varchar (50)
Password	Password for secure purposes.	Varchar (255)
Flight no.	Identifies the flight for which the staff person works.	Foreign key to flight no. in Flight table.

Project Components

4.1 Front End Design:

The application framework we have used is Django. It is a free and open-source web framework written in Python, which follows the Model-View-Template (MVT) architectural pattern. Django's primary goal is to ease the creation of complex, database-driven websites. Python is used throughout, even for settings files and data models.

The *Model* maintains the relationship between the objects and the database and handles validation, association, transactions, and more.
View is the facility within the application that directs traffic, on the one hand, querying the models for specific data, and on the other hand, organizing that data (searching, sorting, messaging it) into a form that fits the needs of a given template.
Template is a presentation of data in a particular format, triggered by a view's decision to present the data. They are script-based template systems like JSP, ASP, PHP, and very easy to integrate with AJAX technology.
For frontend we have used HTML, CSS and Bootstrap. To introduce some dynamic elements in our application we have used Javascript and jQuery.

File Structure for frontend is as follows:
Templates
All the HTML files are stored in this folder. These are used to render the web pages when a particular view calls them.

Models.py
This file contains the definition of the structure of the Airport management system database. It defines all the entities and its attributes, along with primary - foreign key pairs.

Views.py
Variable declaration , querying the database etc. is done in this file. The functionality for each URL is defined here

The application has the following modules:

1. **Creation of security/staff profiles:** The airport management system admin is responsible for creating verified accounts for the security and flight staff. The details and ID of each user is entered during profile creation. The password entered by the user is converted to a hash and stored in the database so that even if the database is compromised, external entity cannot access the user's details as a hash cannot be traced back to the password the user entered.

2. **Login as Security/Staff**: After the admin creates profiles for security staff, they can login to access their personal profile using their individual credentials and they will be redirected to their respective pages. The django-auth system takes care of authenticating users and verifying passwords. Passwords are stored in a secure manner using hashing.

3. **Home page:** This page is accessible without logging in. User can search flights based on source and destination and book suitable flights. Passengers can also view their previous bookings by entering their PNR here.

4. **Security clearance:** This page is private to authorised security personnel. Security staff can clear checked-in passengers for security and also view their profile details. The ID of the security staff is stored as the cleared security status of the passenger.

5. **Staff home:** This page displays details of all passengers of the particular flight. Flight staff can check in passengers for the flight. They can also generate the report of all checked in passenger, which creates a PDF with details of passengers. After this, they can clear the flight for take off. Once this is done, all the details of passengers of the cleared flight are deleted from the database.

6. **View flights page:** User can view all available flights and search based on source and destination airports. They can also book required flights. Selecting required flight redirects to the booking page.

7. **Book flights:** Passenger fills in all the required details. A unique PNR is generated for each passenger booking. The passenger is redirected to the passenger home page on completion of booking process.

8. **Passenger home:** Passenger can enter their booking reference(PNR) to get details of their booking. Passenger can also opt to self check in. They can download e-Ticket with their booking details.

9. **Feedback:** Users can give feedback about the website by selecting the corresponding option on the home page.

4.2 Security Measures:

Below is a list of possible vulnerabilities that may occur while creating an application and how using Django helps avoid them.

1. SQL INJECTIONS

SQL injection is a type of attack where a malicious user is able to execute arbitrary SQL code on a database. This can result in records being deleted or data leakage.

Django's querysets are protected from SQL injection since their queries are constructed using query parameterization. A query's SQL code is defined separately from the query's parameters. Since parameters may be user-provided and therefore unsafe, they are escaped by the underlying database driver.

2. CROSS SITE SCRIPTING (XSS)

Cross-site scripting (XSS) attacks allow a user to inject client side scripts into the browsers of other users. This is usually achieved by storing the malicious scripts in the database where it will be retrieved and displayed to other users, or by getting users to click a link which will cause the attacker's JavaScript to be executed by the user's browser. However, XSS attacks can originate from any untrusted source of data, such as cookies or Web services, whenever the data is not sufficiently sanitized before including in a page.

Using Django templates protects you against the majority of XSS attacks. Django templates escape specific characters which are particularly dangerous to HTML. This protects users from most malicious input.

3. CROSS-SITE REQUEST FORGERY (CSRF)

Cross-Site Request Forgery(CSRF) attacks allow a malicious user to execute actions using the credentials of another user without that user's knowledge or consent. A third-party website will send a request to a web application that a user is already authenticated against (e.g. their bank). The attacker can then access functionality via the victim's already authenticated browser. Targets include web applications like social media, in browser email clients, online banking, and web interfaces for network devices.

Django has built-in protection against most types of CSRF attacks, providing you have enabled and used it where appropriate. CSRF protection works by checking for a secret in each POST request. This ensures that a malicious user cannot simply "replay" a form POST to your website and have another logged in user unwittingly submit that form. The malicious user would have to know the secret, which is user specific (using a cookie).

4. CLICKJACKING PROTECTION

Clickjacking is a type of attack where a malicious site wraps another site in a frame. This attack can result in an unsuspecting user being tricked into performing unintended actions on the target site.

Django contains clickjacking protection in the form of the **`X-Frame-Options middleware`** which in a supporting browser can prevent a site from being rendered inside a frame.

5. BROKEN AUTHENTICATION & SESSION MANAGEMENT

Broken authentication and session management encompass several security issues, all of them having to do with maintaining the identity of a user. If authentication credentials and session identifiers are not protected at all times an attacker can hijack an active session and assume the identity of a user.

6. INSECURE DIRECT OBJECT REFERENCES

Insecure direct object reference is when a web application exposes a reference to an internal implementation object. Internal implementation objects include files, database records, directories, and database keys. When an application exposes a reference to one of these objects in a URL hackers can manipulate it to gain access to a user's personal data.

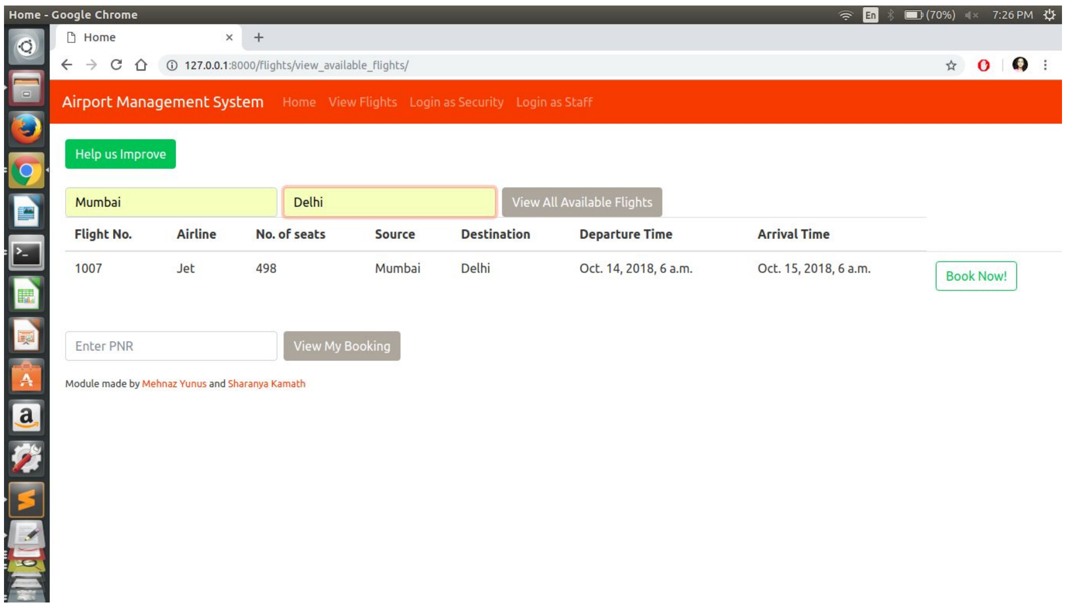
7. SECURITY MISCONFIGURATION

Security misconfiguration encompasses several types of vulnerabilities all centered on a lack of maintenance or a lack of attention to the web application configuration. A secure configuration must be defined and deployed for the application, frameworks, application server, web server, database server, and platform. Security misconfiguration gives hackers access to private data or features and can result in a complete system compromise.

Results & Discussion

1. Home Page

This is the homepage of the Airport Management System. The navbar gives an intuitive set of options which the user can use to login or view flights. The user can also search the available flights of a particular source and destination. By entering his PNR, the user can view his booking.



Home - Google Chrome

127.0.0.1:8000/flights/view_available_flights/

Airport Management System Home View Flights Login as Security Login as Staff

Help us Improve

Mumbai | Delhi | View All Available Flights

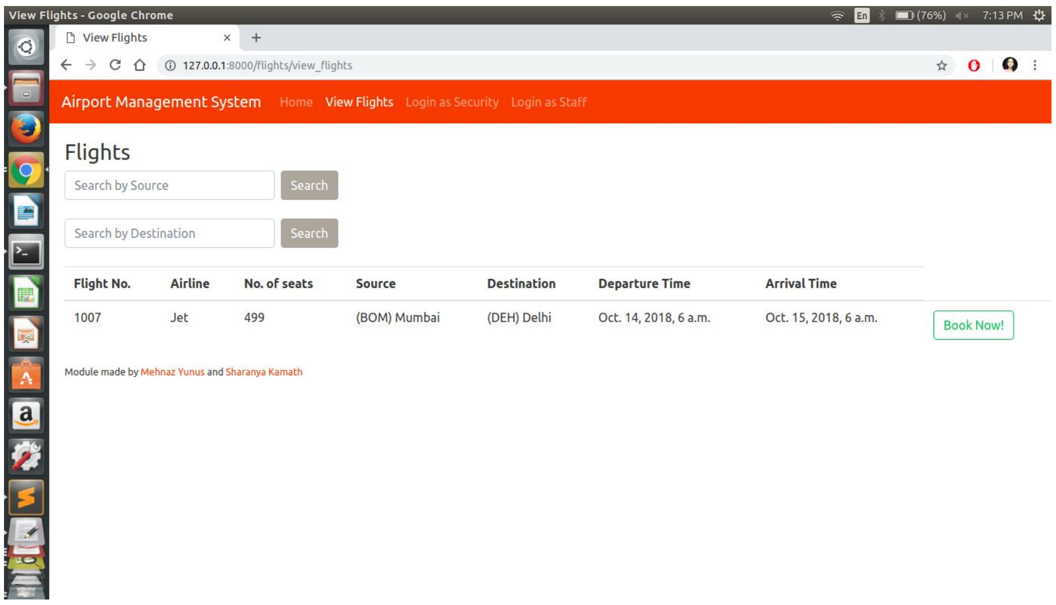
Flight No.	Airline	No. of seats	Source	Destination	Departure Time	Arrival Time	
1007	Jet	498	Mumbai	Delhi	Oct. 14, 2018, 6 a.m.	Oct. 15, 2018, 6 a.m.	Book Now!

Enter PNR | View My Booking

Module made by Mehnaz Yunus and Sharanya Kamath

2. View Flights

The user can view all available flights here, and click on 'book now' to book them. He can search by source or destination.



View Flights - Google Chrome

127.0.0.1:8000/flights/view_flights

Airport Management System Home View Flights Login as Security Login as Staff

Flights

Search by Source | Search

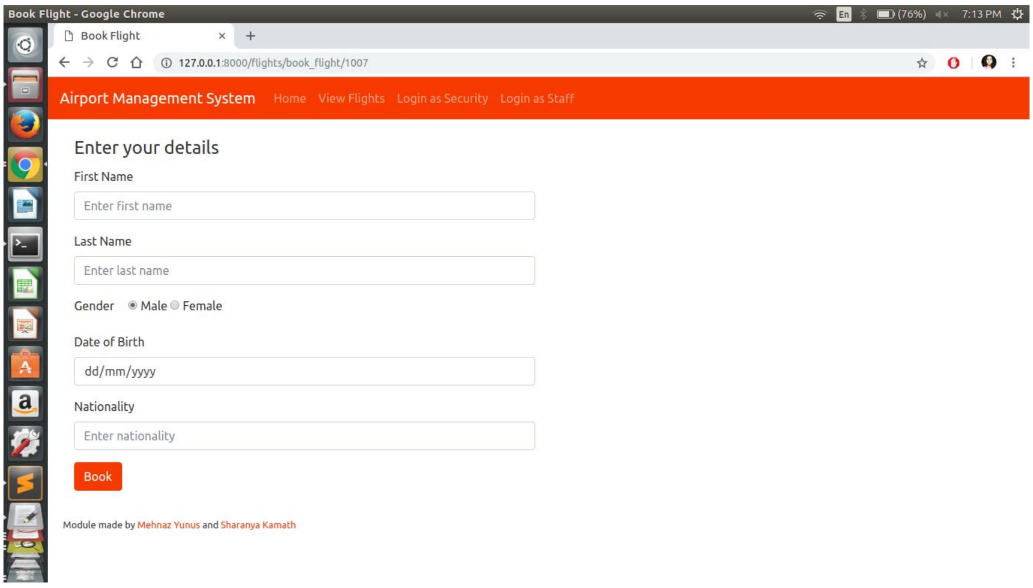
Search by Destination | Search

Flight No.	Airline	No. of seats	Source	Destination	Departure Time	Arrival Time	
1007	Jet	499	(BOM) Mumbai	(DEH) Delhi	Oct. 14, 2018, 6 a.m.	Oct. 15, 2018, 6 a.m.	Book Now!

Module made by Mehnaz Yunus and Sharanya Kamath

3. Book Flight

The passenger has to enter his details to book a flight. The PNR is automatically computed by taking into account flight_no and destination_code.



Book Flight - Google Chrome

127.0.0.1:8000/flights/book_flight/1007

Enter your details

First Name: Enter first name

Last Name: Enter last name

Gender [✓ Male] Female

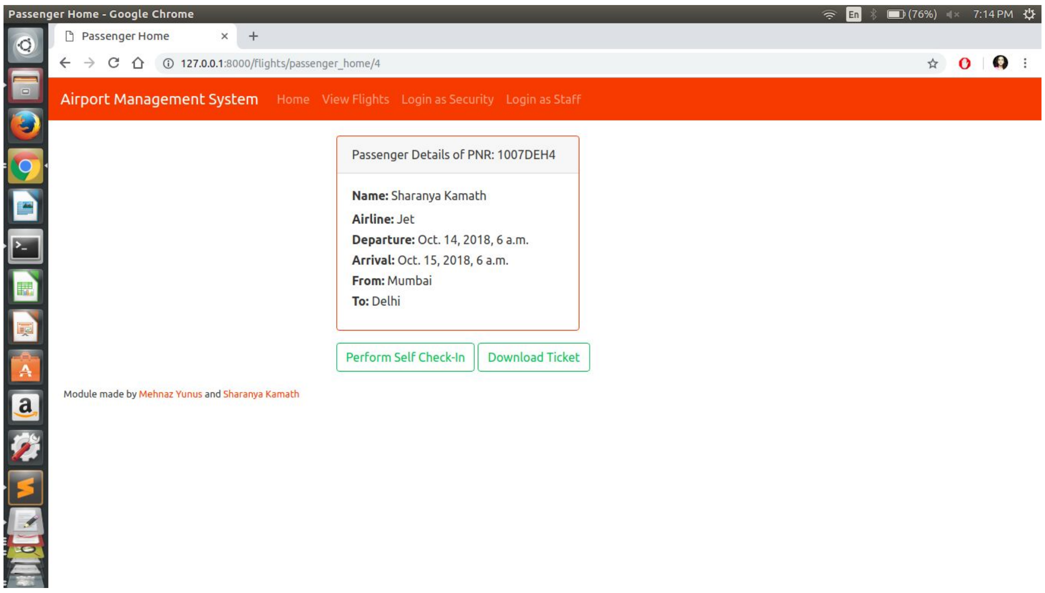
Date of Birth: dd/mm/yyyy

Nationality: Enter nationality

Module made by Mehnaz Yunus and Sharanya Kamath

4. Passenger Home Page

The passenger can view details of his booking and also perform self check-in. He can also download the pdf copy of his ticket.



Passenger Home - Google Chrome

127.0.0.1:8000/flights/passenger_home/4

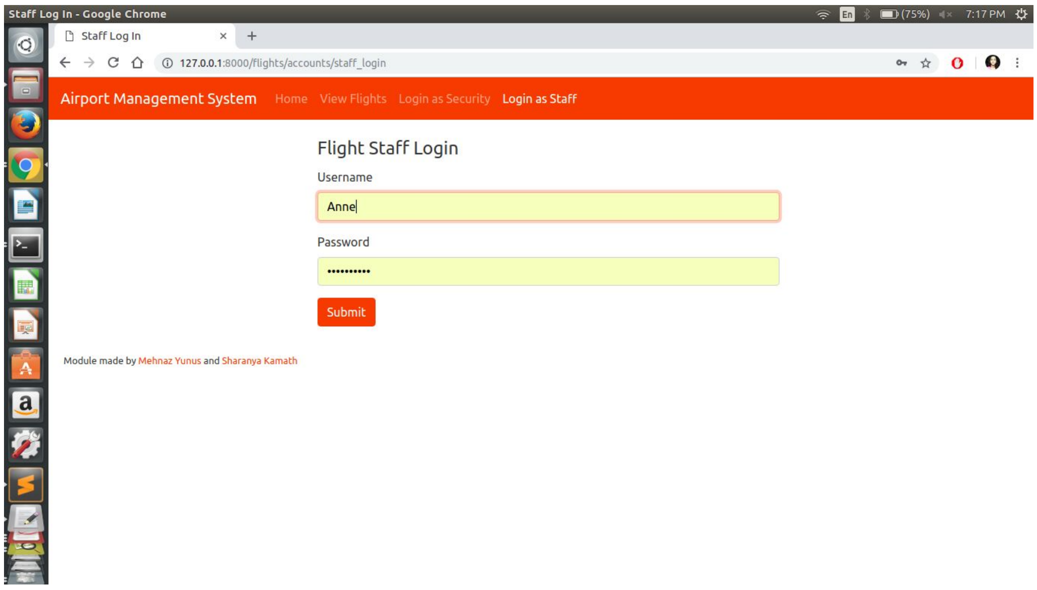
Passenger Details of PNR: 1007DEH4

Name: Sharanya Kamath
Airline: Jet
Departure: Oct. 14, 2018, 6 a.m.
Arrival: Oct. 15, 2018, 6 a.m.
From: Mumbai
To: Delhi

Module made by Mehnaz Yunus and Sharanya Kamath

5. Staff Login

Flight Staff can login to the system via this page.



Staff Log In - Google Chrome

Staff Log In

127.0.0.1:8000/flights/accounts/staff_login

Airport Management System Home View Flights Login as Security Login as Staff

Flight Staff Login

Username: Anne

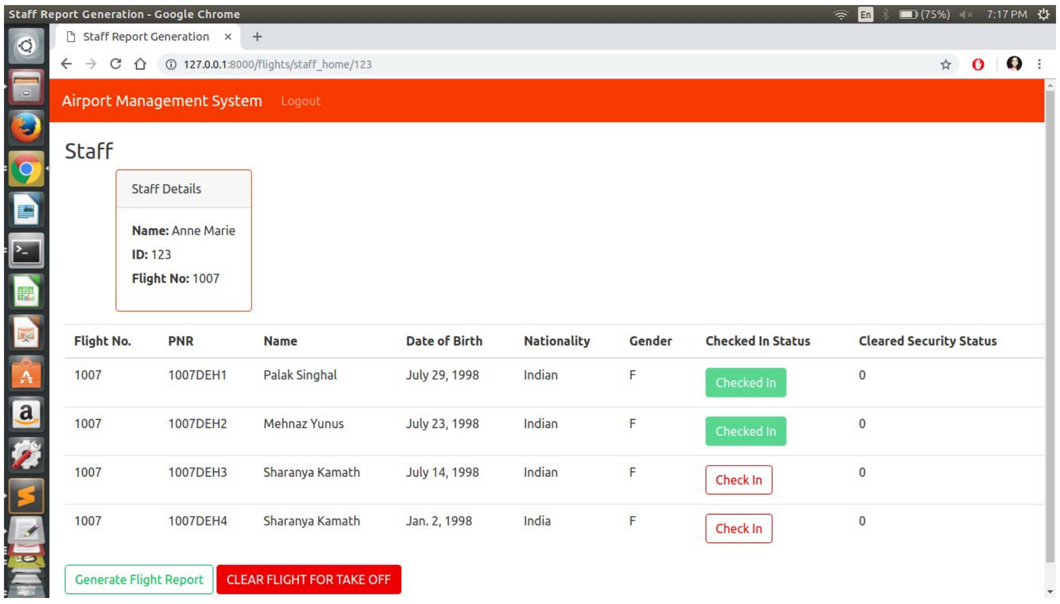
Password: ••••••••••

Submit

Module made by Mehnaz Yunus and Sharanya Kamath

6. Staff Home Page

After logging in, the staff person can view details of all the passengers of his flight. He can check-in those passengers who haven’t performed self check-in. He can generate the flight report and also clear the flight i.e. delete all passengers.



Staff Report Generation - Google Chrome

Staff Report Generation

127.0.0.1:8000/flights/staff_home/123

Airport Management System Logout

Staff

Staff Details

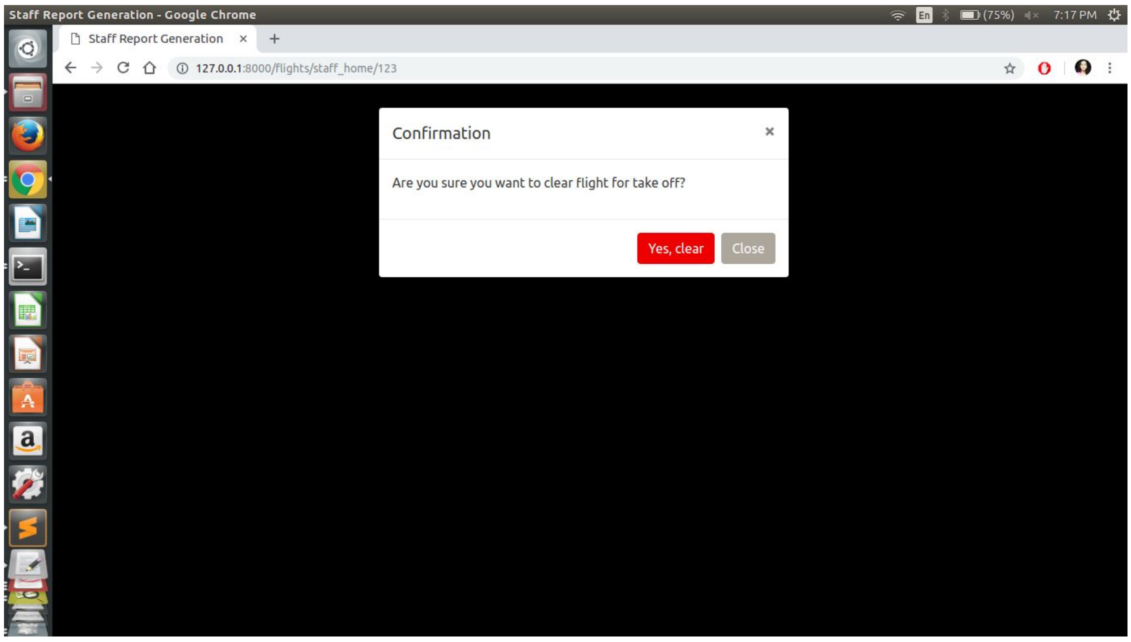
Name: Anne Marie
ID: 123
Flight No: 1007

Flight No.	PNR	Name	Date of Birth	Nationality	Gender	Checked In Status	Cleared Security Status
1007	1007DEH1	Palak Singhal	July 29, 1998	Indian	F	Checked In	0
1007	1007DEH2	Mehnaz Yunus	July 23, 1998	Indian	F	Checked In	0
1007	1007DEH3	Sharanya Kamath	July 14, 1998	Indian	F	Check In	0
1007	1007DEH4	Sharanya Kamath	Jan. 2, 1998	India	F	Check In	0

Generate Flight Report   CLEAR FLIGHT FOR TAKE OFF

7. Confirmation to delete all passengers

The staff person receives a confirmation dialog box before deleting all passengers.



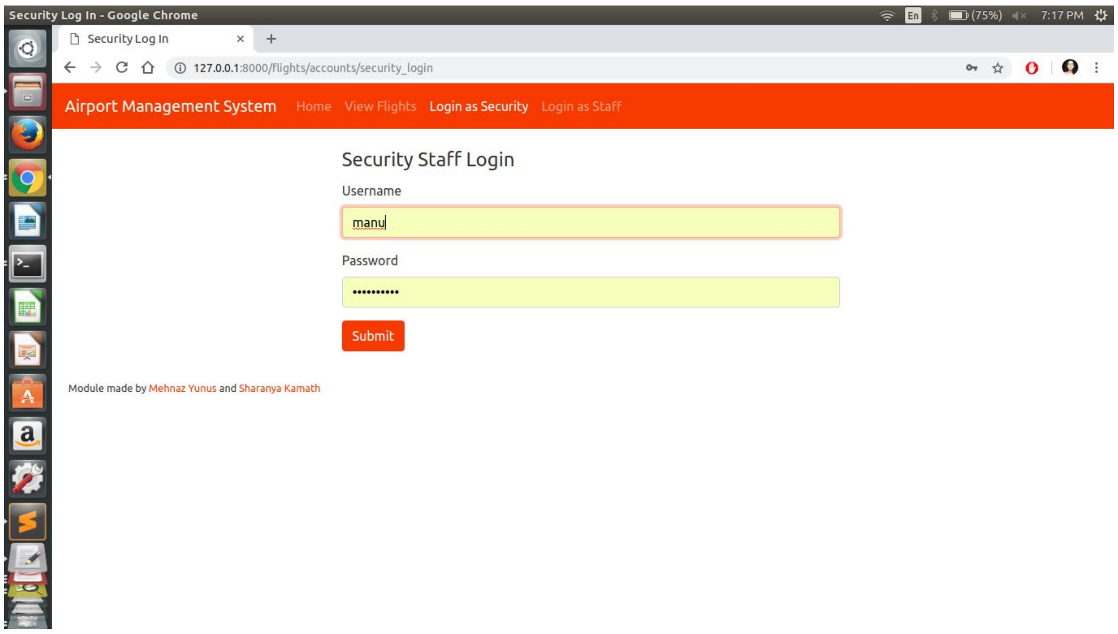
Confirmation

Are you sure you want to clear flight for take off?

Staff Report Generation - Google Chrome
127.0.0.1:8000/flights/staff_home/123
Yes, clear
Close

8. Security Login

Security Personnel can login to the system via this page.



Security Staff Login

Username: manu

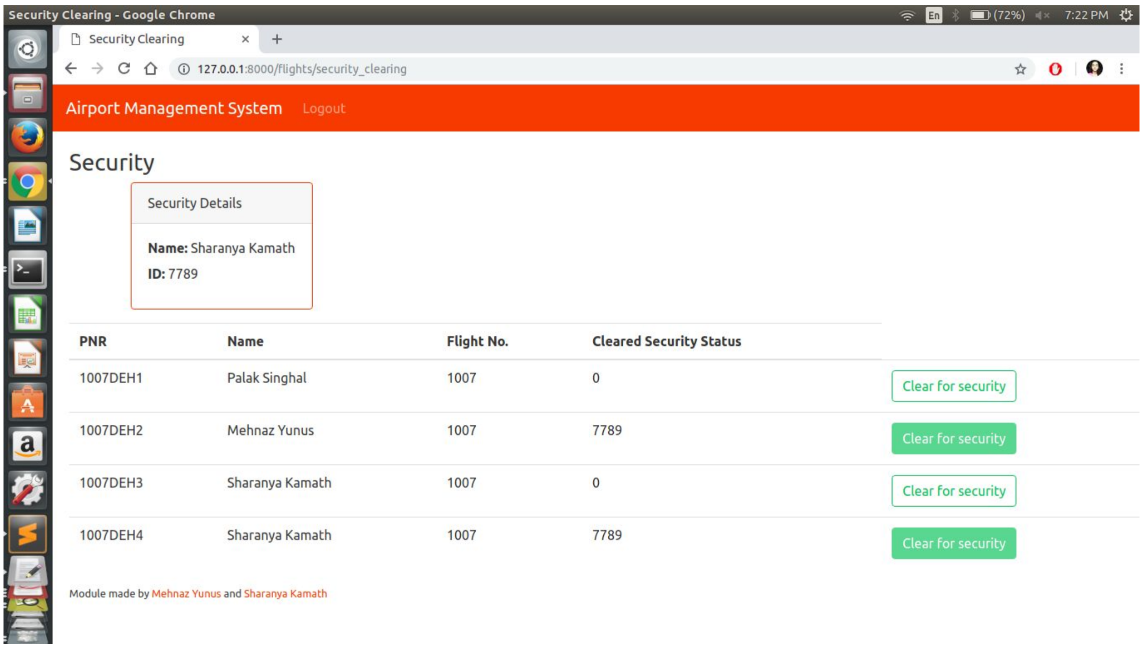
Password: ••••••••••

Module made by Mehnaz Yunus and Sharanya Kamath

Security Log In - Google Chrome
127.0.0.1:8000/flights/accounts/security_login
Airport Management System
Home
View Flights
Login as Security
Login as Staff
Submit

9. Security Home Page

The security personnel can view all the passengers and also clear them for security.


Security Clearing - Google Chrome
127.0.0.1:8000/flights/security_clearing
Airport Management System
Logout

Security

Security Details

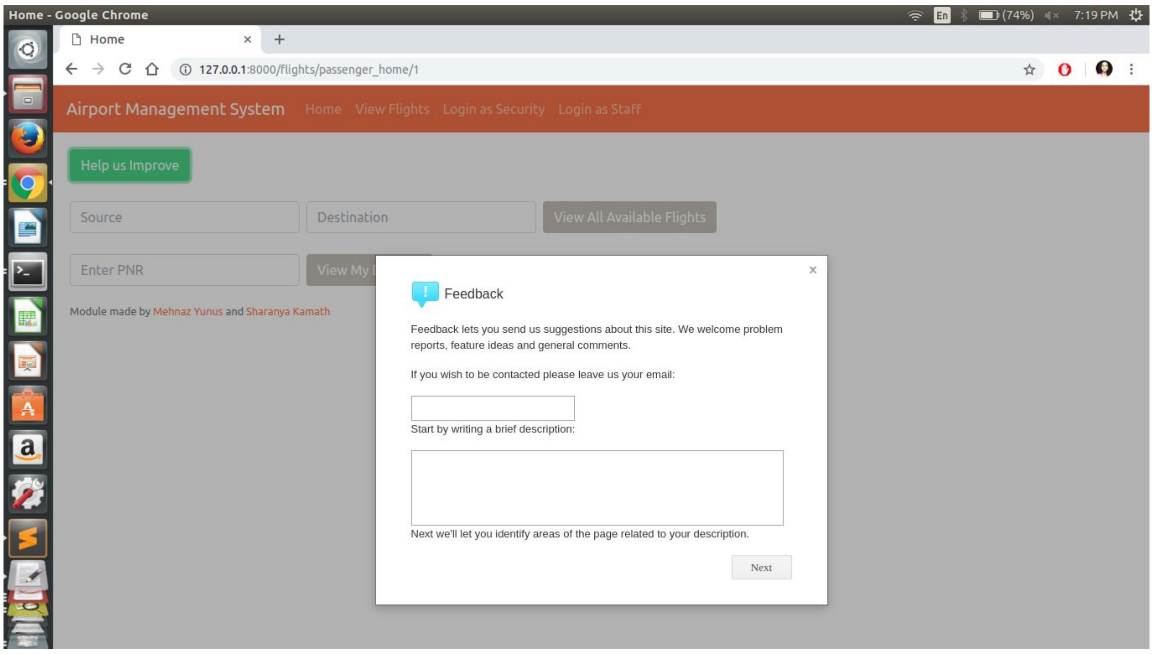
Name: Sharanya Kamath
ID: 7789

PNR	Name	Flight No.	Cleared Security Status	
1007DEH1	Palak Singhal	1007	0	Clear for security
1007DEH2	Mehnaz Yunus	1007	7789	Clear for security
1007DEH3	Sharanya Kamath	1007	0	Clear for security
1007DEH4	Sharanya Kamath	1007	7789	Clear for security

Module made by Mehnaz Yunus and Sharanya Kamath

10. Feedback

The user can provide feedback about the System.



Module made by Mehnaz Yunus and Sharanya Kamath

Feedback

Feedback lets you send us suggestions about this site. We welcome problem reports, feature ideas and general comments.

If you wish to be contacted please leave us your email:

Start by writing a brief description:

Next we'll let you identify areas of the page related to your description.

CONCLUSION

Since this project has been designed exclusively as a project, certain complexities that are faced by any real life manual problem like integrating actual payment mechanisms where users can pay through credit, debit card and some other features which we did not think about while creating the database. However this project can be enhanced whenever desired and we plan to continue working on the project.