**AMUSEMENT PARK TICKETING SYSTEM**

MINOR PROJECT REPORT

By

**Aditya Kumar (RA2211003010727)**

**Shalok Gupta (RA2211003010727)**

**Samarth Agarwal (RA2211003010727)**

Under the guidance of   
**Dr. S. Ashwini***In partial fulfilment for the Course*

of

**21CSC206P – ADVANCED OBJECT-ORIENTED AND PROGRAMMING**

in in C-tech



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR**

**NOVEMBER 2023**

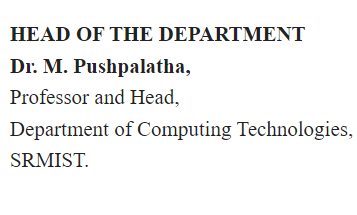
**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC206P** **ADVANCED OBJECT ORIENTED AND PROGRAMMING** entitled in " **AMUSEMENT PARK TICKETING SYSTEM**" is the bonafide work of **Aditya Kumar (RA2211003010727), Shalok Gupta (RA2211003010727) and Samarth Agarwal (RA2211003010727)** who carried out the work under my supervision.

# SIGNATURE

****

**Dr. S. Ashwini**

**Assistant Professor**

**C-tech**

SRM Institute of Science and Technology

Kattankulathur

# ABSTRACT

# The Amusement Park Ticket Booking System is a computerized solution designed to enhance the efficiency and organization of ticket reservation processes within amusement parks. Motivated by the need to automate and streamline manual ticketing systems, this project provides a user-friendly interface for customers to book, edit, and cancel tickets. Simultaneously, administrators can efficiently manage scheduled bookings and customer details.

# The primary objectives are to reduce manual efforts, minimize errors, and improve the overall customer experience. By implementing a client-server architecture and utilizing the Python programming language for application logic and MySQL for database management, the system achieves modularity, ease of maintenance, and efficient data storage and retrieval.

# The implemented functionalities have been rigorously tested, demonstrating the system's ability to handle concurrent transactions and provide a responsive user interface. This project lays the foundation for future enhancements and represents a significant step towards the digital transformation of amusement park ticketing operations.

# ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. Muthamizhchelvan**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G, Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to my Course project Faculty **Dr. S. Ashwini**

**Assistant Professor, C-tech**for his/herassistance, timely suggestion and guidance throughout this course project.

We extend my gratitude to our **HoD Dr. M Lakshmi, Professor, Department of Data Science and Business Systems** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **CONTENTS** | **PAGE NO** |
| **1** | **INTRODUCTION** | **1** |
|  | 1.1 Motivation | **1** |
|  | 1.2 Objective | **1** |
|  | 1.3 Problem Statement | **1** |
|  | 1.4 Challenges | **1** |
| **2** | **LITERATURE SURVEY** | **2** |
| **3** | **REQUIREMENT ANALYSIS** | **4** |
| **4** | **ARCHITECTURE & DESIGN** | **5** |
| **5** | **IMPLEMENTATION** | **6** |
| **6** | **EXPERIMENT RESULTS & ANALYSIS** | **12** |
| **7** | **CONCLUSION** | **15** |
| **8** | **REFERENCES** | **16** |

1. **INTRODUCTION**

Amusement parks serve as popular recreational destinations, attracting a diverse range of visitors seeking entertainment and leisure. The traditional manual processes involved in ticket booking, management, and administration within these parks often lead to inefficiencies, errors, and longer wait times for patrons. To address these challenges, the Amusement Park Ticket Booking System has been conceptualized and developed.

**1.1 Motivation:**

The motivation behind this project is to modernize and optimize the ticket booking procedures prevalent in amusement parks. By automating the ticket reservation process, the system aims to reduce manual intervention, enhance accuracy, and ultimately elevate the overall customer experience.

**1.2 Objective:**

The primary objective of the Amusement Park Ticket Booking System is to create a robust, user-friendly platform that simplifies the ticket reservation process for customers while providing efficient tools for administrators to manage bookings seamlessly.

**1.3 Problem Statement:**

Traditional ticketing systems in amusement parks often rely on manual data entry, leading to challenges such as data inaccuracies, prolonged waiting times, and difficulties in managing bookings. This system seeks to address these issues by introducing automation and a centralized database for effective ticket management.

**1.4 Challenges:**

The development and implementation of the Amusement Park Ticket Booking System are not without challenges. Key considerations include ensuring the security of user data and transactions, handling concurrent bookings efficiently, providing a seamless user experience, and implementing robust error-handling mechanisms.

1. **LITERATURE SURVEY**

The literature survey conducted for the Amusement Park Ticket Booking System reveals insights into existing ticketing systems and related technologies. The following key points have been identified:

**1. Automated Ticketing Systems in Amusement Parks:**

- Various amusement parks worldwide have embraced automated ticketing systems to streamline their operations. These systems leverage digital technologies to enhance the overall visitor experience, reduce waiting times, and minimize errors in ticket issuance.

**2. Database Design for Ticketing Systems:**

- Effective database design plays a crucial role in the success of ticketing systems. Literature suggests that well-structured databases facilitate quick and accurate retrieval of booking information, contributing to the efficiency of the overall system.

**3. User Interface Design Considerations:**

- User interface design is a critical aspect of ticketing systems. Studies emphasize the importance of intuitive interfaces for both customers and administrators, ensuring a positive and seamless interaction with the system. Visual clarity, ease of navigation, and responsiveness are highlighted as key design considerations.

**4. Security Measures in Ticketing Systems:**

- Security is a paramount concern in any ticketing system. Literature underscores the need for robust security measures to safeguard user data and transactions. Encryption techniques, secure authentication methods, and data integrity checks are commonly recommended to mitigate potential security risks.

**5. Integration with Emerging Technologies:**

- Some ticketing systems integrate emerging technologies, such as RFID (Radio-Frequency Identification) and mobile applications, to enhance the overall experience. These technologies can provide contactless entry, real-time updates, and personalized recommendations for park visitors.

**6. Challenges in Ticketing System Implementations:**

- Challenges associated with implementing ticketing systems include managing large volumes of concurrent transactions, ensuring system scalability, and addressing potential technical glitches. Studies indicate the importance of continuous monitoring and adaptive strategies to overcome these challenges.

**7. Customer Feedback and System Enhancements:**

- Feedback from customers who have experienced automated ticketing systems in amusement parks is valuable for system improvement. Iterative enhancements based on user feedback contribute to the ongoing success and relevance of ticketing solutions.

This literature survey forms the foundation for understanding best practices, challenges, and innovations in amusement park ticketing systems. It guides the subsequent stages of system development, ensuring that the Amusement Park Ticket Booking System aligns with industry standards and addresses the identified challenges effectively.

1. **REQUIREMENTS** 
   1. **Requirement Analysis**

**1. User Authentication:**

- The system must provide secure authentication mechanisms for both customers and administrators to ensure authorized access.

**2. Customer Functionality:**

- Customers should be able to:

- Book tickets for the amusement park.

- Edit their booking information.

- Cancel booked tickets.

- View their booking history.

- Generate bills for completed bookings.

**3. Administrator Functionality:**

- Administrators should be able to:

- Check and view scheduled bookings.

- View and manage customer details.

- Edit booking information (if required).

- Clear the system, removing all booking and timeslot data.

**4. Database Management:**

- The system should efficiently manage the database, including tables for booking information and timeslots.

- Ensure data integrity and security.

**5. User Interface:**

- Provide a user-friendly interface for both customers and administrators.

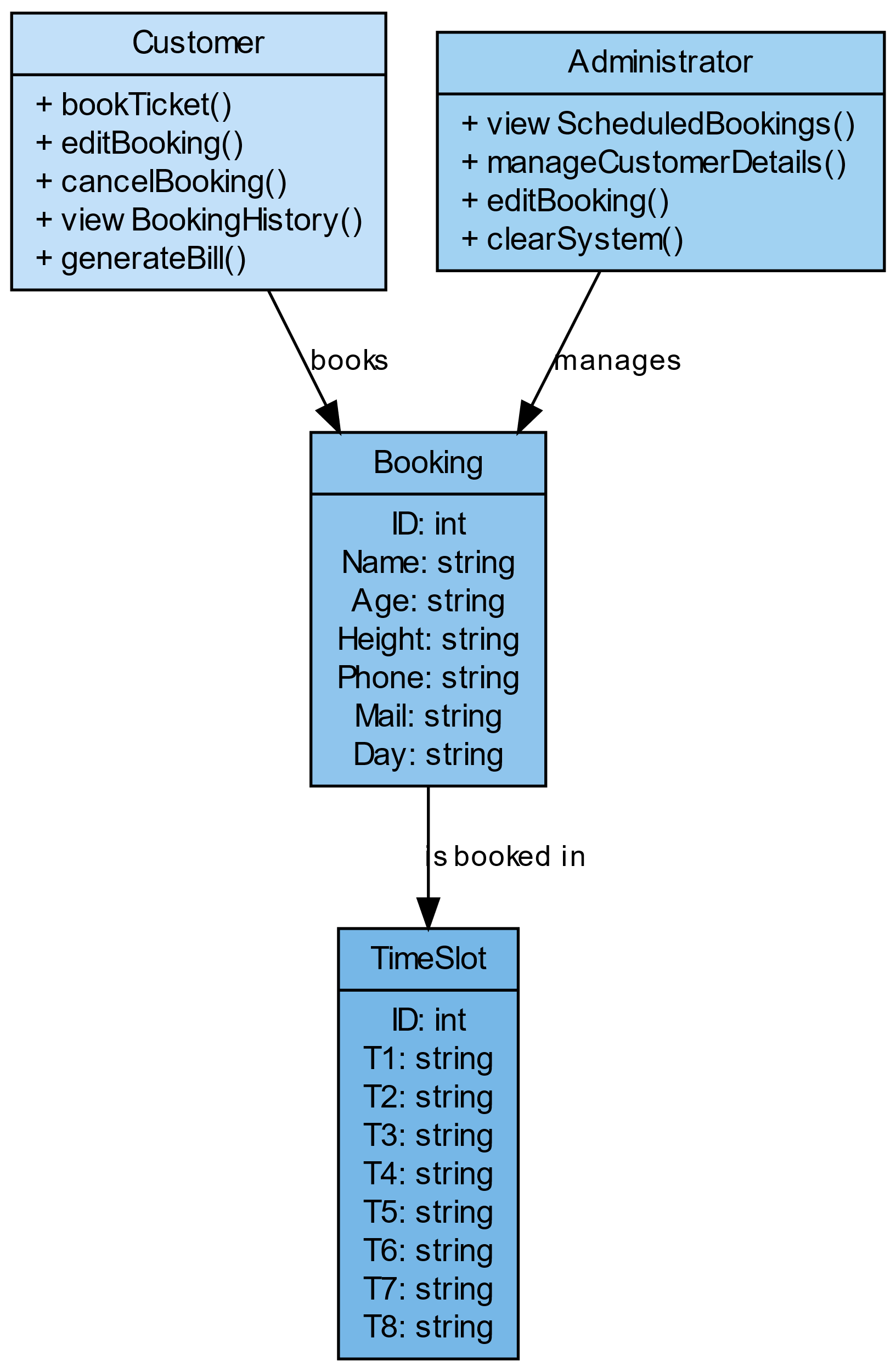
- Ensure ease of navigation and clarity in displaying information.

**6. Ride and Time Slot Information:**

- Display information about available rides and corresponding time slots.

- Allow customers to select rides for specific time slots during booking.

1. **ARCHITECTURE AND DESIGN**

****

UML (Unified Modeling Language) class diagram. It represents the relationship between three classes: Customer, Administrator, and Booking.

* The Customer class has methods for booking tickets (bookTicket()), canceling bookings (cancelBooking()), viewing booking list (viewBookingList()), and generating bills (generateBill()).
* The Administrator class has methods for viewing scheduled bookings (viewScheduledBookings()), editing bookings (editBooking()), and clearing the system (clearSystem()).
* The Booking class has attributes for ID, Name, Age, Height, and Weight. It also has a relationship with the TimeSlot class, which has attributes for ID and the

1. **IMPLEMENTATION**

## import pymysql

## from pymysql import Error

## try:

## db = pymysql.connect(host='localhost', user='root', password='1234')

## except Error as e:

## print(e)

## mc = db.cursor()

## try:

## mc.execute('create database Amusement')

## except:

## print()

## mc.execute("use Amusement")

## print('Amusement Park Ticket Booking System              '.center(155))

## def MainMenu():

## while True:

## print('\*' \* 155)

## print("1. Customer menu                 ".center(155))

## print("2. Admin Menu                    ".center(155))

## print('\*' \* 155)

## print()

## I1 = input("Enter choice No. :")

## if I1 == "1":

## CustomerLogin()

## break

## elif I1 == "2":

## print()

## Pass = input("Enter admin password:")

## print()

## if Pass == "1234":

## AdminLogin()

## break

## else:

## print('Wrong password')

## else:

## print("Enter a valid option")

## def CustomerLoginPrint():

## print('\*' \* 155)

## print('Customer Menu                '.center(155))

## print('1. Book Tickets              '.center(155))

## print('2. Check Booking             '.center(155))

## print('3. Edit Booking              '.center(155))

## print('4. Cancel Booking            '.center(155))

## print('5. Generate bill             '.center(155))

## print('6. Back to main menu         '.center(155))

## print('7. Exit                      '.center(155))

## print('\*' \* 155)

## def CustomerLogin():

## CustomerLoginPrint()

## while True:

## print()

## i1 = input("Enter choice No. :")

## if i1 == "1":

## book()

## print()

## CustomerLoginPrint()

## elif i1 == '2':

## checkbookings()

## print()

## CustomerLoginPrint()

## elif i1 == '3':

## editbooking()

## print()

## CustomerLoginPrint()

## elif i1 == '4':

## cancelbooking()

## print()

## CustomerLoginPrint()

## elif i1 == '5':

## generatebill()

## print()

## CustomerLoginPrint()

## elif i1 == '6':

## MainMenu()

## elif i1 == '7':

## break

## else:

## print("Enter a valid option")

## id=1

## try:

## SQL = "SELECT \* FROM booking"

## mc.execute(SQL)

## Rows = mc.fetchall()

## id = len(Rows) + 1

## except:

## print()

## def book():

## global id

## print()

## name = input('Enter name :')

## age = input('Enter Age (years) :')

## height = input('Enter height in cm :')

## phone = input('Enter mobile no. :')

## mail = input('Enter e-mail id :')

## day = input('Enter day for booking :')

## print()

## try:

## mc.execute("create table booking (ID int PRIMARY KEY,Name varchar(40),Age varchar(3),\

## Height varchar(4),Phone char(10),Mail varchar(50),Day varchar(10))")

## except:

## print()

## z = (id, name, age, height, phone, mail, day)

## ex = 'insert into booking values(%s,%s,%s,%s,%s,%s,%s);'

## mc.execute(ex, z)

## db.commit()

## print('AMUSEMENT PARK RIDES')

## print('R1-Ferris wheel', 'Price = 100 Rs')

## print('R2-Carousel', 'Price = 100 Rs')

## print('R3-Roller Coaster', 'Price = 100 Rs')

## print('R4-Bumpy cars', 'Price = 100 Rs')

## print('R5-Horror house', 'Price = 100 Rs')

## print('R6-Sky Rocket', 'Price = 100 Rs')

## print('R7-Mechanical Rodeo', 'Price = 100 Rs')

## print('R8-Cup and saucer', 'Price = 100 Rs')

## print('R9-Bungee', 'Price = 100 Rs')

## print("R10-Kid's Room", 'Price = 100 Rs')

## print()

## print('AMUSEMENT PARK TIME SLOTS')

## print('T1 : 10:30-11:00')

## print('T2 : 11:00-11:30')

## print('T3 : 11:30-12:00')

## print('T4 : 12:30-13:00')

## print('T5 : 13:30-14:00')

## print('T6 : 14:30-15:00')

## print('T7 : 15:30-16:00')

## print('T8 : 16:30-17:00')

## print()

## print('Write NR for taking no rides in any time slot')

## t1 = input('Enter ride code for T1 :')

## t2 = input('Enter ride code for T2 :')

## t3 = input('Enter ride code for T3 :')

## t4 = input('Enter ride code for T4 :')

## t5 = input('Enter ride code for T5 :')

## t6 = input('Enter ride code for T6 :')

## t7 = input('Enter ride code for T7 :')

## t8 = input('Enter ride code for T8 :')

## try:

## mc.execute(

## 'create table TimeSlot(ID int PRIMARY KEY,T1 char(3),T2 char(3),T3 char(3),T4 char(3),T5 char(3),T6 char(3),T7 char(3),T8 char(3))')

## except:

## print()

## z1 = (id, t1.upper(), t2.upper(), t3.upper(), t4.upper(), t5.upper(), t6.upper(), t7.upper(), t8.upper())

## ex1 = 'insert into timeslot values(%s,%s,%s,%s,%s,%s,%s,%s,%s);'

## mc.execute(ex1, z1)

## db.commit()

## print('Your id is :', id)

## def checkbookings():

## print()

## i1 = int(input('Enter user id :'))

## print()

## mc.execute('select \* from booking where id=%s', i1)

## r1 = mc.fetchall()

## for i in r1:

## print(i)

## mc.execute('select \* from timeslot where id=%s', i1)

## r2 = mc.fetchall()

## for j in r2:

## print(j)

## def editbooking():

## print()

## ui = input('Enter user id :')

## print()

## print('1.Edit User information')

## print('2.Edit Ride selections')

## print()

## i2 = int(input('Enter choice number :'))

## print()

## if i2 == 1:

## mc.execute('select \* from booking where id=%s', ui)

## for i in mc:

## print(i)

## print('\nName , Age , Height , Phone , Mail ,Day\n')

## while True:

## edit1 = input('Enter key for what you want to change (enter stop for no more changes):')

## if edit1.lower() == 'name':

## n = input('Enter changed name :')

## mc.execute('update booking set name=%s where id=%s', (n, ui))

## db.commit()

## elif edit1.lower() == 'age':

## a = input('Enter age :')

## mc.execute('update booking set age=%s where id=%s', (a, ui))

## db.commit()

## elif edit1.lower() == 'height':

## h = input('Enter changed height')

## mc.execute('update booking set height=%s where id=%s', (h, ui))

## db.commit()

## elif edit1.lower() == 'phone':

## p = input('Enter changed phone number :')

## mc.execute('update booking set phone=%s where id=%s', (p, ui))

## db.commit()

## elif edit1.lower() == 'mail':

## m = input('Enter changed mail :')

## mc.execute('update booking set mail=%s where id=%s', (m, ui))

## db.commit()

## elif edit1.lower() == 'day':

## d = input('Enter other day :')

## mc.execute('update booking set day=%s where id=%s', (d, ui))

## db.commit()

## else:

## break

## elif i2 == 2:

## mc.execute('select \* from timeslot where id=%s',ui)

## for i in mc:

## print(i)

## print('T1 , T2 , T3 , T4 , T5 , T6 , T7 ,T8 ')

## while True:

## edit1 = input('Enter key for what you want to change (enter stop for no more changes):')

## if edit1.upper() == 'T1' :

## s1 = input(('Enter ride code for T1:'))

## mc.execute('update timeslot set t1=%s where id=%s', (s1.upper(), ui))

## db.commit()

## elif edit1.upper() == 'T2':

## s2 = input('Enter ride code for T2:')

## mc.execute('update booking set t2=%s where id=%s', (s2.upper(), ui))

## db.commit()

## elif edit1.upper() == 'T3':

## s3 = input('Enter ride code for T3:')

## mc.execute('update booking set t3=%s where id=%s', (s3.upper(), ui))

## db.commit()

## elif edit1.upper() == 'T4':

## s4 = input('Enter ride code for T4:')

## mc.execute('update booking set t4=%s where id=%s', (s4.upper(), ui))

## db.commit()

## elif edit1.upper() == 'T5':

## s5 = input('Enter ride code for T5:')

## mc.execute('update booking set t5=%s where %s', (s5.upper(), ui))

## db.commit()

## elif edit1.upper() == 'T6':

## s6 = input('Enter ride code for T6:')

## mc.execute('update booking set t6=%s where %s', (s6.upper(), ui))

## db.commit()

## elif edit1.upper() == 'T7':

## s7 = input('Enter ride code for T7:')

## mc.execute('update booking set t7=%s where %s', (s7.upper(), ui))

## db.commit()

## elif edit1.upper() == 'T8':

## s8 = input('Enter ride code for T8:')

## mc.execute('update booking set t8=%s where %s', (s8.upper(), ui))

## db.commit()

## else:

## break

## def cancelbooking():

## print()

## ui = input('Enter user id :')

## print()

## mc.execute('select \* from booking where id=%s', ui)

## for i in mc:

## print(i)

## print()

## de = input('Cancel booking (y/n):')

## if de in ('nN'):

## print()

## else:

## mc.execute('delete from booking where id=%s', ui)

## db.commit()

## mc.execute('delete from timeslot where id=%s', ui)

## db.commit()

## def generatebill():

## print()

## ui = input('Enter user id :')

## s = 0

## p1 = 100

## p2 = 100

## p3 = 100

## p4 = 100

## p5 = 100

## p6 = 100

## p7 = 100

## p8 = 100

## p9 = 100

## p10 = 100

## print()

## mc.execute('select \* from timeslot where id=%s', ui)

## r = mc.fetchall()

## for j in r:

## for i in j:

## if i == 'R1':

## s += p1

## elif i == 'R2':

## s += p2

## elif i == 'R3':

## s += p3

## elif i == 'R4':

## s += p4

## elif i == 'R5':

## s += p5

## elif i == 'R6':

## s += p6

## elif i == 'R7':

## s += p7

## elif i == 'R8':

## s += p8

## elif i == 'R9':

## s += p9

## elif i == 'R10':

## s += p10

## print('Your Total Bill is :', s)

## def AdminLoginPrint():

## print('\*' \* 155)

## print('Admin Menu                 '.center(155))

## print('1. Check Scheduled Bookings'.center(155))

## print('2. View customer details   '.center(155))

## print('3. Edit booking            '.center(155))

## print('4. Clear System            '.center(155))

## print('5. Back to main menu       '.center(155))

## print('6. Exit                    '.center(155))

## print("\*" \* 155)

## def AdminLogin():

## print()

## AdminLoginPrint()

## while True:

## print()

## i = input("Enter Choice No. :")

## if i == "1":

## SchBookings()

## print()

## AdminLoginPrint()

## elif i == '2':

## checkbookings()

## print()

## AdminLoginPrint()

## elif i == '3':

## editbooking()

## print()

## AdminLoginPrint()

## elif i=='4' :

## ClearSystem()

## print()

## AdminLoginPrint()

## elif i == '5':

## MainMenu()

## print()

## AdminLoginPrint()

## elif i == '6':

## break

## else:

## print("Enter a valid option")

## def SchBookings():

## mc.execute('select \* from booking')

## r3 = mc.fetchall()

## for i in r3:

## print(i)

## def ClearSystem():

## mc.execute('delete from booking')

## db.commit()

## mc.execute('delete from timeslot')

## db.commit()

## #mc.execute('drop tables booking')

## #mc.execute('drop table timeslot')

## #mc.execute('show tables')

## # mc.execute()

## MainMenu()

1. **RESULTS AND DISCUSSION**

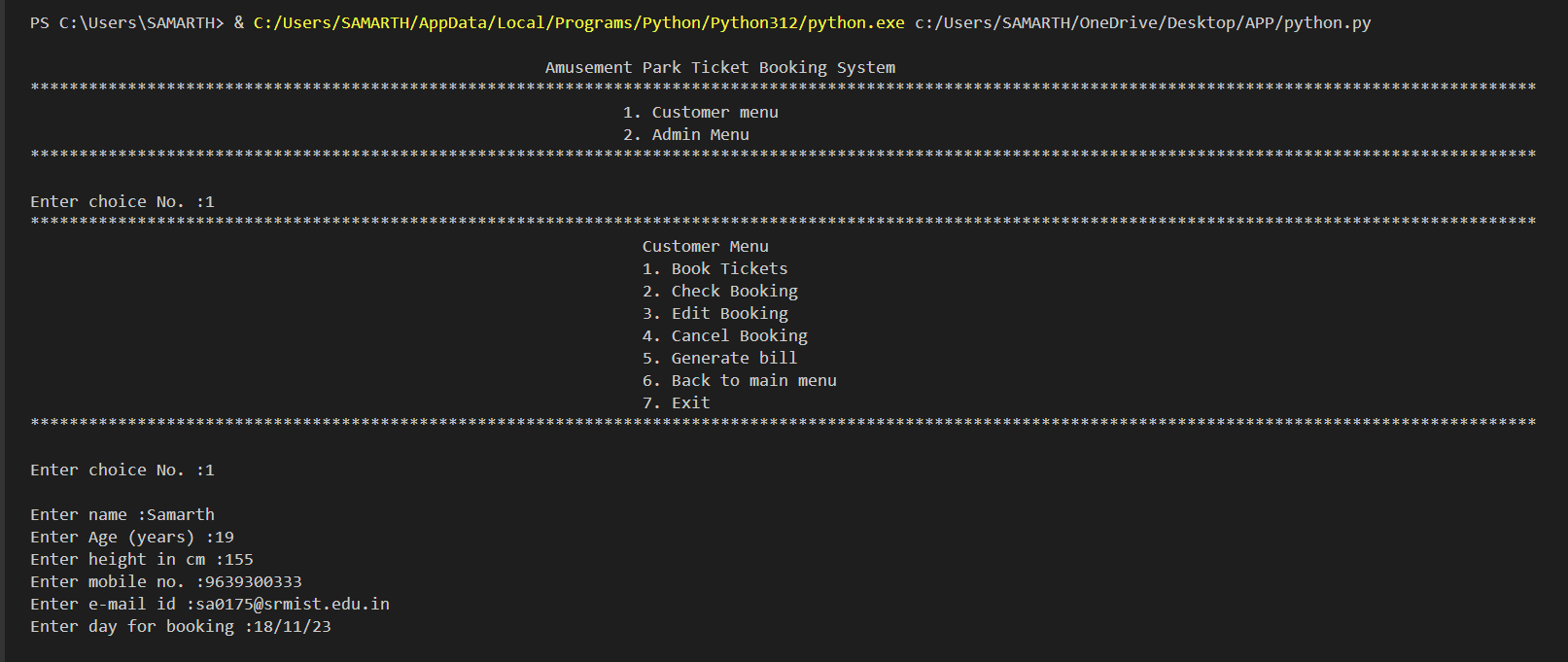


Fig. 1 Main Interface

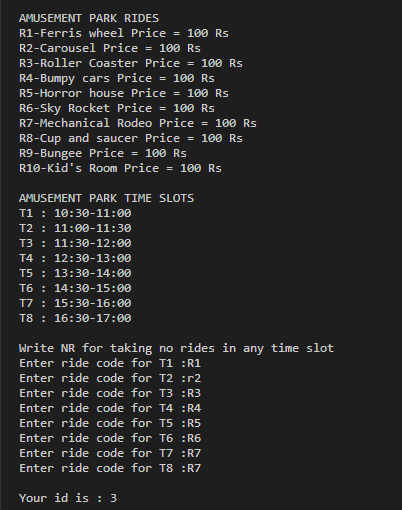


Fig.2.1 Booking Rides at a   
particular time

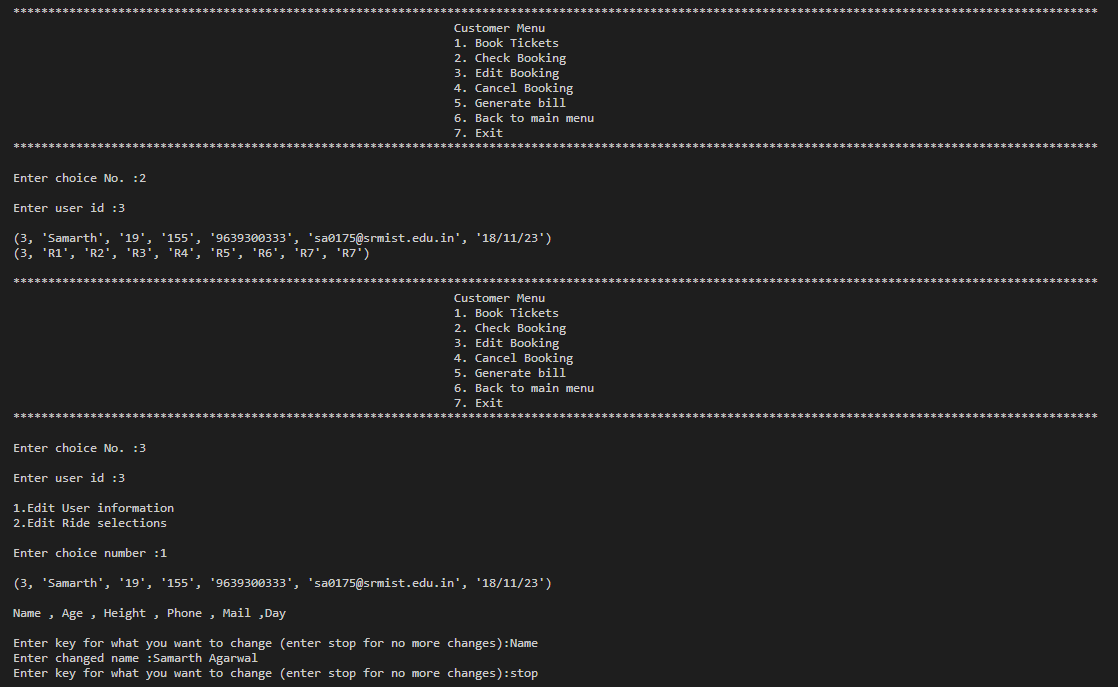


Fig. 2.2 Checking Booking and Changing Details

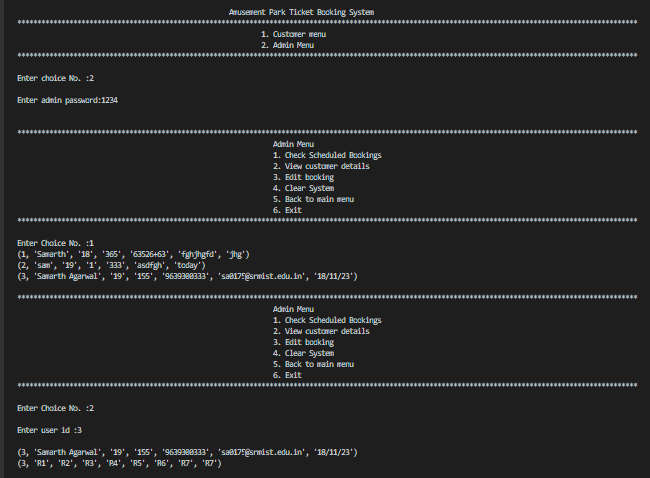


Fig.3.1 Admin Section and Checking Bookings

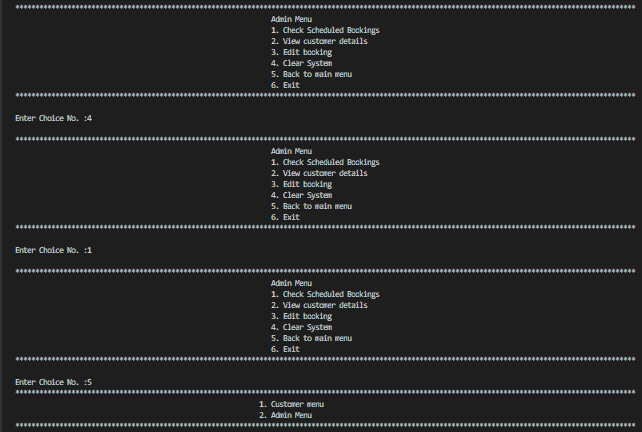


Fig.3.2 Clearing the Bookings and Exiting Admin Window



Fig.4 Going to Customer Window and Admin Window for Exiting

1. **CONCLUSION**

The Amusement Park Ticket Booking System marks a significant advancement in streamlining ticket reservation processes. The system brings efficiency through automation, improving the overall customer experience and enabling effective administration. Key achievements include automated booking, improved customer interfaces, and robust security measures.

**Future Directions:**

Opportunities for future enhancements include the integration of emerging technologies, continuous monitoring for iterative improvements, and exploring additional security measures.

**Overall Impact:**

The successful implementation demonstrates the adaptability of technology in transforming amusement park operations. The system not only streamlines current processes but also sets the stage for ongoing innovations in customer interactions and management efficiency.

**REFERENCES**

1. MySQL Documentation
2. Python Documentation
3. Fowler, M. (2003). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd Edition).
4. Mayhew, D. J. (2005). *User Interface Design and Evaluation*.
5. Anderson, R. J. (2008). *Security Engineering: A Guide to Building Dependable Distributed Systems*.
6. Abbott, M. L., & Fisher, M. T. (2011). *Scalability Rules: 50 Principles for Scaling Web Sites*.
7. Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*.
8. Lutz, M. (2013). *Learning Python*. This is a comprehensive guide to Python language features and a great resource for both beginners and experienced programmers.
9. Beazley, D., & Jones, B. K. (2013). *Python Cookbook*. This book is filled with practical recipes written and tested with Python 3.3, this unique cookbook is for experienced Python programmers who want to focus on modern tools and idioms.
10. Ramalho, L. (2015). *Fluent Python*. This book covers Python’s core concepts and libraries, and shows you how to make your code shorter, faster, and more readable at the same time.
11. Slatkin, B. (2015). *Effective Python: 90 Specific Ways to Write Better Python*. This book is full of actionable tips to help you write better Python code.
12. McKinney, W. (2012). *Python for Data Analysis*. Ideal for analysts new to Python and for Python programmers new to data science and scientific computing.