A project report on

ONE PIECE LOTTERY

Submitted in partial fulfilment of the requirement for the award of bachelor of computer application

Of

KANNUR UNIVERSITY

By

ARJUN SANTHOSH

ASHWANTH C

SHARATH V K

Concord Arts and Science College Muttannur

Kannur University

CONCORD ARTS AND SCIENCE COLLEGE

CONCORD EDUCITY, MUTTANNUR, (PO) PATTANNUR

ONE PIECE LOTTERY

PROJECT REPORT

2022-2023

BACHELOR OF COMPUTER APPLICATION

KANNUR UNIVERSITY

CONCORD ARTS AND SCIENCE COLLEGE

CONCORD EDUCITY, MUTTANNUR, (PO) PATTANNUR

BACHELOR OF COMPUTER APPLICATION

BONAFIDE CERTIFICATE

Certified that this project report "**ONE PIECE LOTTERY"** is the bona-fide work of **ASHWANTH C Reg.No:KI20BCAR21** who carried out the project work under my super vision certified further that to the best of my knowledge the work reported here does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Head of the Department Project Coordinator

External Examiner

Date:

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task will be complete without mentioning the people, Who made it possible, Whose constant guidance and encouragement crowns all efforts with success

At the very outset we would like to express a deep sense of gratitude to our Head of the Department Smt. Anusree IP for having laid tracks that led us towards a bright future and providing us all the information about the project

We also wish to express our sincere gratitude to Mr Kathiru Shanthi Kumar for his constant guidance and valuable advice for the completion of the project. We are also delighted in extending our thanks to our friends, family for their cooperation throughout the preparation and completion of our project.

TABLE OF CONTENT

ABSTRACT

1. ABSTRACT

ONE PIECE LOTTERY is a web application .

We all know that online lottery has become wide spread since its introduction because of its benefits it has to offer. While there are a ton of advantages. The most notable benefits it has over paper lottery is the credibility and transfer ability of playing the lottery online. The process of selecting winners is transparent for all the players involved, Which improves their trust and confidence and encourages them to participate in additional lottery games.

The internet has become a huge source of information and has made things easier for people in a variety of ways, including the lottery. It cannot only help in participating, But is also useful in number selection.

1.1 PURPOSE

ONE PIECE LOTTERY is a web application project which is handled by three module. One module is handled by admin. Second module is handled by Lottery. And third one is handled by User. This project deals with an area of that focus to buy the lottery online and also publishing the result.

1.2 SCOPE

The scope of an online lottery project can vary depending on the goals and objectives of the project. However, here are some common elements that may be included in the scope:

- Website Development: The development of an online platform where users can buy lottery tickets, view results, and claim their prizes.

- Lottery Management System: The development of a system that manages lottery draws, generates winning numbers, and ensures fairness and transparency.
- User Management System: The development of a system to manage user accounts, user profiles, and provide customer support.

1.3 OVERVIEW

The application consist of three modules. The first module is the Admin who has full authority over the application. Administrator who has the ability to control the website, those who can approve the separate lottery and block the fake lottery, and who also has the ability to see the result as same as that of the other two modules.

The second module Lottery, there are many lottery's. Each lottery has their username and password to login. Lottery can manage all the lottery tickets. Lottery can add, remove lottery at the specified time. Lottery can start the lottery draw, it provides the lottery prize, it generates the lottery user,and also publish the result and view the result as same as that of other two modules.

The third module is User. Each user has their username and password to login. Each user can see the lottery that they buy.The user who buy the lottery becomes the Lottery user that where generated by the lottery. The user has the ability to see the result of the lottery also which were not taken by the user.

SYSTEM ANALYSIS

2.SYSTEM ANALYSIS

System analysis is the process of gathering and interpreting facts, problems and using the information to recommend improvements of the system.

2.1 EXISTING SYSTEM

In the case of paper lottery tickets have to be collected from all the retailers and centralized. All tickets have to be checked to find the winnings and then the amount has to be paid back to the retailers. This is time consuming. Customers don't receive their price on time. If the lottery tickets is not maintained properly it can be lost, washed off etc..

All these problems are eliminated in the case of online lottery because everything is done online and the system is automated.

2.2 LIMITATIONS OF EXISTING SYSTEM

- Keeping track of the paper lottery is a difficult task
- Less efficiency
- Lack of security
- Difficult to maintain
- Time consuming
- Loss of trust

2.3 PROPOSED SYSTEM

Proposed system is a web and python based application. This system deals with an area of that focus to buy the lottery online and also publishing the result.

The lottery and the user must be registered by entering the required details. The user can see the lottery and buy the lottery. And publish the result. The result can be view by the User, Lottery and Admin.

2.4 ADVANTAGE OF THE PROPOSED SYSTEM

- Save on operational overheads
- No need to keep track

2.5 SOFTWARE REQUIREMENT SPECIFICATION

SRS forms the basics of software development. The SRS is the official statement of what is required of the system developers. It should include a detailed specification of the system requirement. Hence a high quality SRS is a prerequisite to high quality software. The project consists of three modules.

ADMIN:

1. Login
2. Manage Lottery (Approve, View,Block)
3. View Lottery
4. View Result
5. Log out

LOTTERY:

1. Login (Owner)
2. Manage Lottery(tickets)
3. Manage Lottery draws
4. Manage Lottery prize
5. Generate Lottery user
6. Publish Lottery Result
7. View Result
8. Logout

USER:

1. Login
2. View Lottery
3. Enter the UPI id
4. Buy Lottery
5. View the result

6. Logout

2.6 SYSTEM REQUIREMENTS

HARDWARE:-

- Processor : Pentium III or above
- Speed : 1.1 Ghz or above
- Hard Disk : 20 GB or above
- RAM : 4 GB or above
- Monitor : 15 VGA Color
- Keyboard : Standard Windows Keyboard

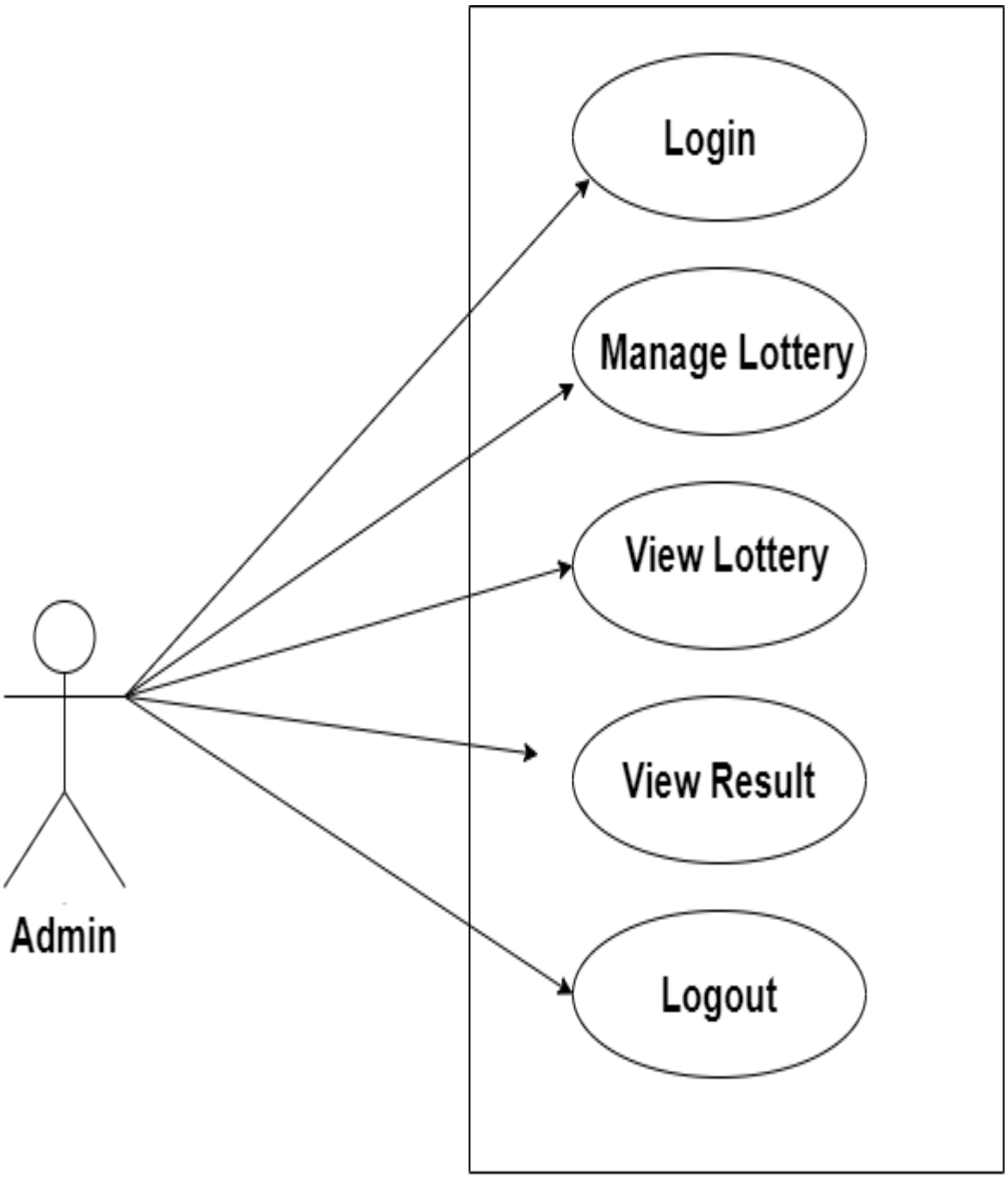
SOFTWARE:-

- Operating system : Windows 7 and above
- Web application: Python flask
- API : python : flask restful
- frontend: html, CSS, Meterializecss(CSS frame work)
- Database: Firebase Database
- Text Editor: Sublime text
- IDLE: python IDLE

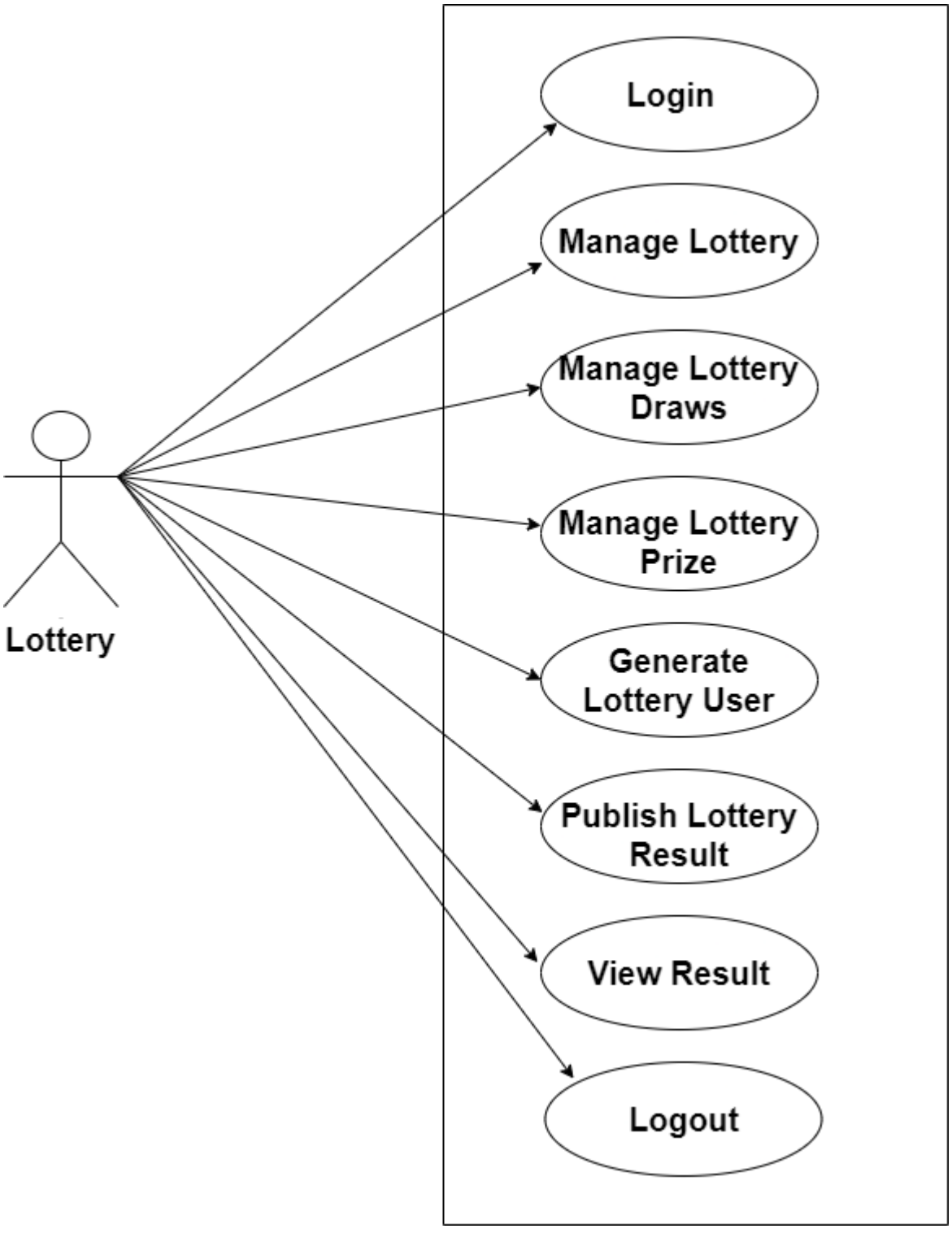
2.7 REQUIREMENT MODELING

Use Case Diagram

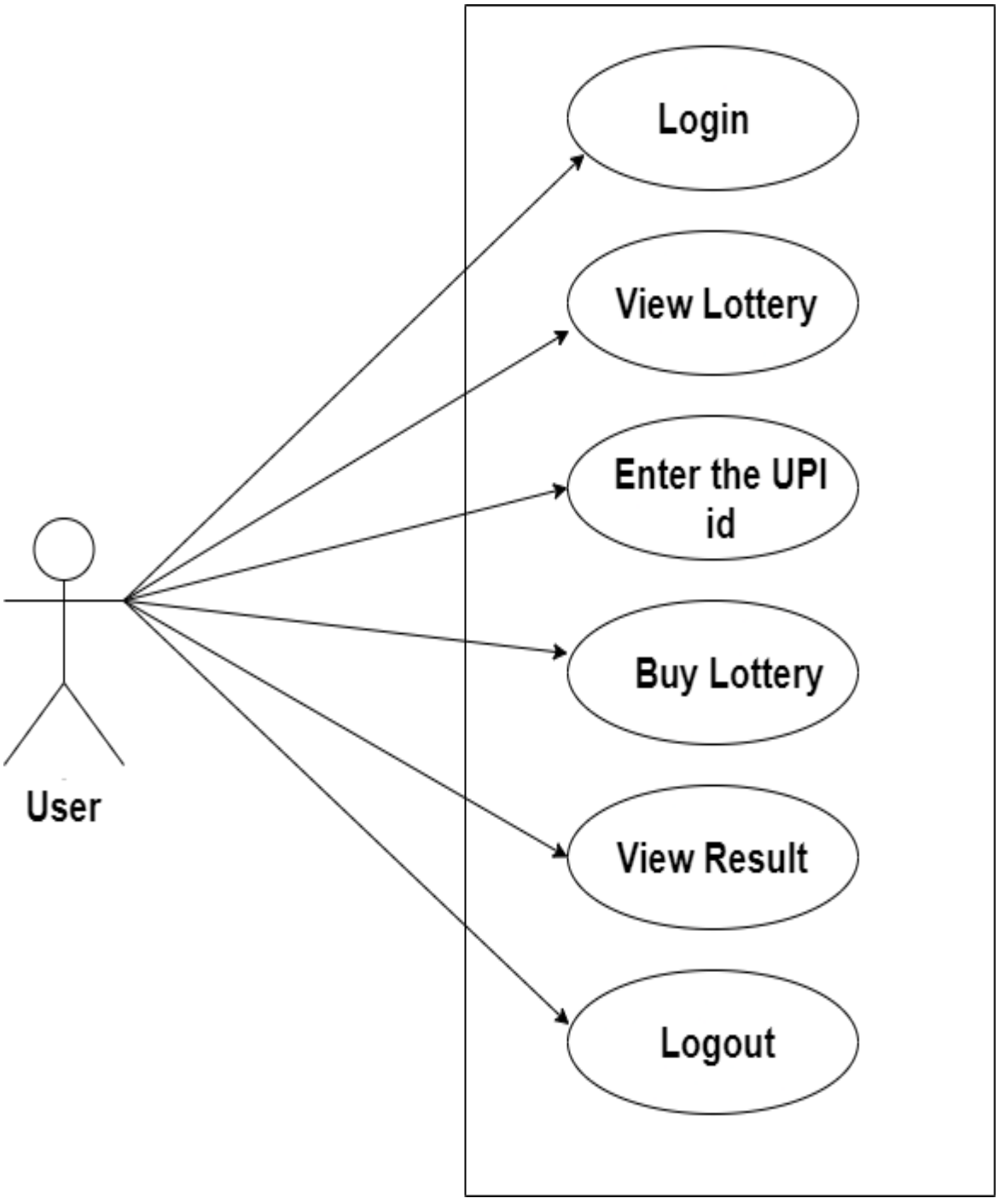
- Admin



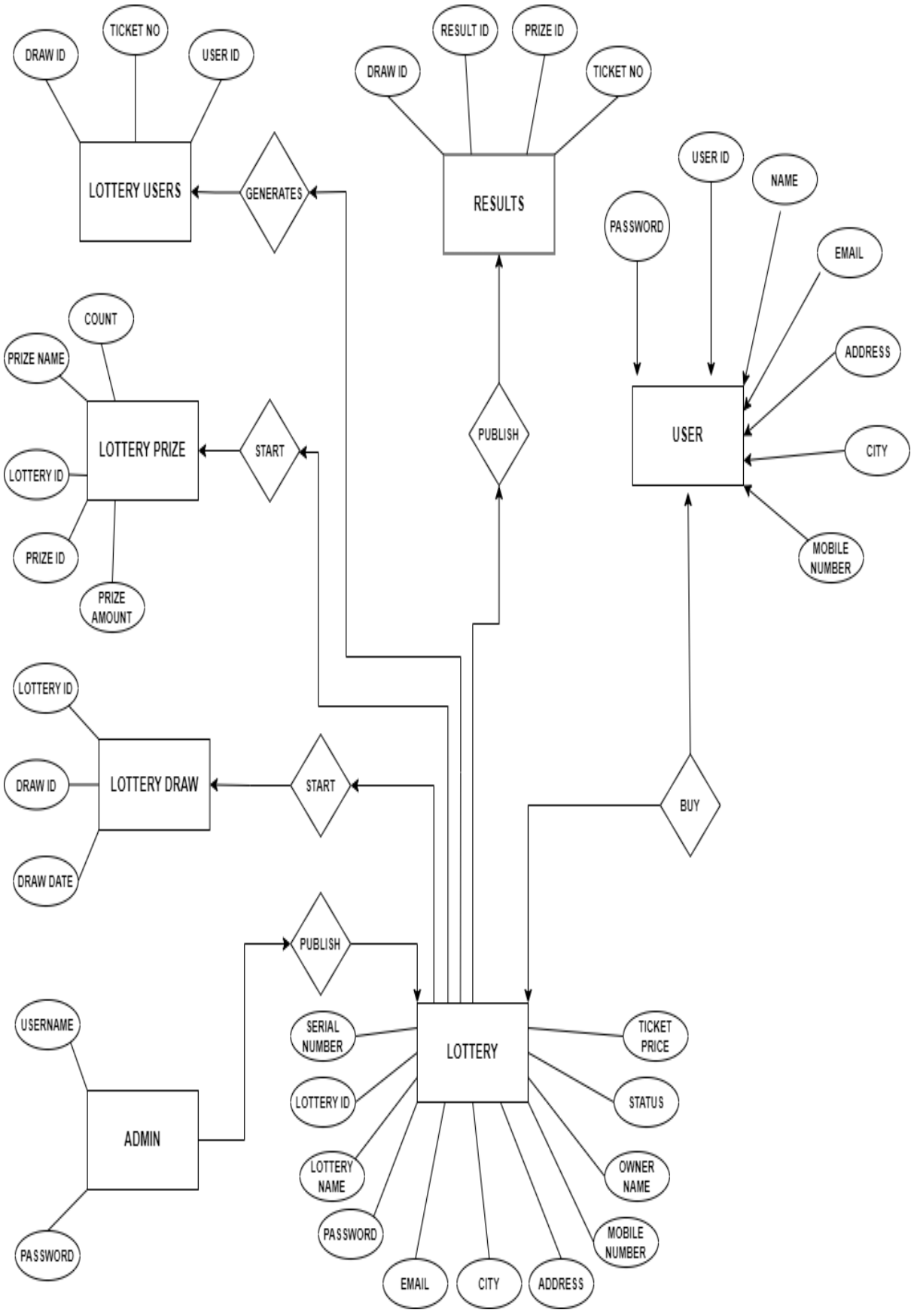
- Lottery


Login
Manage Lottery
Manage Lottery Draws
Manage Lottery Prize
Generate Lottery User
Publish Lottery Result
View Result
Logout
Lottery

- User

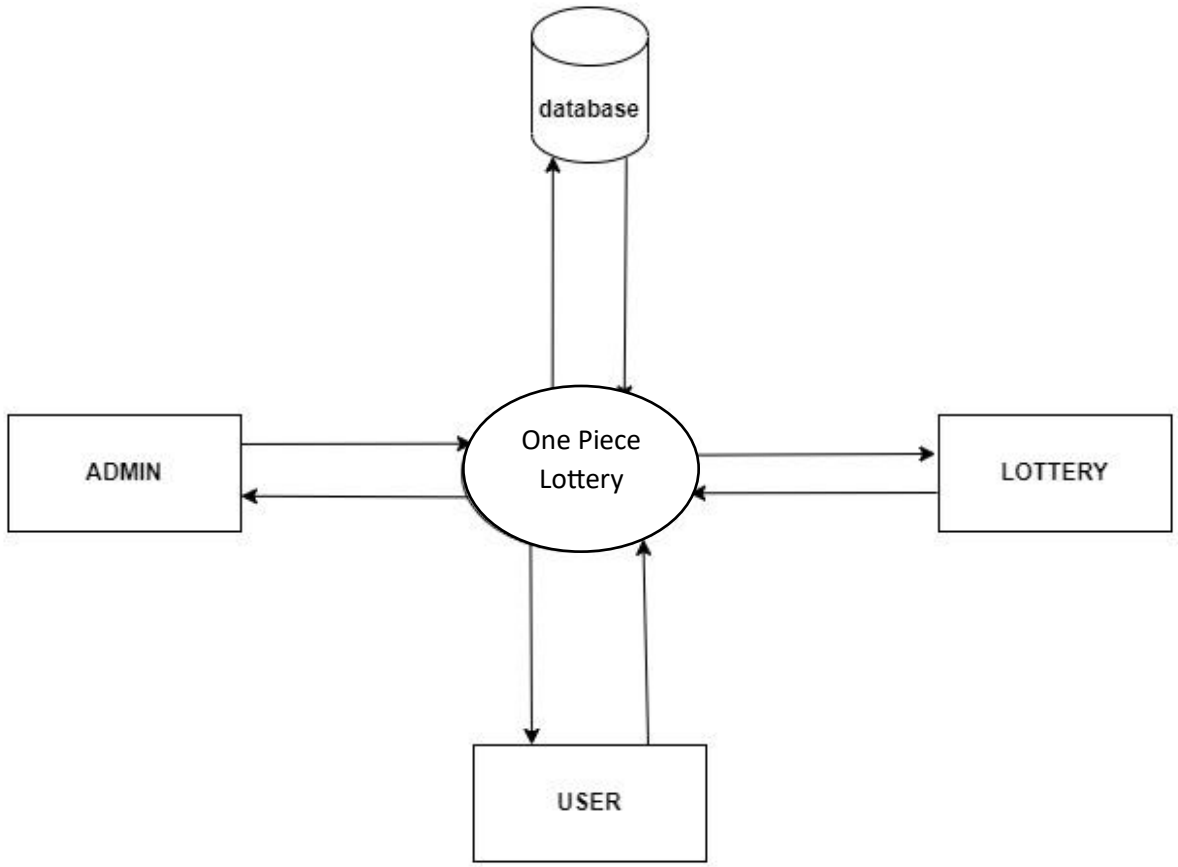

Login
View Lottery
Enter the UPI id
Buy Lottery
View Result
Logout
User

Entity Relationship Diagram

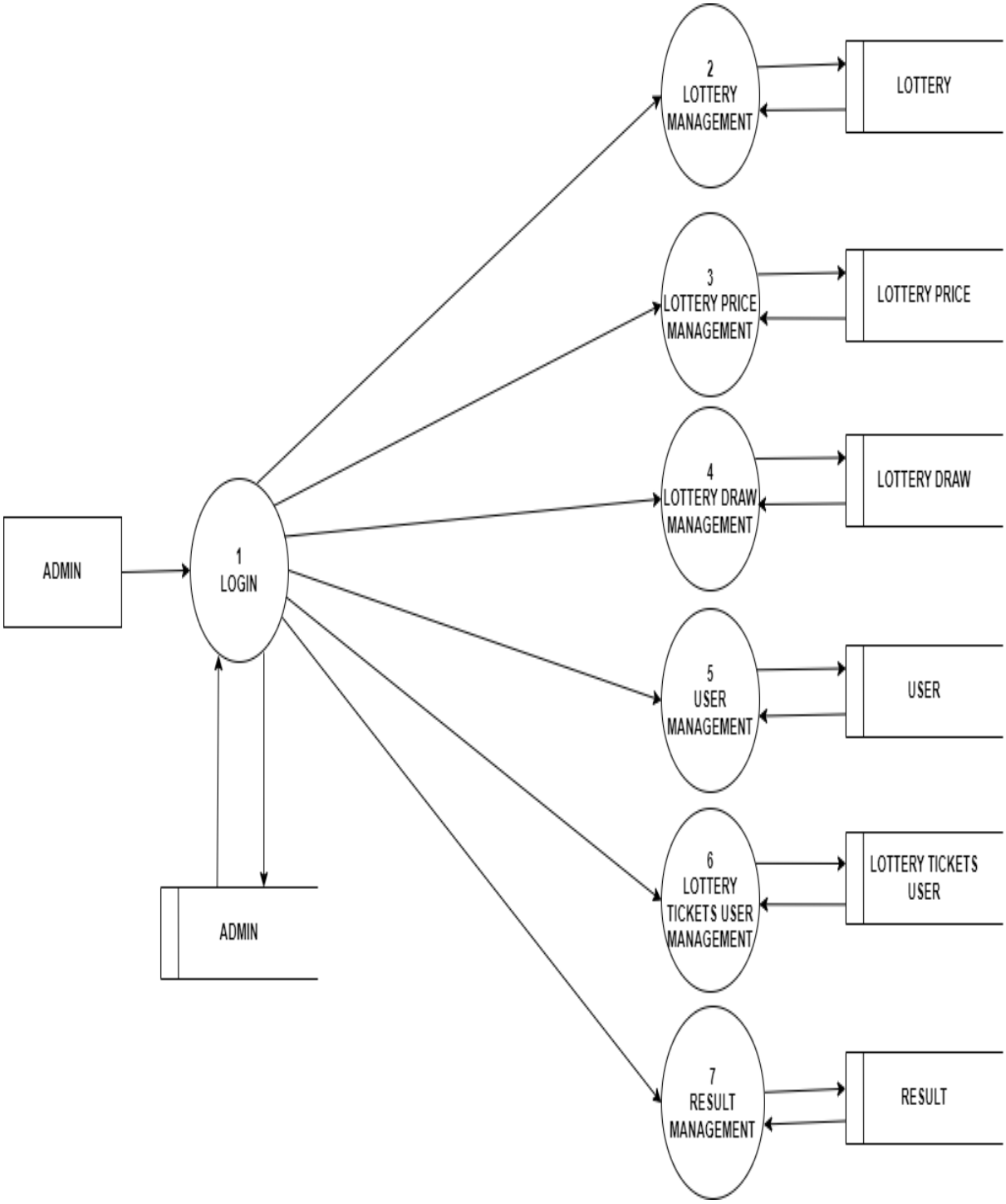


Data Flow Diagram

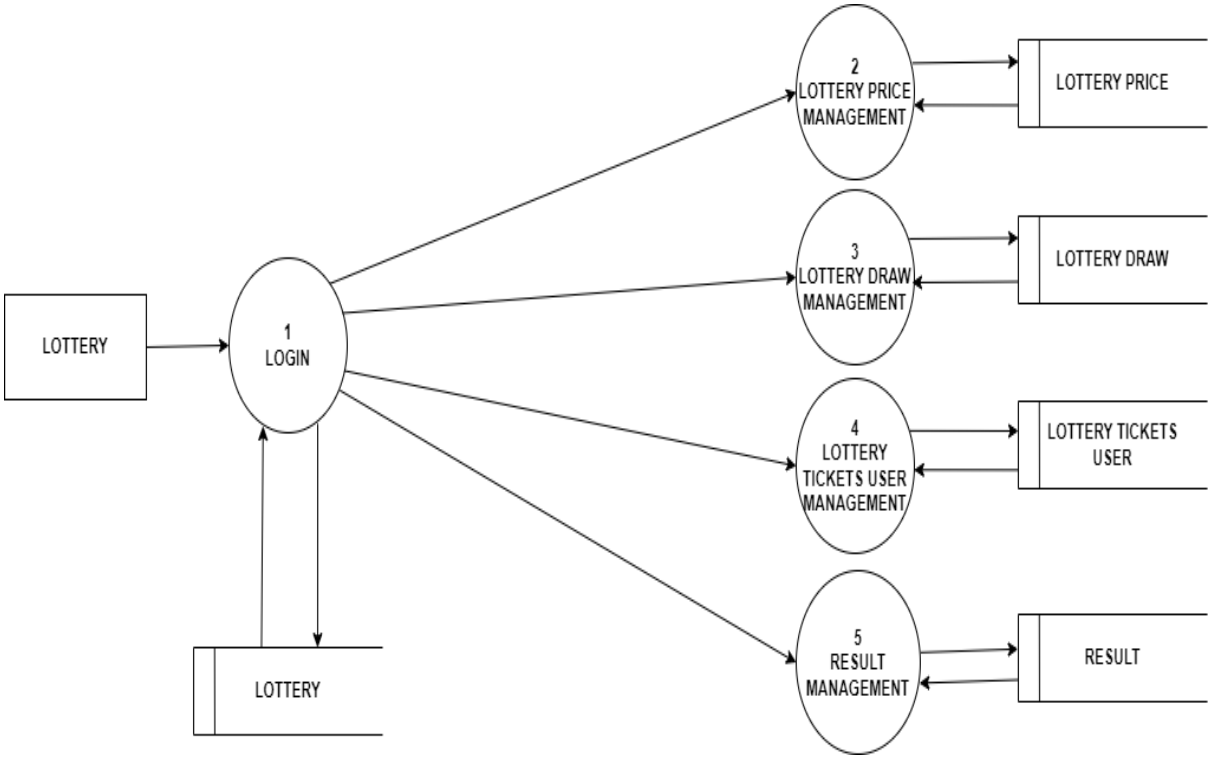
LEVEL 0



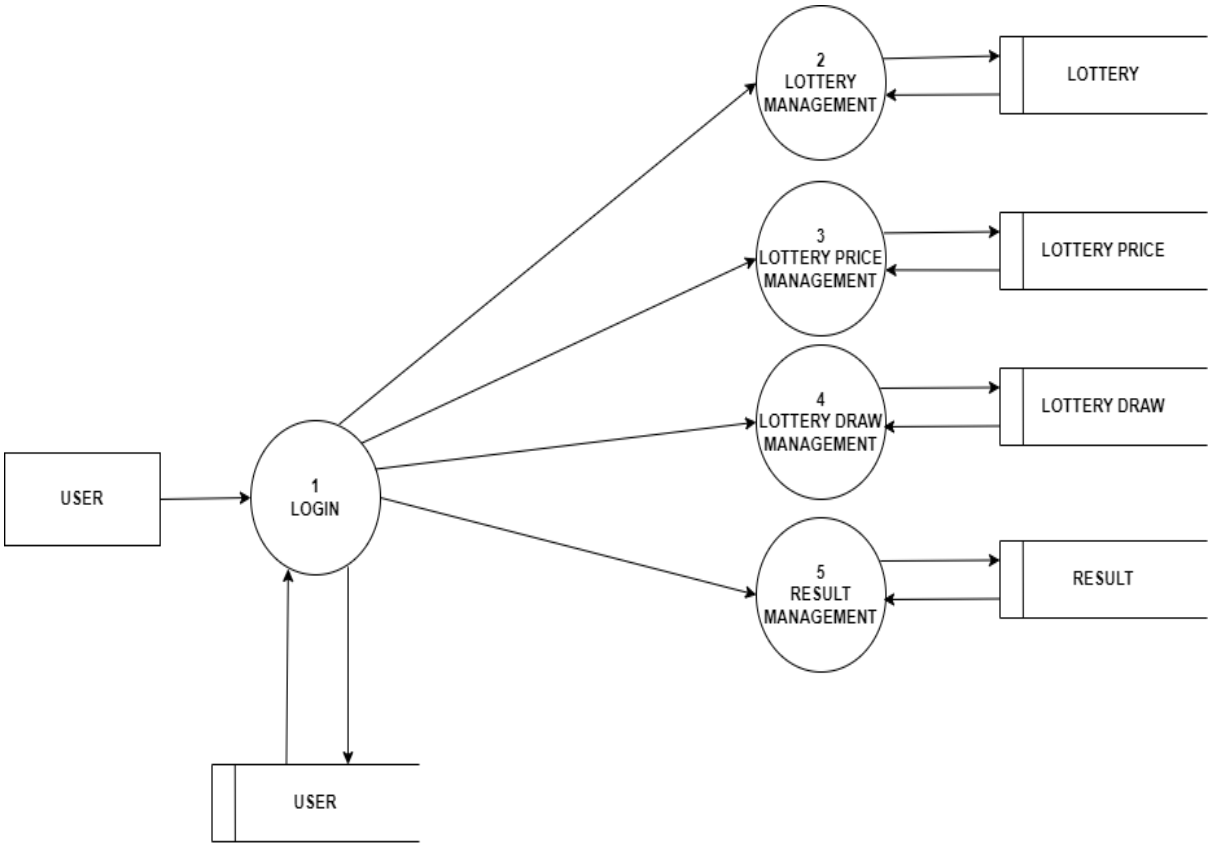
LEVEL 1 ADMIN



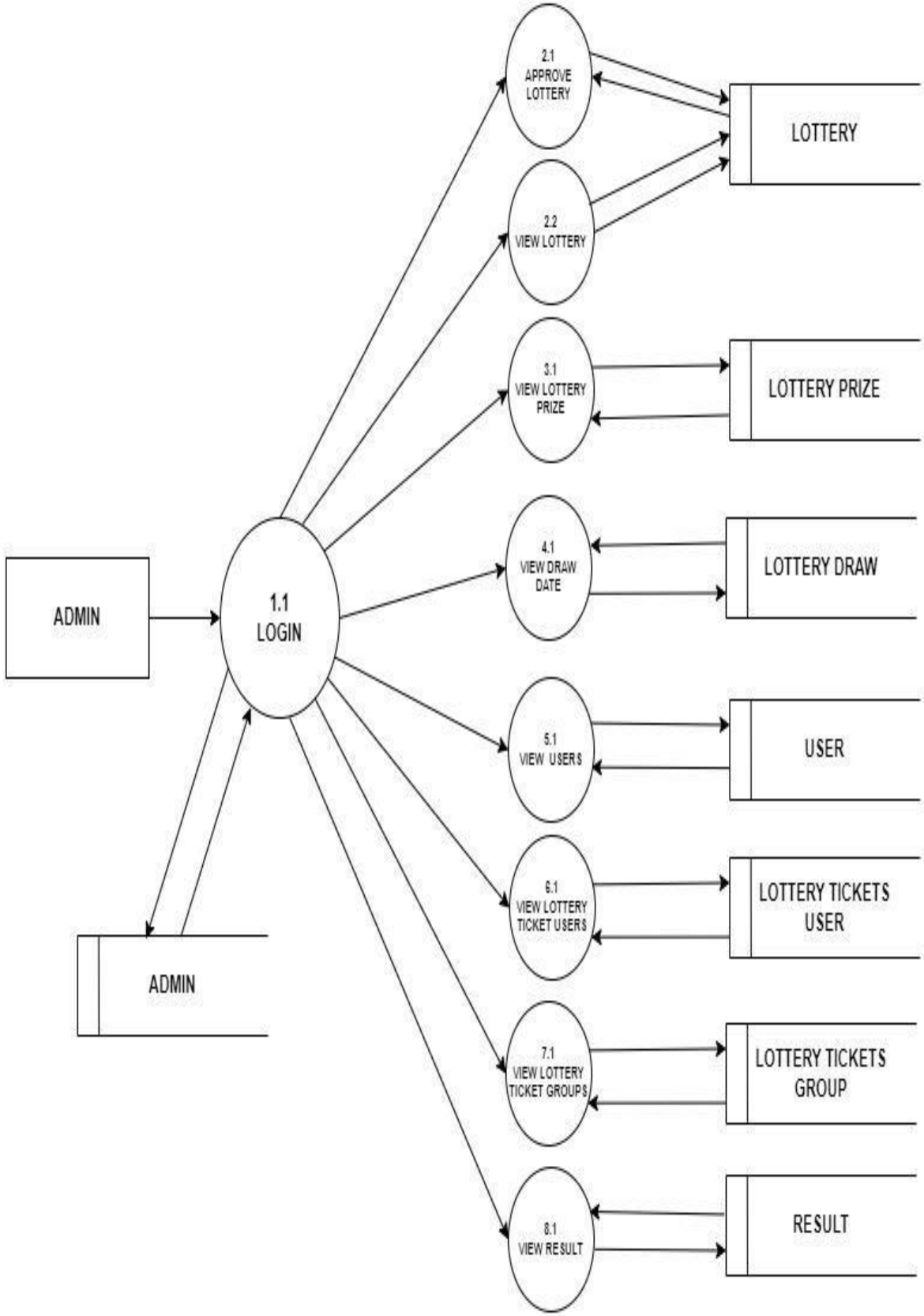
LEVEL 1 LOTTERY



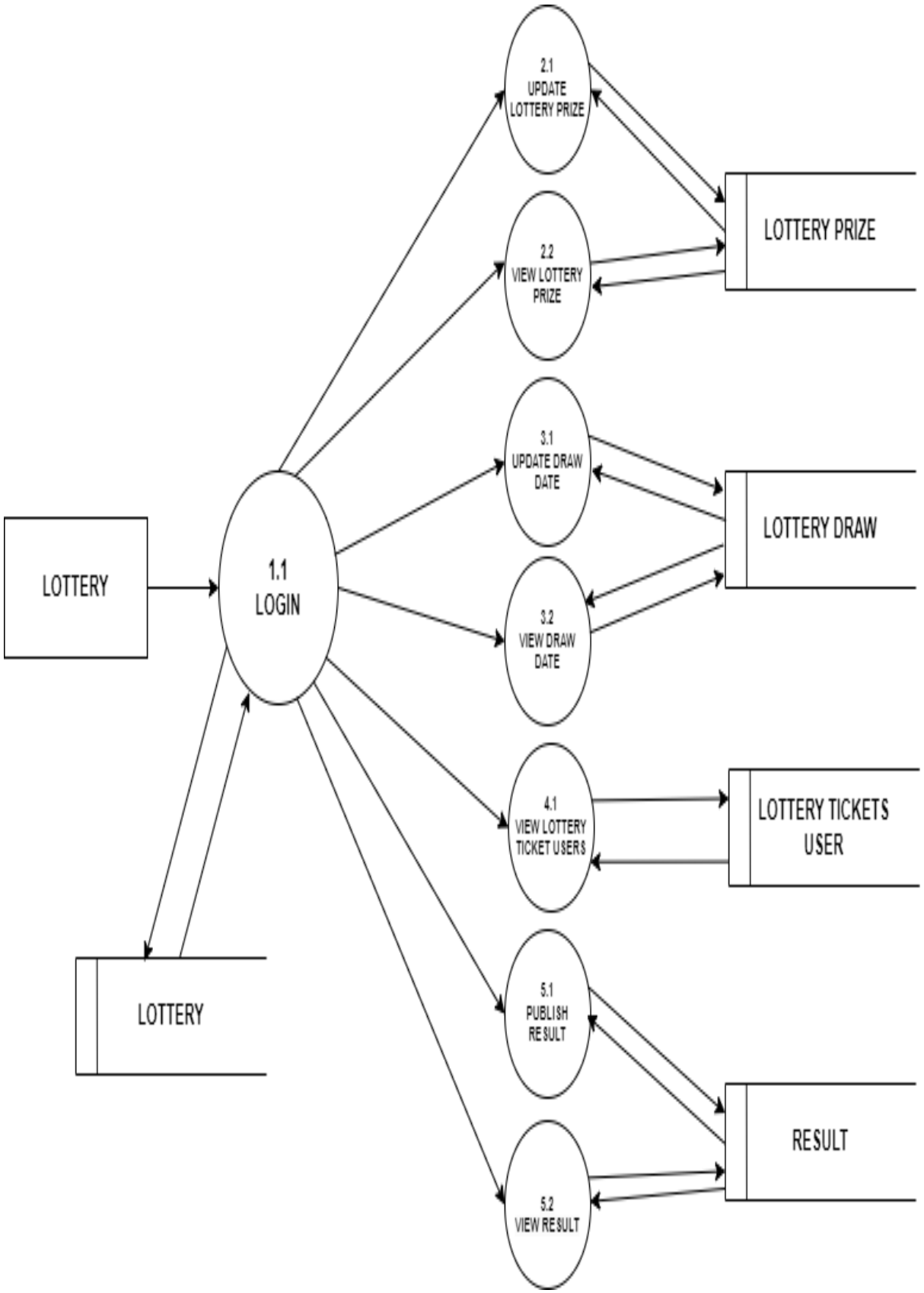
LEVEL 1 USER



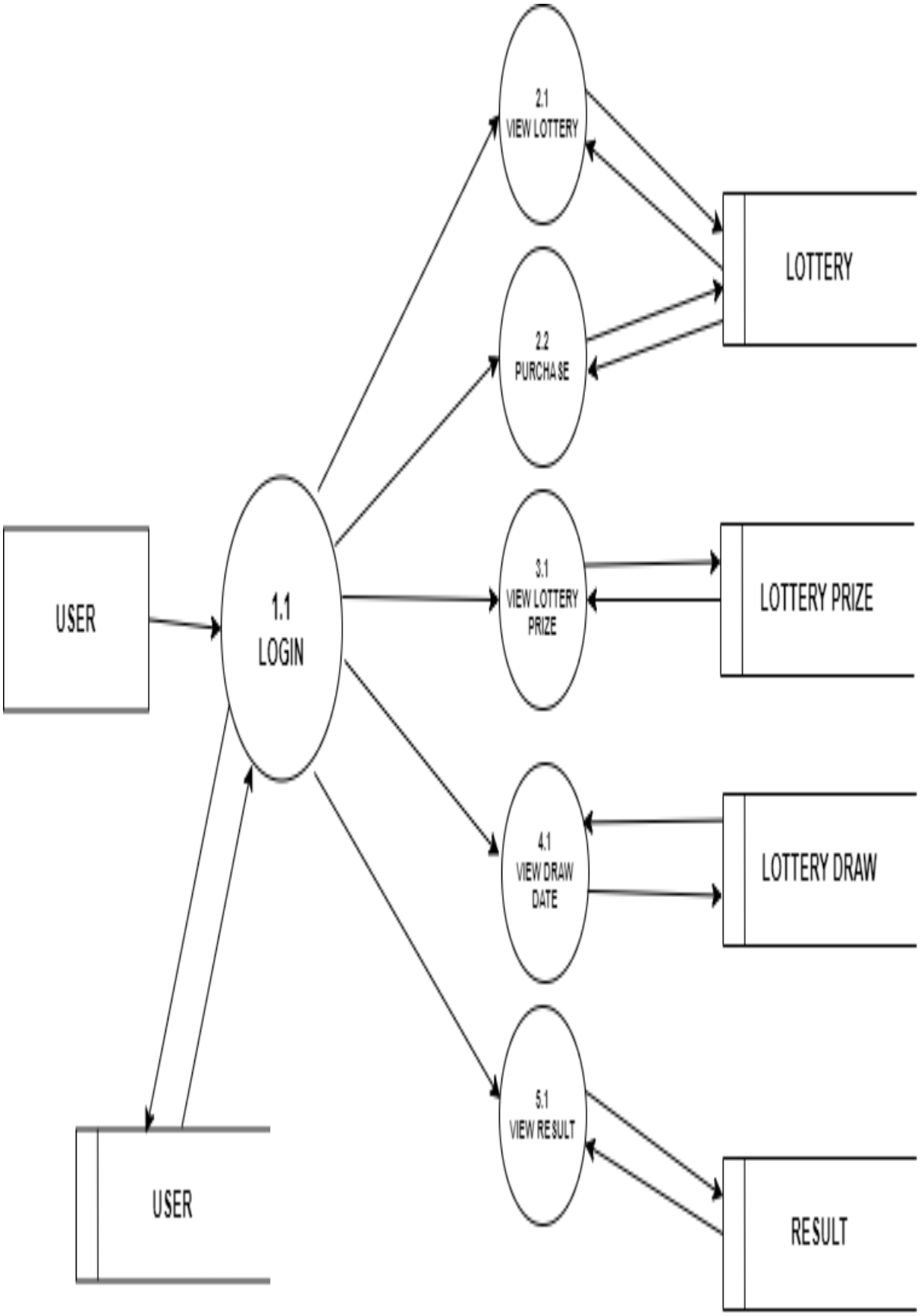
LEVEL 2 ADMIN



LEVEL 2 LOTTERY



LEVEL 2 USER


2.1
VIEW LOTTERY
LOTTERY
2.2
PURCHASE
USER
1.1
LOGIN
3.1
VIEW LOTTERY
PRIZE
LOTTERY PRIZE
4.1
VIEW DRAW
DATE
LOTTERY DRAW
5.1
VIEW RESULT
USER
RESULT

2.8 FEASIBILITY STUDY

A feasibility study for an online lottery would typically include the following key components:

Market Analysis: This would involve researching the market demand for online lotteries, as well as analyzing the competition and identifying potential target markets.

Technical Requirements: This would involve identifying the hardware, software, and other technical infrastructure needed to support an online lottery platform.

Legal and Regulatory Compliance: This would involve researching and complying with the legal and regulatory requirements for online lotteries in the targeted market(s).

Financial Feasibility: This would involve analyzing the revenue potential and cost structure of the online lottery platform, including the costs of developing and maintaining the platform, as well as the potential revenue from ticket sales and other sources.

Operational Feasibility: This would involve assessing the feasibility of running an online lottery platform, including identifying the necessary personnel, resources, and processes needed to operate the platform effectively.

Risk Analysis: This would involve identifying and evaluating potential risks associated with running an online lottery platform, such as regulatory compliance, security risks, and financial risks.

Overall, a feasibility study for an online lottery would require careful research and analysis of the market demand, technical requirements, legal and regulatory compliance, financial feasibility, operational feasibility, and potential risks associated with operating an online lottery platform.

3.SYSTEM DESIGN

3.SYSTEM DESIGN

Designing an online lottery system can be a complex process that involves multiple components and considerations. Here's an overview of some of the key elements that you'll need to consider when designing an online lottery system:

Ticket purchasing: Users should be able to purchase tickets online through your platform. This will require integration with payment gateways for secure transactions, and you'll need to implement a system to generate and manage tickets.

Random number generation: Since lotteries are based on random numbers, you'll need to implement a secure and reliable system for generating these numbers. You'll also need to ensure that the system is fair and free of any bias or manipulation.

Ticket validation: When users purchase tickets, they'll need a way to check if they've won. This requires a system for validating the winning numbers and matching them against purchased tickets.

Prize distribution: Once winners have been identified, you'll need to distribute prizes. This will involve managing prize pools and implementing a system for delivering prizes to winners.

Security: Online lottery systems must be secure to prevent hacking, fraud, and other forms of abuse. This requires implementing robust security measures like encryption, multi-factor authentication, and intrusion detection systems.

Legal compliance: Online lotteries are heavily regulated, so you'll need to ensure that your system complies with all relevant laws and regulations. This may involve obtaining licenses, following specific rules for game types, and implementing measures to prevent underage gambling.

Scalability: As your user base grows, your system must be able to handle increased traffic and usage. This requires designing a scalable architecture that can handle increased demand without sacrificing performance or reliability.

User experience: Finally, it's important to design an online lottery system that provides a smooth and intuitive user experience. This involves designing an easy-to-use interface, optimizing load times, and providing helpful resources for users.

Overall, designing an online lottery system requires careful planning and attention to detail. By considering these key elements, you can create a secure, reliable, and user-friendly platform that provides a fair and exciting experience for all players.

3.1 MODULE DESCRIPTION

ADMIN

- **VIEW LOTTERY LIST:** admin can view list of lotteries that registered into the system and it details.
- **APPROVE LOTTERY:** admin can approve lottery then only it can be start selling
- **VIEW USER LIST:** admin can view the list of users and their profile that registered into the system
- **VIEW RESULTS:** admin can view the results of lottery.

LOTTERY

- **LOTTERY REGISTRATION:** can register lottery with essential details into the system.
- **VIEW LOTTERY DETAILS:** can view the lottery details and edit the details.
- **START SELLING:** can start the selling of the lottery.
- **PUBLISH RESULT:** can publish the result after the draw date.
- **VIEW RESULT:** can view the result of the lottery.

USER

- **USER REGISTRATION:** user will register into the system with his/her details.
- **VIEW PROFILE:** user can view his/her profile and can edit that.
- **VIEW LOTTERY:** user can view the list of lotteries that available for the draw and its details.
- **PURCHASE LOTTERY (SELF):** user can purchase lottery by himself.
- **VIEW LOTTERY RESULT:** user can view the lottery results that published.

3.2 DATABASE DESIGN

A database is an integrated collection of data and provides centralized access to the data. Usually the centralized data managing the software is called RDBMS.

In this project we were using the database named Google firebase.

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

The Firebase Real-time Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, real-time events continue to fire, giving the end user a responsive experience.

In database design several database objectives are considered

- Easy of learning and use
- Controlled redundancy
- Data independence
- More information at low cost
- Accuracy and integrity
- Recovery of failure
- Privacy and security
- Performance

3.3 TABLE STRUCTURE

A database is a collection of related data. The organization of data in a database must represent the underlying meaning or semantics of data correctly and efficiently. During table design the nature of content of record type, data items that are to be included in the record and characteristics of each data item such as its name, type and width have to be considered. Various factors are considered during the table design phase of the system analysis. Some of them are listed below.

- Purpose of table
- Availability of hardware
- Method of access
- File activity
- File size
- Output requirements
- Input requirements File organization

The design also deals with the design of physical databases. A key field determines how the access to be implemented. Suitable key fields are identified in various tables and are coded appropriately. The type, width and size of the fields are identified

The designing of the tables in database is done according to the rules specified for the database as described above. In this proposed project, 8 tables are used and some of them connect using foreign keys. Inserting and retrieval of values is easy by designing the database in this way.

TABLE NAME: LOTTERY

FIELD	**TYPE**	**CONSTRAINS**	**DESCRIPTION**
lottery_id	VARCHAR(5)	Primary key	lottery id
lottery_name	VARCHAR(20)	Not null	name of the lottery
serial_number	VARCHAR(5)	Not null	serial number of the lottery
ticket_price	INT(5)	Not null	ticket price
status	VARCHAR(20)	Not null	status of the lottery
owner_name	VARCHAR(10)	Not null	name of the owner
mobile_number	INT(10)	Not null	mobile number of the owner
address	VARCHAR(100)	Not null	address of the owner
city	VARCHAR(20)	Not null	city of the owner
email	VARCHAR(20)	Not null	email of the owner
password	VARCHAR(10)	Not null	password of the owner

TABLE NAME: LOTTERY_PRIZE

FIELD	**TYPE**	**CONSTRAINS**	**DESCRIPTION**
prize_id	VARCHAR(5)	Primary key	prize id
lottery_id	VARCHAR(5)	Foreign key	lottery id
price_name	VARCHAR(20)	Not null	name of the prize
count	INT(5)	Not null	prize count
price amount	INT(20)	Not null	prize amount

TABLE NAME: LOTTERY_DRAW

FIELD	**TYPE**	**CONSTRAINS**	**DESCRIPTION**
draw_id	VARCHAR(5)	Primary key	draw id
lottery_id	VARCHAR(20)	Foreign key	lottery id
draw_date	VARCHAR(20)	Not null	draw date

TABLE NAME: USER

FIELD	TYPE	CONSTRAINS	DESCRIPTION
user_id	VARCHAR(5)	Primary key	user id
name	VARCHAR(20)	Not null	name of the user
mobile_number	INT(10)	Not null	mobile number of the user
address	VARCHAR(100)	Not null	address of the user
city	VARCHAR(20)	Not null	city of the user
email	VARCHAR(20)	Not null	email of the user
password	VARCHAR(10)	Not null	password of the user

TABLE NAME: LOTTERY_TICKETS_USER

FIELD	TYPE	CONSTRAINS	DESCRIPTION
draw_id	VARCHAR(5)	Foreign key	draw id
ticket_no	VARCHAR(20)	Not null	draw date
user_id	VARCHAR(5)	Foreign key	user id

TABLE NAME: RESULTS

FIELD	TYPE	CONSTRAINS	DESCRIPTION
result_id	VARCHAR(5)	Primary key	result id
draw_id	VARCHAR(5)	Foreign key	draw id
prize_id	VARCHAR(5)	Foreign key	prize id
ticket_no	VARCHAR(20)	Not null	ticket number

3.4 SAMPLE CODE

- Registration

```
ClassReg(Resource):
User        def post(self):
        userargs=userRegParser.parse_args()
                userCnt = db.child(childName).child('userCnt').get().val()
                if userCnt == None:
                        userCnt = 0
                userCnt += 1
                userID = 'USR' + str(userCnt + 100)
                db.child(childName).child('userCnt').set(userCnt)
                fname = userID + '.jpg'
                f = args['photo']
                del args['photo']
                storage.child(childName).child('userImage').child(fname).put(f)
                args['imgUrl']                                                  =
storage.child(childName).child('userImage').child(fname).get_url(None)
                userList = db.child(childName).child('userList').get().val()
                if userList == None:
                        userList = {}
                userList[userID] = args
                db.child(childName).child('userList').set(userList)
                return userList

        def get(self):
                userList = db.child(childName).child('userList').get().val()
                if userList == None:
                        userList = {}
                return userList
```

- Login

```
class Login(Resource):
```

```
  def post(self):
    args=loginParser.parse_args()
    loginID = ''
    if args['cat'] == 'user':
      userList = db.child(childName).child('userList').get().val()
      if userList == None:
          userList = {}
      for i in userList:
        if userList[i]['email'] == args['username'] and userList[i]['password'] ==
args['password']:
          loginID = i
          break
    elif args['cat'] == 'lottery':
     lotteryList = db.child(childName).child('lotteryList').get().val()
     if lotteryList == None:
            lotteryList = {}
     for i in lotteryList:
            if lotteryList[i]['email'] == args['username'] and
lotteryList[i]['password'] == args['password']:
                  loginID = i
                  break
    elif args['cat'] == 'admin':
     adminEmail =
db_admin.child(childName).child('adminEmail').get().val()
     if adminEmail == None:
            adminEmail = 'admin@gmail.com'
            db_admin.child(childName).child('adminEmail').set(adminEmail)
     adminPassword =
db_admin.child(childName).child('adminPassword').get().val()
     if adminPassword == None:
            adminPassword = 'admin123'
```

```
    db_admin.child(childName).child('adminPassword').set(adminPasswor
d)
    if adminEmail == args['username'] and adminPassword ==
args['password']:
            loginID = 'admin'
    else:
     abort(400, message='invalid category')
    return loginID
```

- Lottery Registration

```
class LotteryReg(Resource):
    def post(self):
            args=lotteryRegParser.parse_args()
            if args['ticketPrice'] <= 0:
                    abort(400, message='not less than zero')
            if args['firstPrice'] <= 0:
                    abort(400, message='not less than zero')
            if args['secondPrice'] <= 0:
                    abort(400, message='not less than zero')
            if args['thirdPrice'] <= 0:
                    abort(400, message='not less than zero')
            if args['fourthPrice'] <= 0:
                    abort(400, message='not less than zero')
            if args['fifthPrice'] <= 0:
                    abort(400, message='not less than zero')
            if args['fourthCnt'] <= 0:
                    abort(400, message='not less than zero')
            if args['fifthCnt'] <= 0:
                    abort(400, message='not less than zero')
            lotteryCnt = db.child(childName).child('lotteryCnt').get().val()
            if lotteryCnt == None:
                    lotteryCnt = 0
```

```
            lotteryCnt += 1
            lotteryID = 'LOT' + str(lotteryCnt + 100)
            db.child(childName).child('lotteryCnt').set(lotteryCnt)
            fname = lotteryID + '.jpg'
            f = args['photo']
            del args['photo']
            storage.child(childName).child('lotteryImage').child(fname).put(f)
            args['imgUrl']                                                    =
storage.child(childName).child('lotteryImage').child(fname).get_url(None)
            lotteryList = db.child(childName).child('lotteryList').get().val()
            if lotteryList == None:
                  lotteryList = {}
            lotteryList[lotteryID] = args
            lotteryList[lotteryID]['status'] = 'notApproved'
            lotteryList[lotteryID]['stage'] = 'initial'
            db.child(childName).child('lotteryList').set(lotteryList)
            return lotteryList

      def get(self):
            lotteryList = db.child(childName).child('lotteryList').get().val()
            if lotteryList == None:
                  lotteryList = {}
            approvedLotteryList = {}
            notApprovedLotteryList = {}
            blockedLotteryList = {}
            for i in lotteryList:
                  if lotteryList[i]['status'] == 'approved':
                        approvedLotteryList[i] = lotteryList[i]
                  elif lotteryList[i]['status'] == 'notApproved':
                        notApprovedLotteryList[i] = lotteryList[i]
                  elif lotteryList[i]['status'] == 'blocked':
                        blockedLotteryList[i] = lotteryList[i]
            lotteryDict = {
```

```
                'approved'    : approvedLotteryList,
                'notApproved' : notApprovedLotteryList,
                'blocked'     : blockedLotteryList
            }
            return lotteryDict
```

- Lottery Action

```
class LotteryAction(Resource):
  def post(self):
    args=lotteryActionParser.parse_args()
    lotteryList = db.child(childName).child('lotteryList').get().val()
    if lotteryList == None:
      lotteryList = {}
    drawCnt = db.child(childName).child('drawCnt').get().val()
    if drawCnt == None:
     drawCnt = 0
    drawList = db.child(childName).child('drawList').get().val()
    if drawList == None:
     drawList = {}
    resultCnt = db.child(childName).child('resultCnt').get().val()
    if resultCnt == None:
     resultCnt = 0
    resultList = db.child(childName).child('resultList').get().val()
    if resultList == None:
     resultList = {}
    purchasedTickets                                                    =
db.child(childName).child('purchasedTickets').get().val()
    if purchasedTickets == None:
     purchasedTickets = {}
    if args['lotteryID'] in lotteryList:
      if args['action'] == 'approve':
        if lotteryList[args['lotteryID']]['status'] == 'notApproved':
          lotteryList[args['lotteryID']]['status'] = 'approved'
```

```
        else:
          abort(400, message='cant be approved')
      elif args['action'] == 'block':
        if lotteryList[args['lotteryID']]['status'] == 'approved':
          lotteryList[args['lotteryID']]['status'] = 'blocked'
        else:
          abort(400, message='cant be blocked')
      elif args['action'] == 'unblock':
        if lotteryList[args['lotteryID']]['status'] == 'blocked':
          lotteryList[args['lotteryID']]['status'] = 'approved'
        else:
          abort(400, message='cant be unblocked')
      elif args['action'] == 'start_selling':
            if args['drawDate']:
                  if (lotteryList[args['lotteryID']]['status'] == 'approved') and
(lotteryList[args['lotteryID']]['stage'] == 'initial' or
lotteryList[args['lotteryID']]['stage'] == 'result_published'):
                        drawCnt += 1
                        drawID = 'DRAW' + str(drawCnt + 100)
                        db.child(childName).child('drawCnt').set(drawCnt)
                        del args['action']
                        drawList[drawID] = args
                        drawList[drawID]['status'] = 'start_selling'
                        lotteryList[args['lotteryID']]['stage'] = 'start_selling'
                  else:
                        abort(400, message='cant start selling')
            else:
                  abort(400, message='draw date required')
      elif args['action'] == 'publish_result':
            if lotteryList[args['lotteryID']]['status'] == 'approved' and
lotteryList[args['lotteryID']]['stage'] == 'start_selling':
                  for i in drawList:
```

```
                    if drawList[i]['lotteryID'] == args['lotteryID'] and
drawList[i]['status'] == 'start_selling':
                        drawID = i
            purchasedUsers = []
            for i in purchasedTickets:
                if purchasedTickets[i]['drawID'] == drawID:
                    purchasedUsers.append(i)
            resultDict = {}
            if purchasedUsers == []:
                resultDict['first'] = { 'pos' : 1, 'purchaseID' :
'notAvailable', 'ticketNo' : 'notAvailable' }
            else:
                if len(purchasedUsers) == 1:
                    purchaseID = purchasedUsers[0]
                else:
                    purchaseID =
purchasedUsers[random.randint(0,len(purchasedUsers)-1)]
                resultDict['first'] = { 'pos' : 1, 'purchaseID' :
purchaseID, 'ticketNo' : purchasedTickets[purchaseID]['ticketNo'] }
                purchasedUsers.remove(purchaseID)
            if purchasedUsers == []:
                resultDict['second'] = { 'pos' : 1, 'purchaseID' :
'notAvailable', 'ticketNo' : 'notAvailable' }
            else:
                if len(purchasedUsers) == 1:
                    purchaseID = purchasedUsers[0]
                else:
                    purchaseID =
purchasedUsers[random.randint(0,len(purchasedUsers)-1)]
                resultDict['second'] = { 'pos' : 1, 'purchaseID' :
purchaseID, 'ticketNo' : purchasedTickets[purchaseID]['ticketNo'] }
                purchasedUsers.remove(purchaseID)
            if purchasedUsers == []:
```

```
                resultDict['third'] = { 'pos' : 1, 'purchaseID' :
'notAvailable', 'ticketNo' : 'notAvailable' }
        else:
                if len(purchasedUsers) == 1:
                        purchaseID = purchasedUsers[0]
                else:
                        purchaseID                                    =
purchasedUsers[random.randint(0,len(purchasedUsers)-1)]
                resultDict['third'] = { 'pos' : 1, 'purchaseID' :
purchaseID, 'ticketNo' : purchasedTickets[purchaseID]['ticketNo'] }
                purchasedUsers.remove(purchaseID
        fourthNo = random.randint(1000,5000)
        fourthList =[]
        for i in range(1, lotteryList[args['lotteryID']]['fourthCnt']+1):
                flag = 0
                if purchasedUsers != []:
                        for j in purchasedUsers:
                                if
purchasedTickets[j]['ticketNo'].endswith(str(fourthNo)):
                                        fourthDict = { 'pos' : i, 'fourthNo' :
fourthNo, 'purchaseID' : j, 'ticketNo' : purchasedTickets[j]['ticketNo'] }
                                        purchasedUsers.remove(j)
                                        flag = 1
                                        break
                if flag == 0:
                        fourthDict = { 'pos' : i, 'fourthNo' : fourthNo,
'purchaseID' : 'notAvailable', 'ticketNo' : 'notAvailable' }
                        fourthList.append(fourthDict)
        resultDict['fourth'] = fourthList
        fifthNo = fourthNo
        while (fifthNo == fourthNo) :
                fifthNo = random.randint(1000,5000)
        fifthList =[]
```

```
                for i in range(1, lotteryList[args['lotteryID']]['fifthCnt']+1):
                    flag = 0
                    if purchasedUsers != []:
                        for j in purchasedUsers:
                            if
purchasedTickets[j]['ticketNo'].endswith(str(fifthNo)):
                                fourthDict = { 'pos' : i, 'fifthNo' :
fifthNo, 'purchaseID' : j, 'ticketNo' : purchasedTickets[j]['ticketNo'] }
                                purchasedUsers.remove(j)
                                flag = 1
                                break
                    if flag == 0:
                        fifthDict = { 'pos' : i, 'fifthNo' : fifthNo,
'purchaseID' : 'notAvailable', 'ticketNo' : 'notAvailable' }
                        fifthList.append(fifthDict)
                resultDict['fifth'] = fifthList
                drawList[drawID]['status'] = 'result_published'
                drawList[drawID]['lotteryDetails'] =
lotteryList[args['lotteryID']]
                lotteryList[args['lotteryID']]['stage'] = 'result_published'
                resultCnt += 1
                resultID = 'RESULT' + str(resultCnt + 100)
                db.child(childName).child('resultCnt').set(resultCnt)
                resultList[resultID] = resultDict
                resultList[resultID]['drawID'] = drawID
                resultList[resultID]['lotteryID'] = args['lotteryID']
            else:
                abort(400, message='cant publish result')
        else:
            abort(400, message='invalid action')
        db.child(childName).child('resultList').set(resultList)
        db.child(childName).child('drawList').set(drawList)
        db.child(childName).child('lotteryList').set(lotteryList)
```

```
        return lotteryList[args['lotteryID']]
    else:
        abort(400, message='lottery not found')
```

- Lottery Result

```
class LotteryResults(Resource):
    def get(self, drawID):
        drawList = db.child(childName).child('drawList').get().val()
        if drawList == None:
            drawList = {}
        resultList = db.child(childName).child('resultList').get().val()
        if resultList == None:
            resultList = {}
        purchasedTickets =
db.child(childName).child('purchasedTickets').get().val()
        if purchasedTickets == None:
            purchasedTickets = {}
        userList = db.child(childName).child('userList').get().val()
        if userList == None:
            userList = {}
        if not drawID in drawList:
            abort(400, message='lottery draw not found')
        if drawList[drawID]['status'] != 'result_published':
            abort(400, message='result not published')
        for i in resultList:
            if resultList[i]['drawID'] == drawID:
                resultDetails = resultList[i]
                break
        if resultDetails['first']['purchaseID'] != 'notAvailable':
            userID =
purchasedTickets[resultDetails['first']['purchaseID']]['userID']
            resultDetails['first']['userID'] = userID
            resultDetails['first']['userDetails'] = userList[userID]
```

```
        if resultDetails['second']['purchaseID'] != 'notAvailable':
            userID =
purchasedTickets[resultDetails['second']['purchaseID']]['userID']
            resultDetails['second']['userID'] = userID
            resultDetails['second']['userDetails'] = userList[userID]
        if resultDetails['third']['purchaseID'] != 'notAvailable':
            userID =
purchasedTickets[resultDetails['third']['purchaseID']]['userID']
            resultDetails['third']['userID'] = userID
            resultDetails['third']['userDetails'] = userList[userID]
        for i in resultDetails['fourth']:
            if i['purchaseID'] != 'notAvailable':
                userID = purchasedTickets[i['purchaseID']]['userID']
                i['userID'] = userID
                i['userDetails'] = userList[userID]
        for i in resultDetails['fifth']:
            if i['purchaseID'] != 'notAvailable':
                userID = purchasedTickets[i['purchaseID']]['userID']
                i['userID'] = userID
                i['userDetails'] = userList[userID]
        return resultDetails
```

3.5 DESIGN

- Input Design:

IDesigning the input for an online lottery web application is a crucial aspect of creating a user-friendly experience for your users. Here are some important considerations when designing the input for an online lottery web application:

Ticket purchase form: You'll need to create a form that allows users to purchase lottery tickets. This form should be easy to understand and use, with clear instructions and validation messages for each field.

Ticket selection: Make it easy for users to select the ticket they want to play, including the ticket type, draw date, and ticket price.

Number selection: For games that require users to select their own numbers, create an intuitive and user-friendly interface for selecting numbers. This can include dropdown menus, input fields, or a visual interface like a grid or wheel.

Quick pick option: To make it easier for users who don't want to select their own numbers, consider offering a "quick pick" option that generates a random set of numbers for them.

Payment options: Users should be able to easily select their payment method and enter payment details, with clear instructions and validation messages throughout the process.

Confirmation page: Once the user has completed their purchase, provide a clear and concise confirmation page that displays their ticket details, including the game, draw date, numbers selected, and payment details.

Error handling: Ensure that your input design includes robust error handling that helps users identify and correct any errors in their input. This can include inline validation messages, error notifications, and error messages that clearly explain the issue and how to fix it.

Accessibility: Consider accessibility when designing your input form, including support for screen readers, keyboard navigation, and other accessibility features that help users with disabilities.

Mobile optimization: Many users will access your online lottery web application from their mobile devices, so it's important to ensure that your input design is optimized for mobile devices. This can include responsive design, touch-friendly inputs, and simplified layouts that are easy to use on smaller screens.

Overall, designing the input for an online lottery web application requires careful consideration of user needs, usability, and accessibility. By following these best practices, you can create an input design that provides a seamless and enjoyable experience for your users.

- Output Design:

Designing the output for an online lottery web application requires careful consideration of several factors, including the user interface, the types of information to be displayed, and the overall user experience. Here are some key elements to consider when designing the output for an online lottery web application:

Lottery Results: The main output of a lottery web application is the lottery results. The results should be displayed in a clear and concise manner, with the winning numbers and any relevant information prominently displayed.

The results should be easy to read and understand, and users should be able to easily find the information they are looking for.

User Interface: The user interface of the lottery web application should be designed to be intuitive and user-friendly. Users should be able to easily navigate the application, find the information they need, and purchase tickets if they choose to do so. The interface should be responsive and work well on all devices, including desktops, tablets, and smartphones.

Ticket Purchase: If the web application allows users to purchase lottery tickets, the ticket purchase process should be straightforward and easy to complete. Users should be able to select the numbers they want to play, choose the number of tickets they want to purchase, and complete the payment process with minimal hassle.

Overall, the design of the output for an online lottery web application should focus on providing a seamless user experience that is easy to navigate and understand. The lottery results should be displayed clearly and prominently, and the ticket purchase process should be straightforward and hassle-free. With the right design, an online lottery web application can provide a fun and exciting way for users to try their luck and potentially win big.

4. SYSTEM IMPLEMENTATION

Implementing an online lottery web application involves several components and steps. Here is a high-level overview of the process:

Planning: The first step in implementing an online lottery web application is to plan the project. This includes identifying the requirements of the system, designing the architecture of the application, and creating a project plan.

Database Design: The next step is to design the database for the application. This includes creating tables to store user data, lottery ticket data, and other relevant information.

User Management: The online lottery web application needs to have a user management system in place. This includes user registration, login, and authentication. Users should be able to create an account, manage their profile, and view their ticket purchase history.

Ticket Purchase: The application should allow users to purchase lottery tickets online. This involves integrating a payment gateway for accepting payments, generating unique ticket numbers, and storing the ticket data in the database.

Lottery Drawing: The application should have a mechanism for conducting the lottery drawing. This could involve integrating a third-party lottery API or building a custom algorithm for selecting winning numbers.

Security: The online lottery web application must be secure to prevent unauthorized access to user data and ensure fair play. This includes implementing secure communication protocols, using encryption to protect sensitive data, and implementing measures to prevent fraud.

Testing: Before deploying the application, it should be thoroughly tested to ensure that all features are working as intended and that there are no bugs or security vulnerabilities.

Deployment: Finally, the application can be deployed to a web server and made available to the public.

Overall, implementing an online lottery web application requires a thorough understanding of web development, database design, and security best practices. It is essential to plan the project carefully and follow industry standards to ensure the application is robust and secure.

5. SYSTEM TESTING

System testing for an online lottery web application involves a thorough testing process to ensure that the application is working correctly and meets the requirements of the stakeholders. Here are some steps that can be taken to conduct system testing for an online lottery web application:

Functional Testing: This involves testing the functionality of the lottery web application. Testers should verify that the application is performing as expected by performing various functional tests such as verifying that the lottery tickets can be purchased, that users can view the winning numbers, and that the website can handle multiple user requests at the same time.

Usability Testing: This type of testing evaluates how user-friendly the application is. Testers should consider the ease of use, navigation, and overall user experience of the application. This type of testing should be performed from the perspective of both experienced and novice users.

Compatibility Testing: It is essential to ensure that the web application can run on multiple devices and operating systems without any compatibility issues. Testers should test the lottery web application on various browsers and devices to ensure that it works correctly.

Security Testing: Security is an essential aspect of an online lottery web application. Testers should conduct security testing to identify any vulnerabilities that could be exploited by hackers. This testing should include checking for security issues.

Performance Testing: The online lottery web application should be able to handle a large number of users without slowing down or crashing. Testers should conduct performance testing to determine the application's performance under heavy load and identify any bottlenecks.

Regression Testing: Any changes or updates to the lottery web application should be tested to ensure that they do not affect the application's existing functionality. Testers should conduct regression testing to identify any defects that may have been introduced as a result of the changes.

By following these testing steps, the online lottery web application can be tested thoroughly to ensure its proper functionality, usability, compatibility, security, performance, and regression readiness.

6. CONCULSION AND FUTURE ENHANCEMENT

An online lottery system named ONE PIECE LOTTERY is developed. In contrast to the paper lottery system, the proposed model is being easy to use and time consuming. The system has many future enhancements one important feature want to implememt is shared buying by creating the group. In paper lottery, Syndicates are used because the logic is quite simple. The more tickets you buy, The more chance of winning you get. Therefore people often from syndicates with there friends or co-workers. But the problem is that someone has to take responsibility of the lottery.

In this system we can list the members of the people who buy lottery by creating the group that will list the number of peoples and their share price. So its easy to implement at the time of publishing the result.

Other feature such as fee payment has to be include also has to improve the security. The application mainly focused on selling and buying the lottery and provide the result through online.

Also In future we aim to further improve the security to make the proposed method even more suitable for real time application.

7.BIBLIOGRAPHY

Websites :

- Wikipedia
- ChatGPT
- www.keralalotteryresult.com

Application :

- Online lottery games such as
- Dream 11
- Rummy Circle

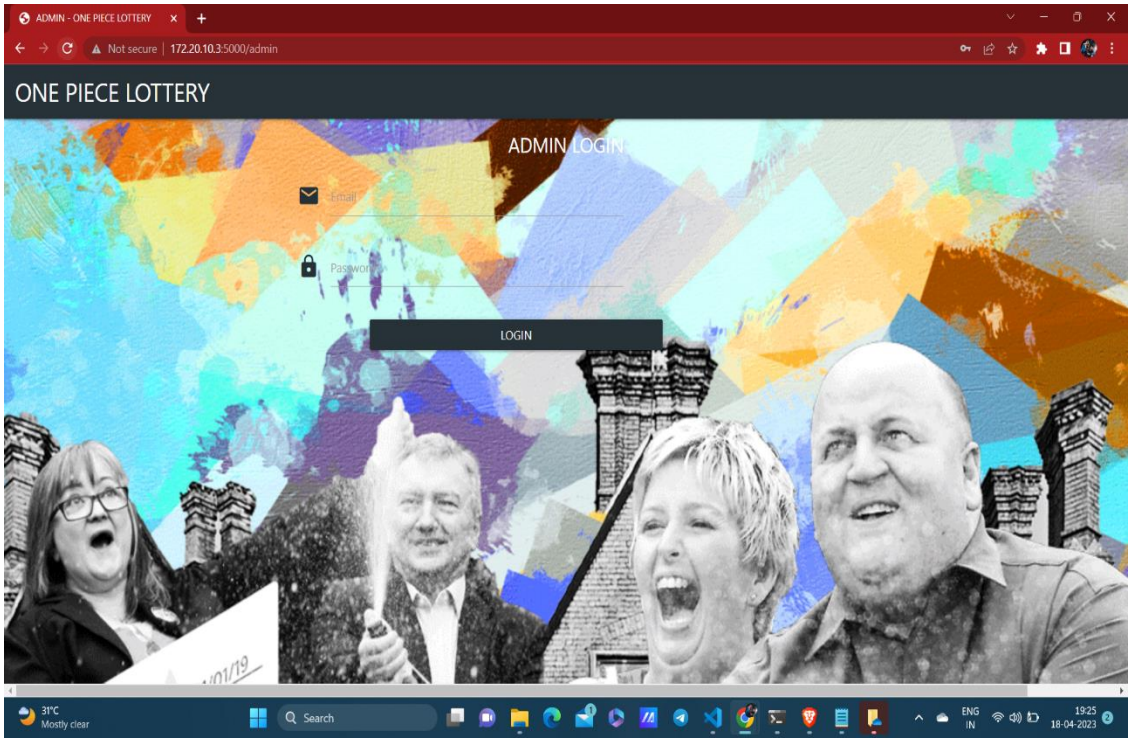
Books:

- Software Engineering-Roger s pressman

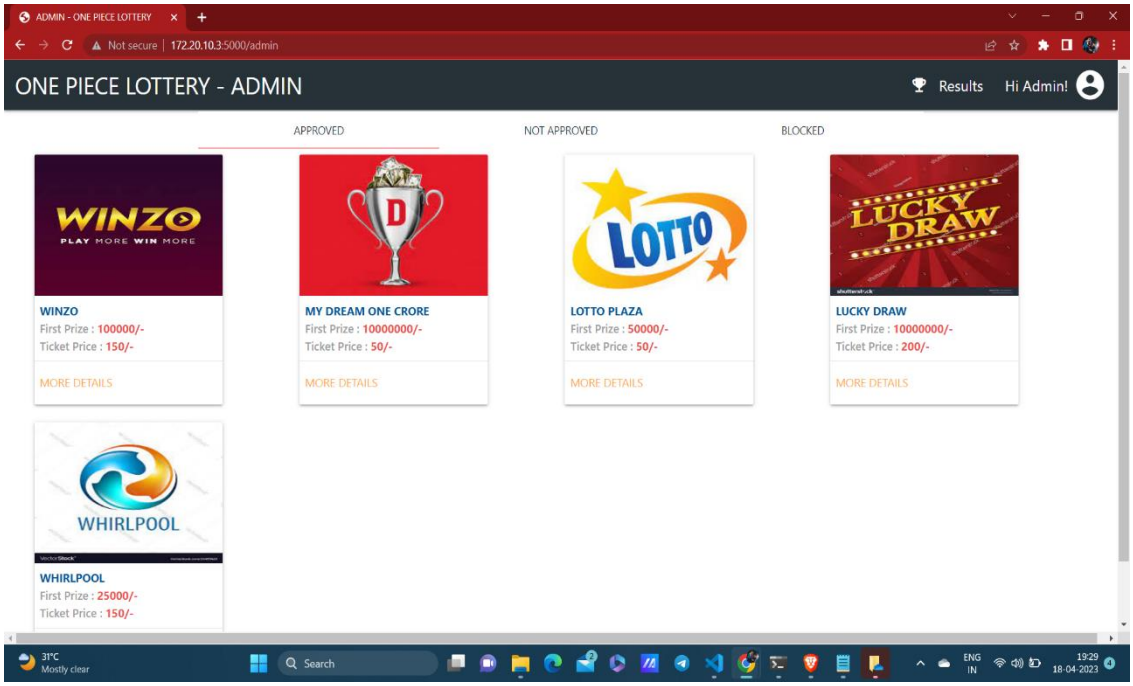
Software Testing Principle's and Practices-Srinivasan Desikan

8.APPENDIX

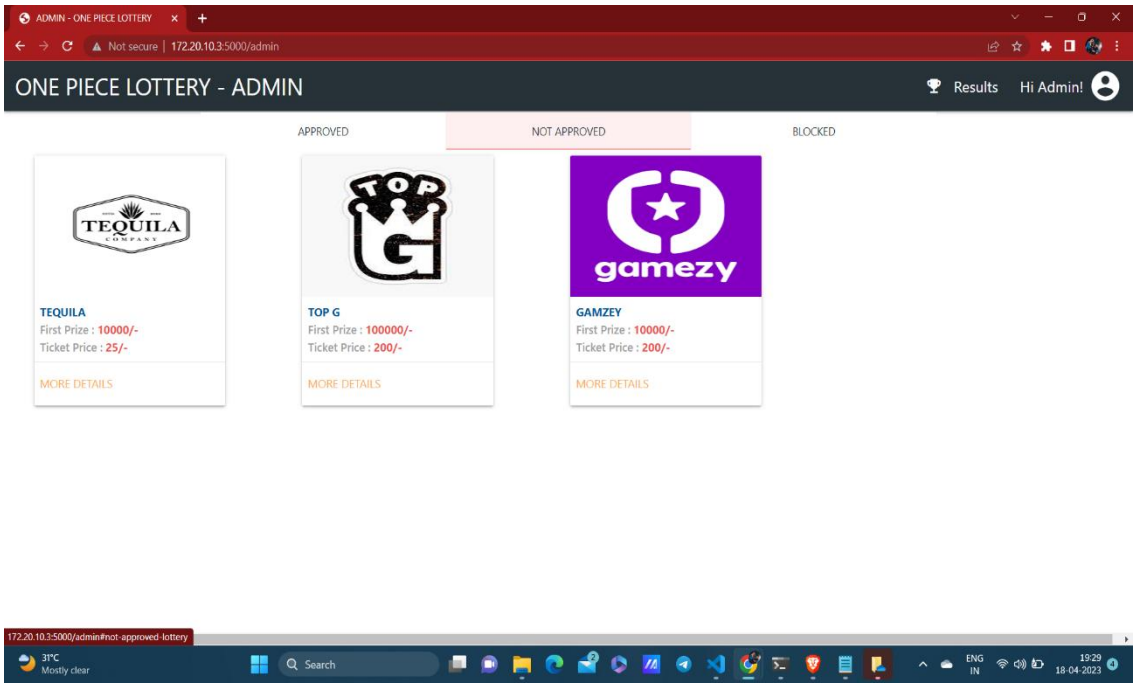
ADMIN LOGIN



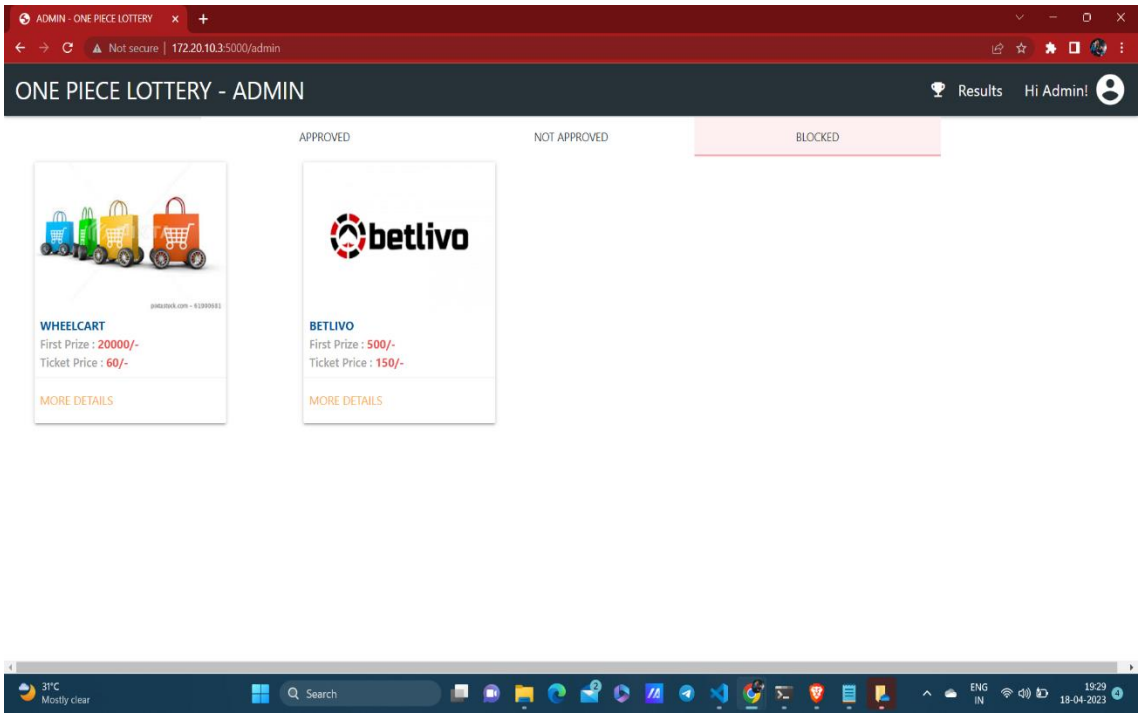
APPROVED LOTTERY



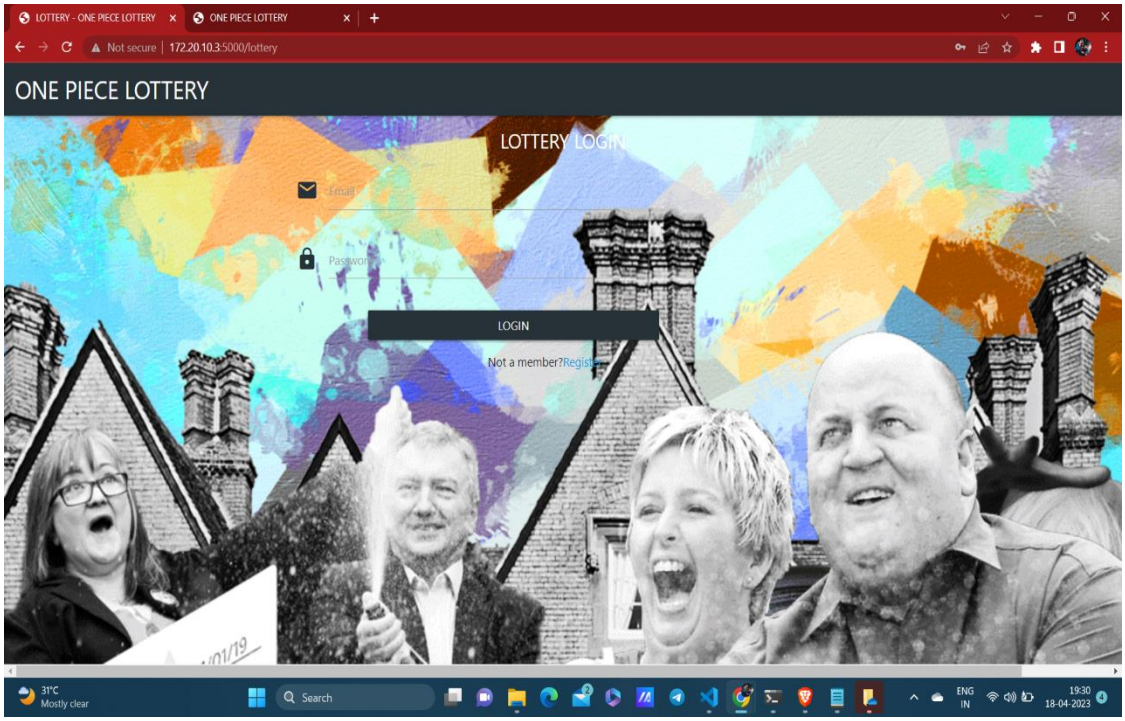
NOT APPROVED LOTTERY



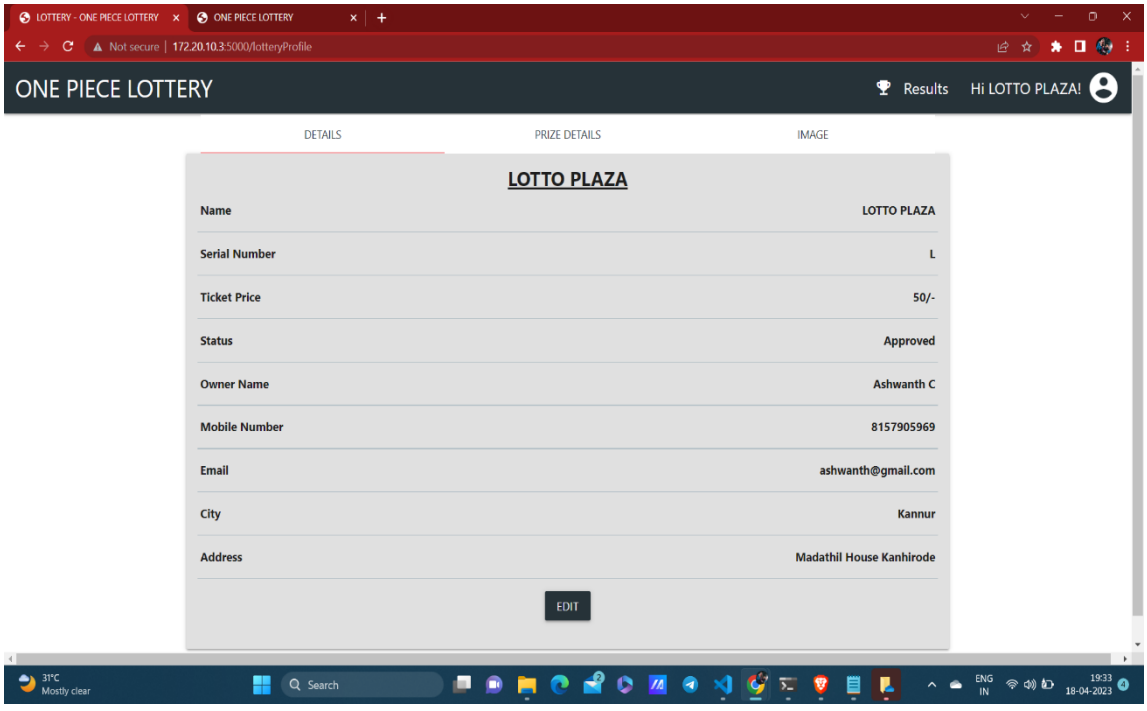
BLOCKED LOTTERY



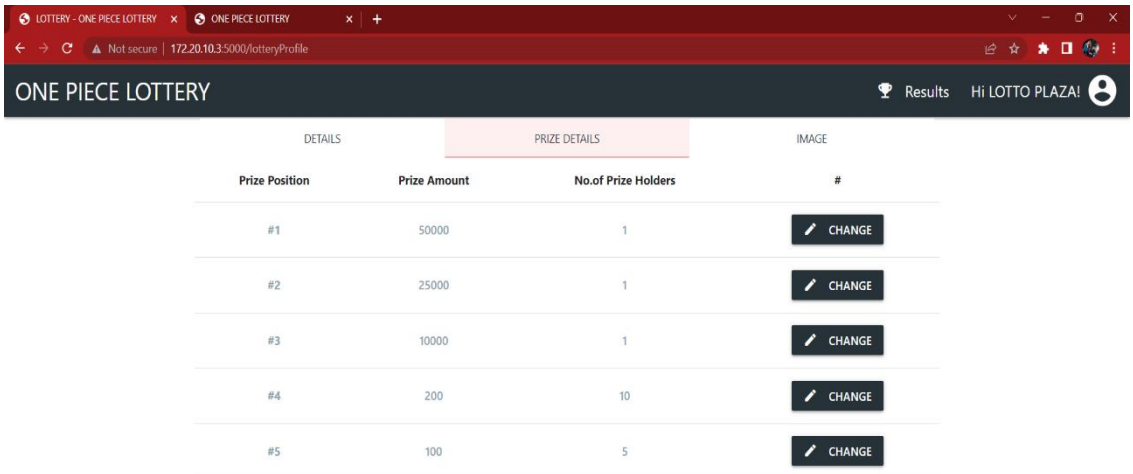
LOTTERY LOGIN



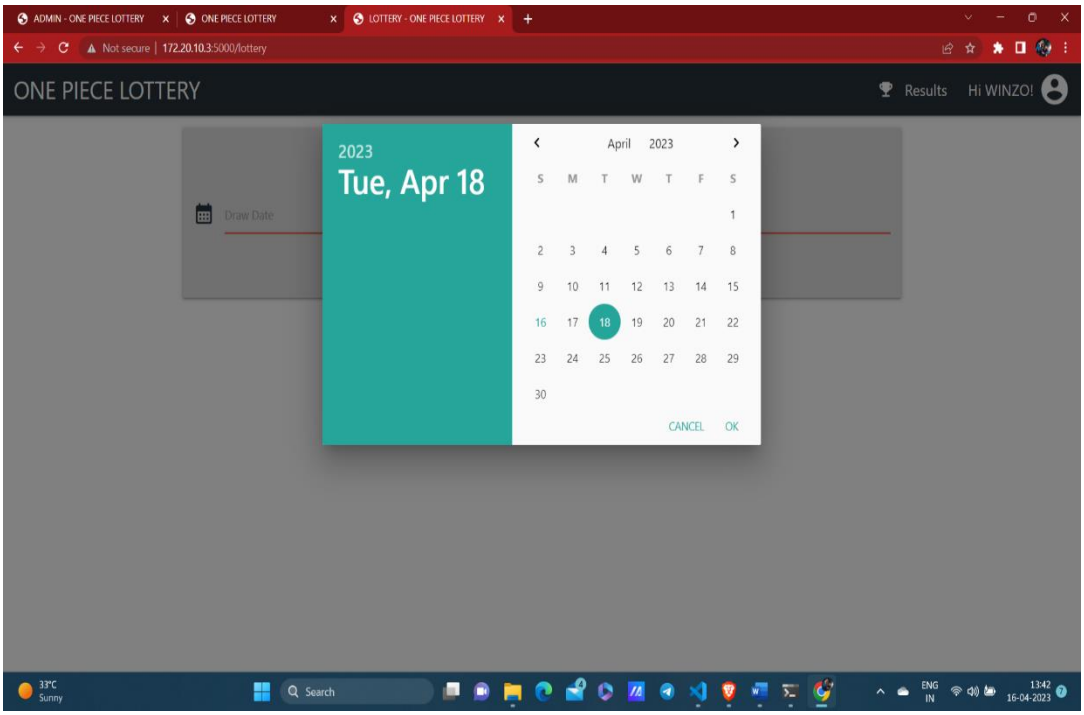
LOTTERY DETAILS AND EDIT



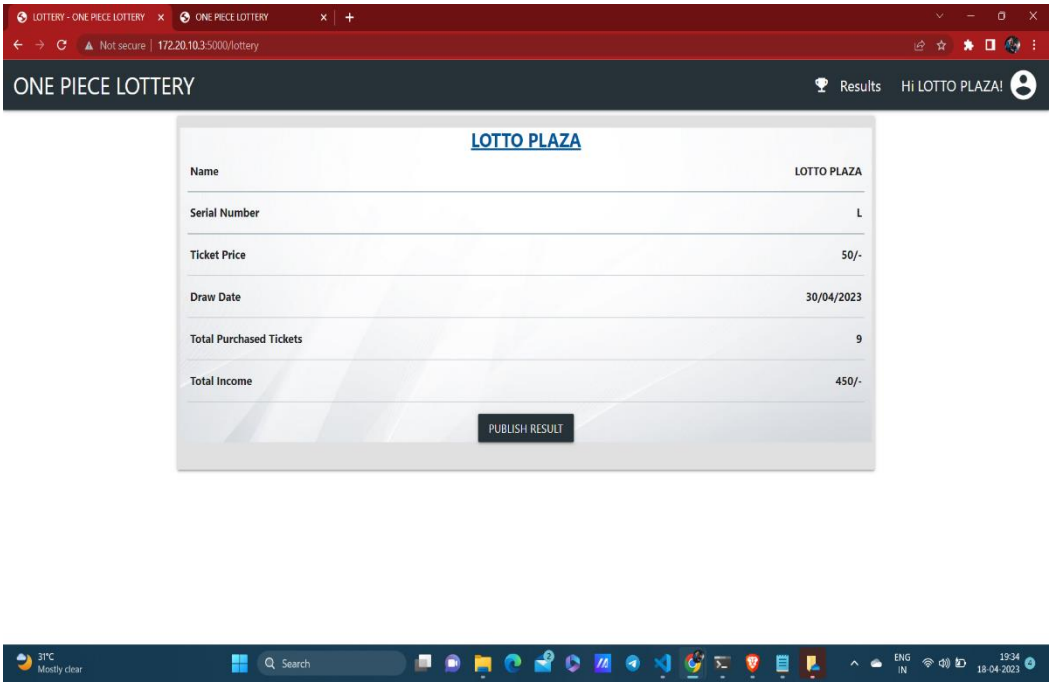
LOTTERY PRIZE DETAIL AND EDIT



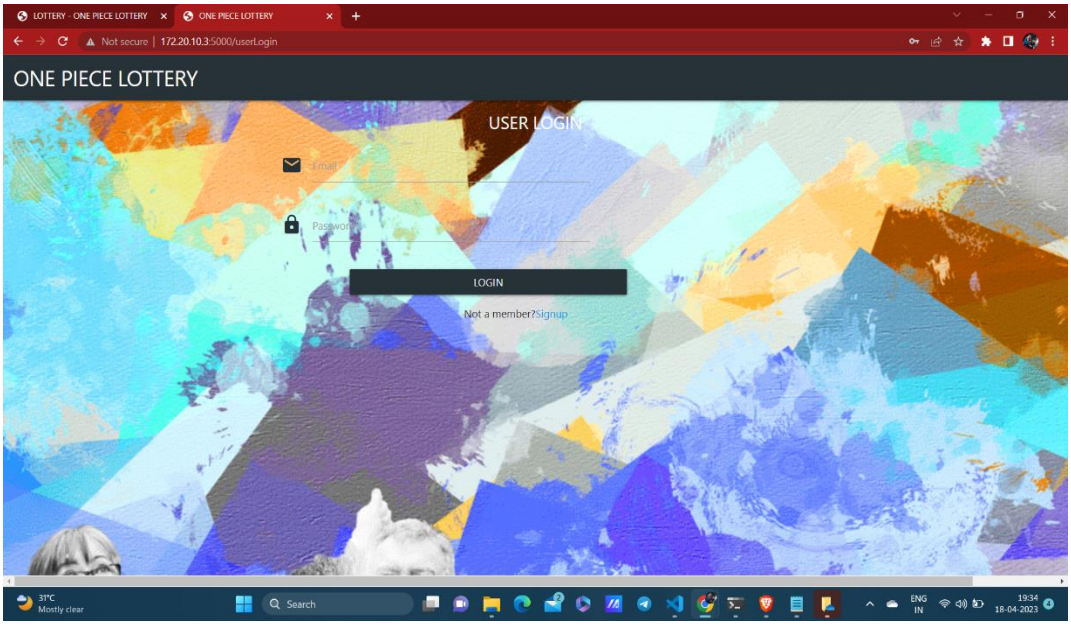
LOTTERY SELLING



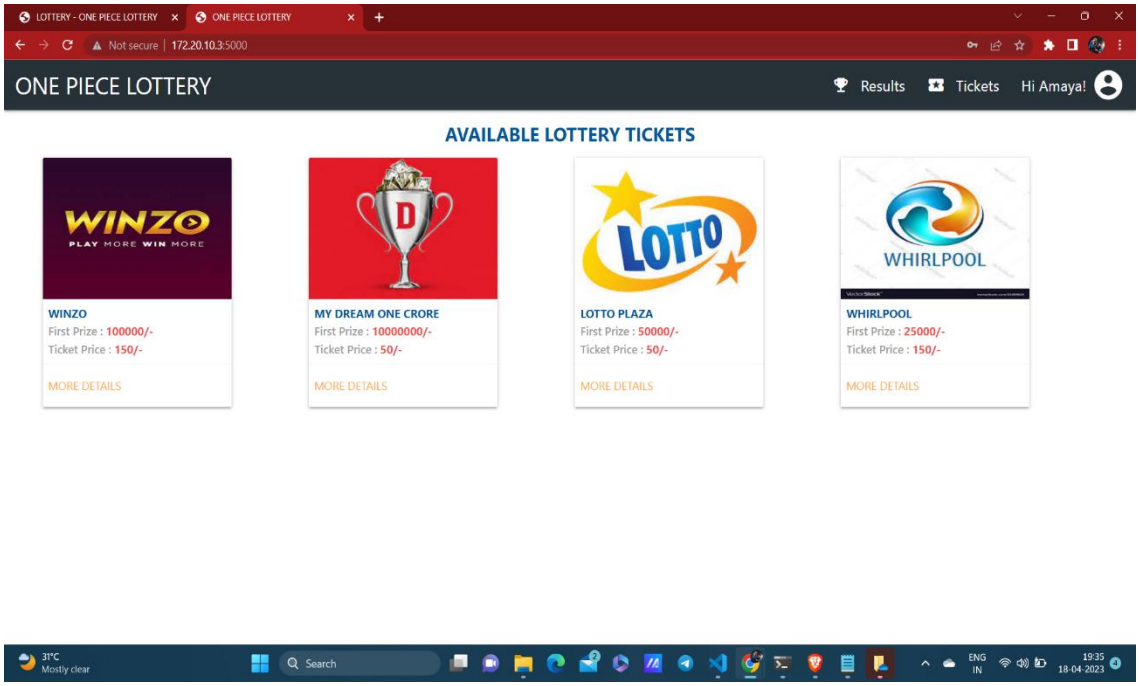
PUBLISH RESULT



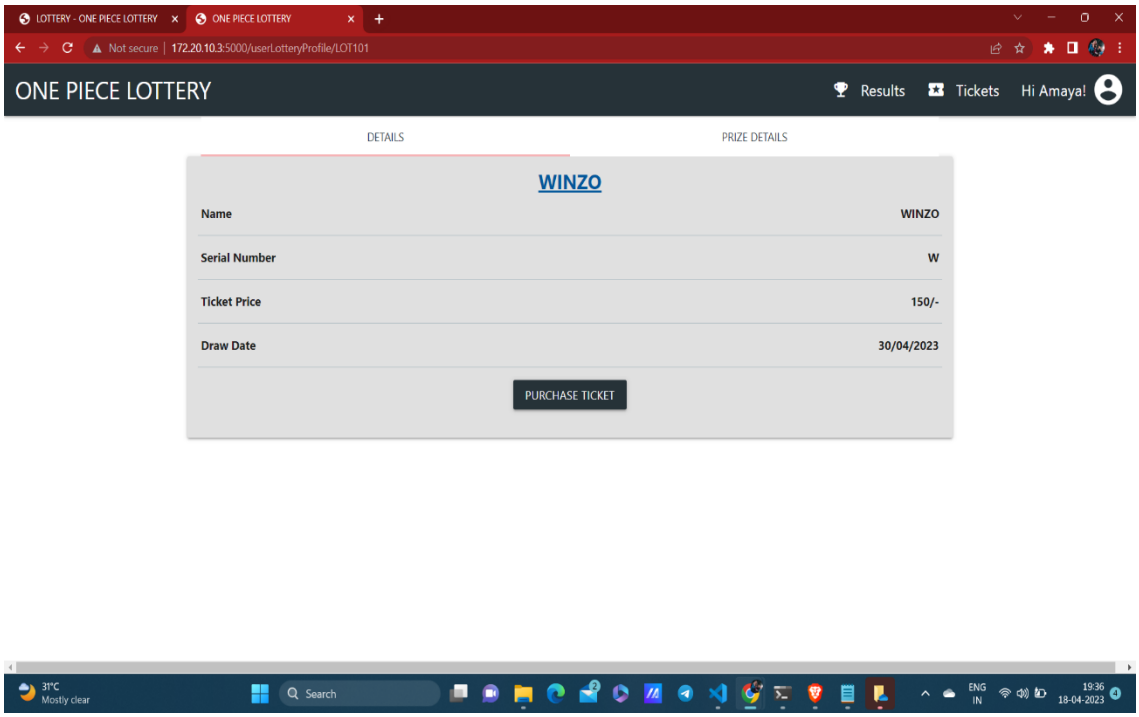
USER LOGIN



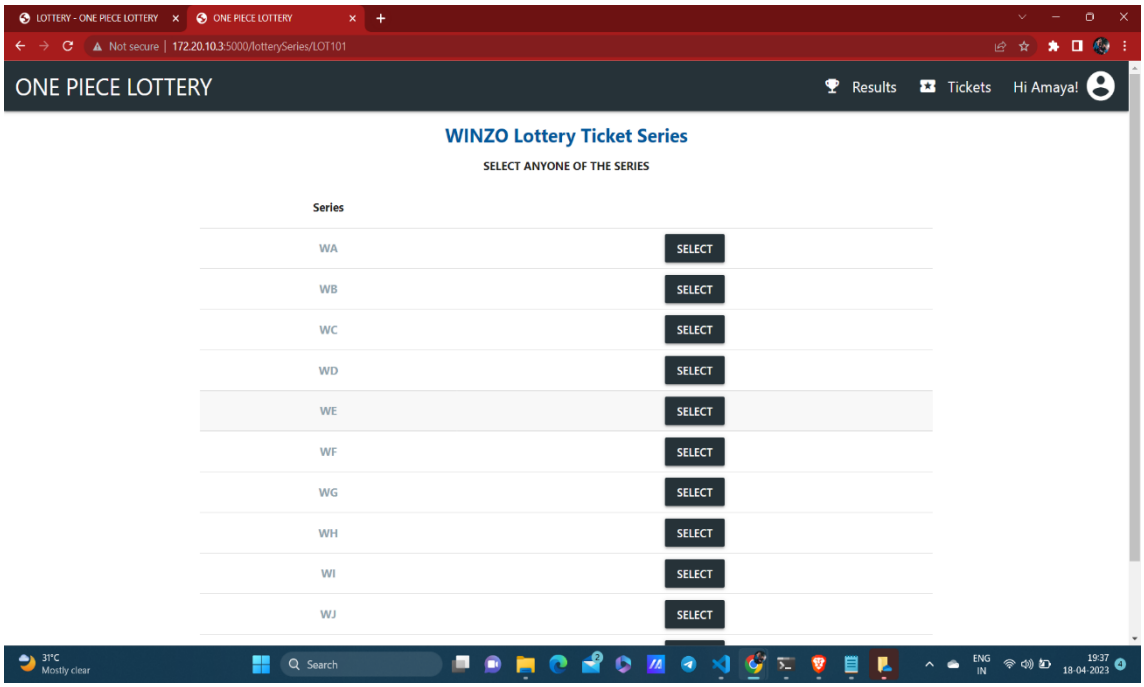
AVAILBALE LOTTERY TICKETS



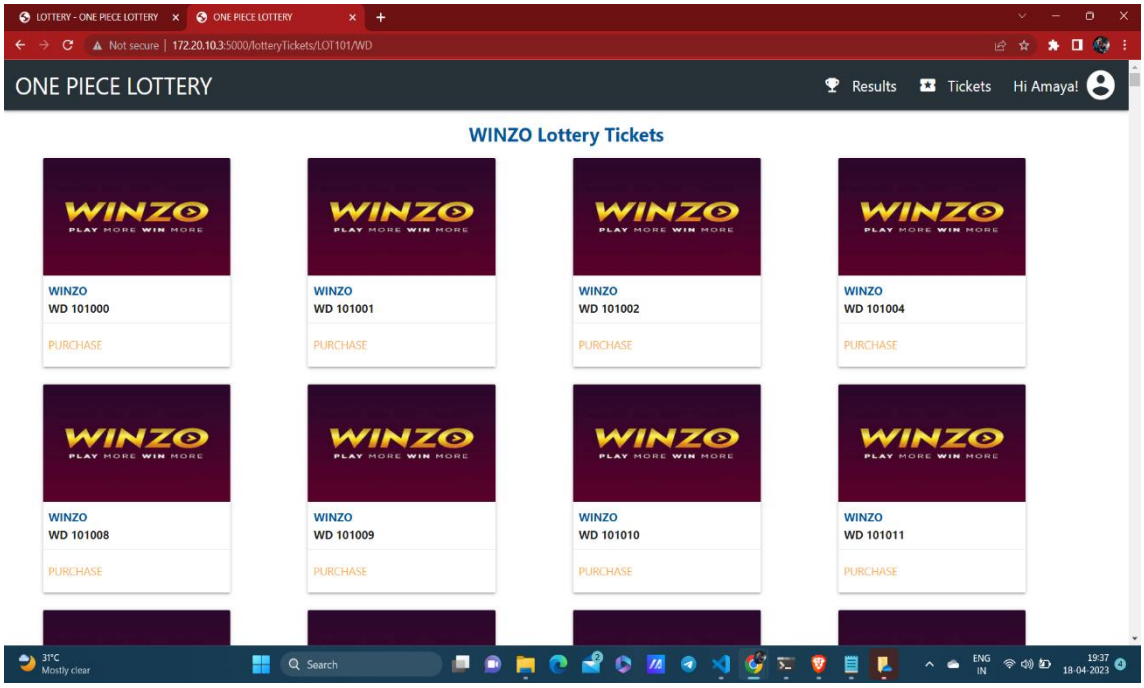
PURCHASING LOTTERY
STEP 1



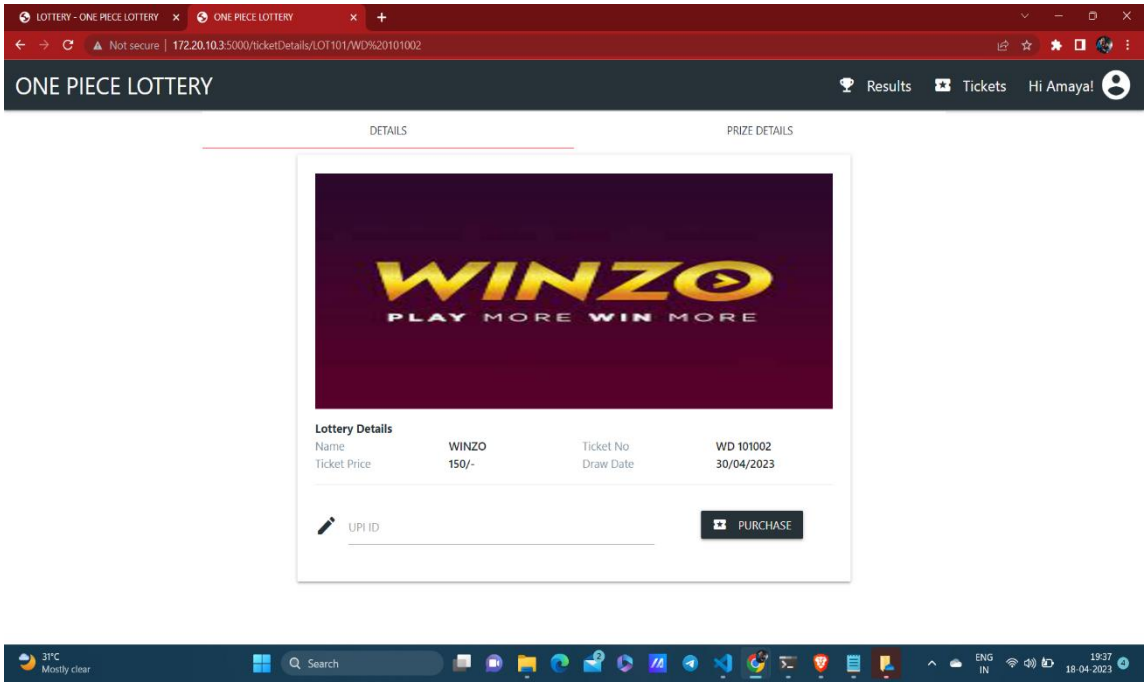
STEP 2


LOTTERY - ONE PIECE LOTTERY
ONE PIECE LOTTERY
Not secure | 172.20.10.3:5000/lotterySeries/LOT101
ONE PIECE LOTTERY
Results
Tickets
Hi Amaya!
WINZO Lottery Ticket Series
SELECT ANYONE OF THE SERIES
Series
WA
WB
WC
WD
WE
WF
WG
WH
WI
WJ
SELECT

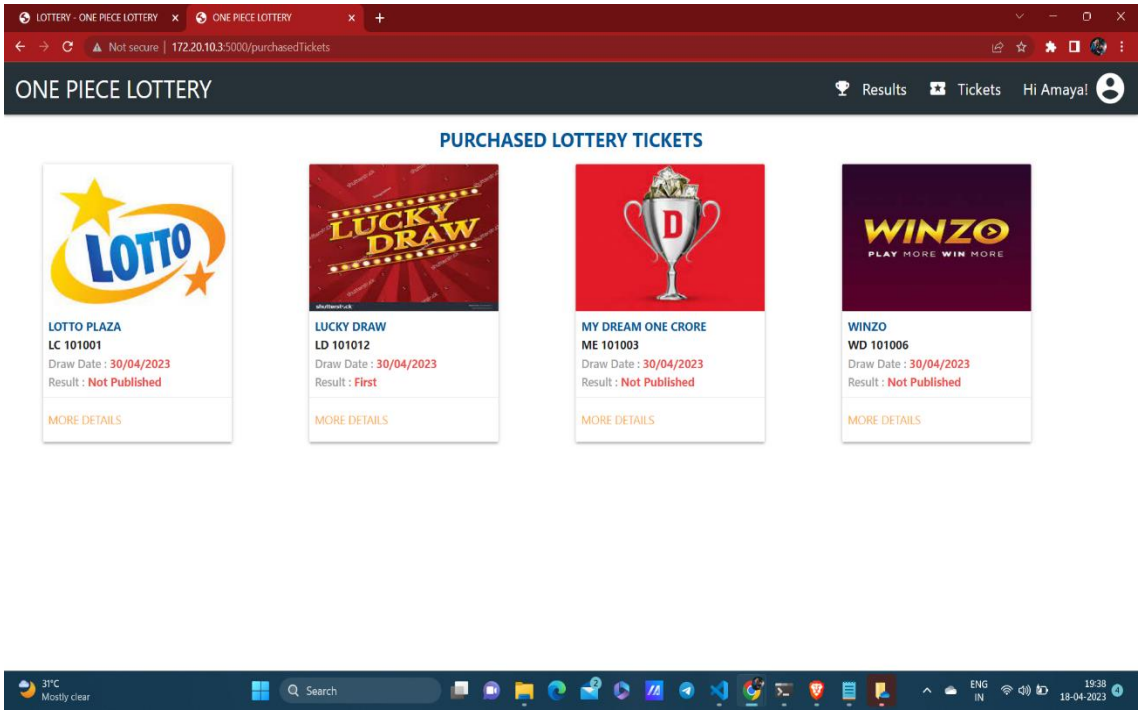
STEP 3


LOTTERY - ONE PIECE LOTTERY
ONE PIECE LOTTERY
Not secure | 172.20.10.3:5000/lotteryTickets/LOT101/WD
ONE PIECE LOTTERY
Results
Tickets
Hi Amaya!
WINZO Lottery Tickets
WINZO
WD 101000
WD 101001
WD 101002
WD 101004
WD 101008
WD 101009
WD 101010
WD 101011
PURCHASE

STEP 4

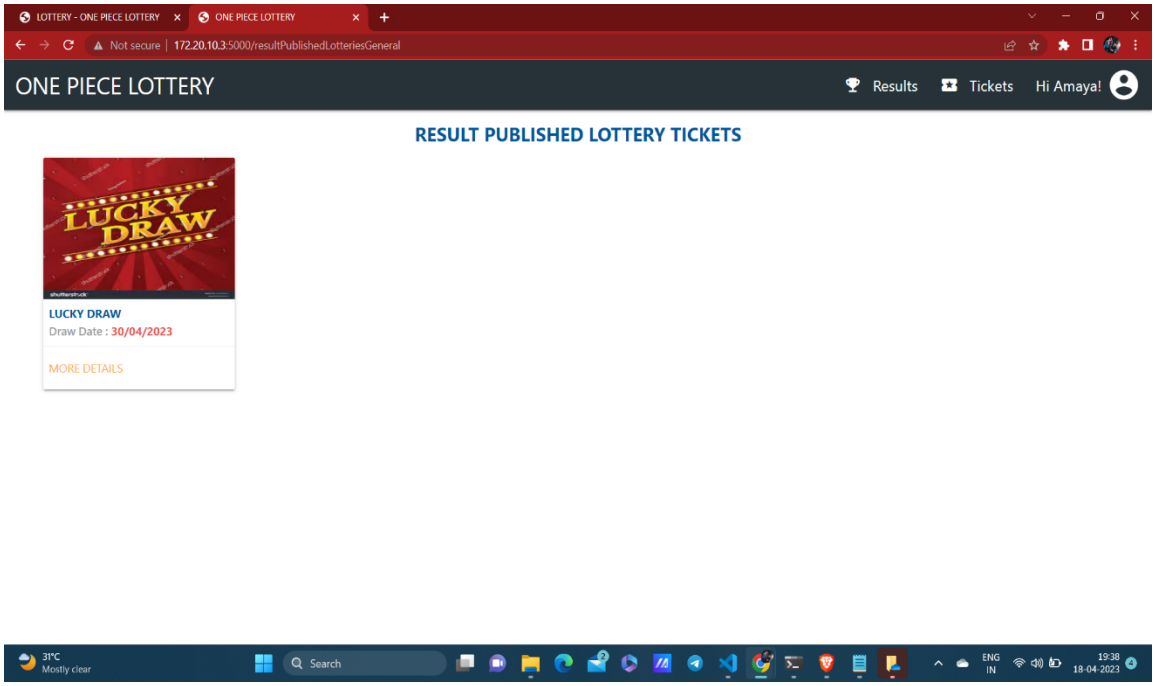


USER PURCHASED LOTTERY TICKETS



LOTTERY TICKETS RESULTS VIEW

A)



B)