Name:Daniel Mark:

Database Assignment

Purpose and End User of my Database

I have an idea for improving the high school sports betting system. My plan is to create a database that requires users to enter their name and their balance will appear under their name. This will allow users to choose which sport they want to bet on and then they can specify how much they want to bet on the team they have chosen. This will make the high school betting system more reliable and provide a safer and more secure way for users to participate in high school betting.

Describe at least 3 implications that are relevant to your database and its use by the end user and why they are important

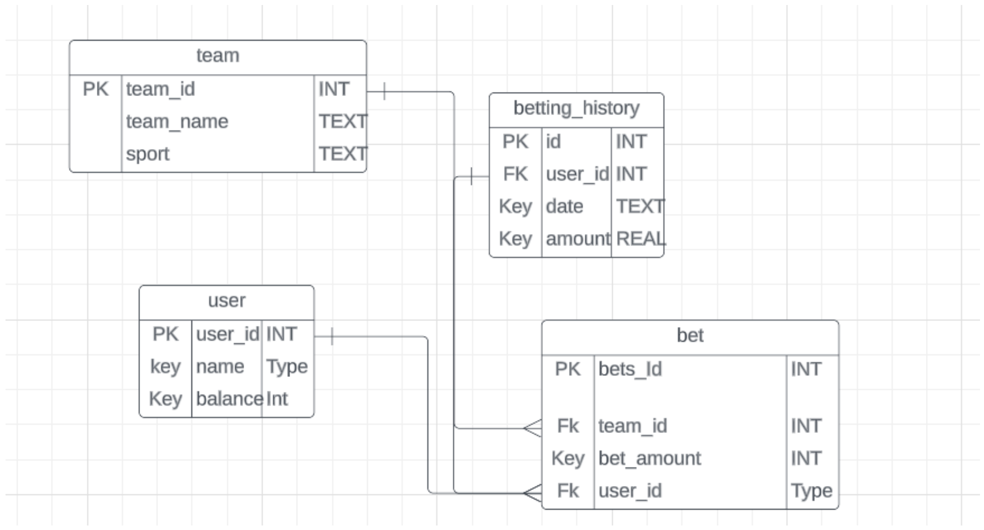
The user should always be able to have full **Accessibility** to the app, they should also be allowed to access it from anywhere, at any time, and from any device. Users have different preferences when accessing online platforms. Some may want to use a desktop or iPad. But some people choose to use mobile, so the database must be device-compatible so more users can use it. Allowing users to access the platform that is most convenient. Users can place bets on their mobile phone while being on the go instead of going home and placing a bet on the computer and then going back to what they were doing. By providing full accessibility the users will get engaged in the platform providing user satisfaction. It also allows individuals from various locations and time zones to access high school betting.
Future-proofing the database has to be in my database to satisfy the user. This can be achieved by regular maintenance of this platform is key to making it run up to date. This involves monitoring and resolving bugs in the software, performance issues, and security checks and for example, adding new functions such as asking for security checks like what age you are. This database also has to be manually updated. This is because the developers can add features to how they like to what they want. An example would be adding different fonts to the database or making it easier for the user.

Privacy My data should have enough privacy that the person with the right user name can only enter this is to keep my user nice and secure on the program. my data will also be secured this is because the account remains to the holder of the account so if people do not know the person's account name they wouldn't be able to access the person's account. This can also be controlled by the database. If i random person gets onto the account and takes the money I could make the person regain his money back by returning the sum number. This can make the user feel safe and secure while using the

Name:Daniel Mark:

program.

Database Design- Your Entity Relationship Diagram.



Database Testing Table: SQL Statements

Purpose	**SQL Statement**	**Result Success?**
to select everything from the user. And then select the table name called the name	SELECT * FROM user WHERE name = 'name'	yes

Name:Daniel Mark:

Select the balance total from the user.	SELECT balance FROM user WHERE name = ‘name’	yes
Select all the sports names from the sports	SELECT sports_name FROM sport	yes
Select the team name from the team and then the result is from the sports name.	SELECT teamname FROM team WHERE team_id = ‘sports_id’	yes
Update the user's balance if to take money or put money on the user account	UPDATE user SET balance = balance + amount WHERE name = ‘name’	Yes
Inserts account history. From the bet they placed.	INSERT INTO betting_history(user_id, date, amount)VALUES ({name}),Date (now), (amount)	Yes
Prints out the account history data	SELECT date,amount FROM betting_history WHERE user_id = {?}	

Relevant Implications- Explain how your database addresses the relevant implications that you identified at the start.

My code finally functions and is finished, therefore I can start explaining the relevant implications. Firstly I said that my code had to be accessible to any and anywhere that's what my code can do. My code can only currently function on a computer, but later in my life I couldn't start working on mobile and later on any device. While currently, the program works best if accessed with a computer. The user if they had a computer can still access the program from any and everywhere. Users can also be allowed access from any country and any time zone according to the program. The future-proofing of this program I made can be monitored and regularly maintained as I have the most control over the program. I can also update the program manually as said above, I could also add different fonts to make the user satisfied and more engaged with the program. If a person requests to change the name or something I can change it manually to the exact thing they requested so they wouldn't have to

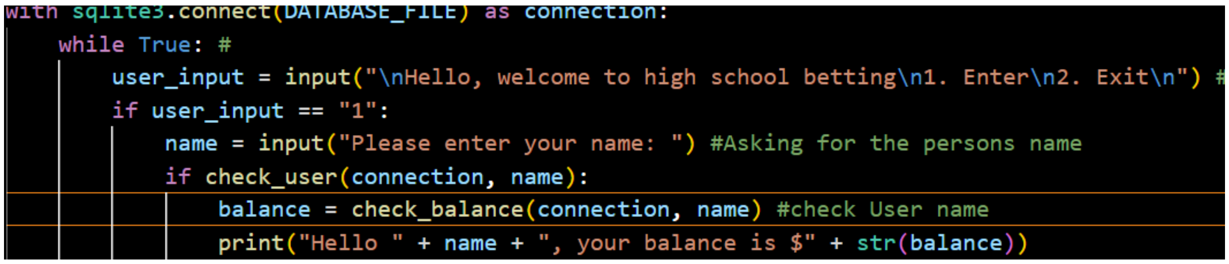
Name:Daniel Mark:

choose another brand because they will be satisfied with the exact thing they asked for. I said that I should have good security and that's what I have done. The program can ask for your identity when entering the program to make sure you are the right person entering the program. This is to keep the user safe and nobody can take their money. My security can also be controlled by me as I have the database for every account and can control everyone's account. The next time I could improve would be by asking the user to enter a new password and the password gets updated into the data to keep the user more safe.

Showcase:

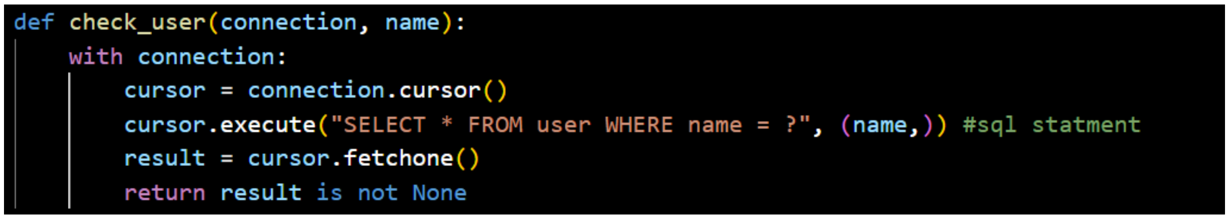
Give evidence of your database and the Python code that interfaces with it. Use screenshots or a short video. Explain how it improved, how it functions, how it was tested etc.

From the start of the code, I decided to ask the user for their name to enter the program.
The starting screen



```
with sqlite3.connect(DATABASE_FILE) as connection:
    while True: #
        user_input = input("\nHello, welcome to high school betting\n1. Enter\n2. Exit\n") #
        if user_input == "1":
            name = input("Please enter your name: ") #Asking for the persons name
            if check_user(connection, name):
                balance = check_balance(connection, name) #check User name
                print("Hello " + name + ", your balance is $" + str(balance))
```

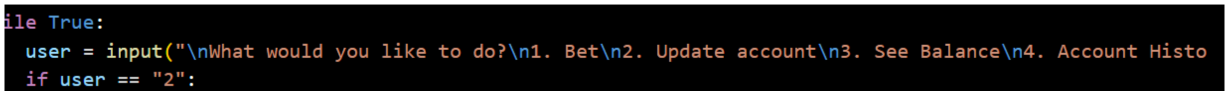
This is a check user function



```
def check_user(connection, name):
    with connection:
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM user WHERE name = ?", (name,)) #sql statment
        result = cursor.fetchone()
        return result is not None
```

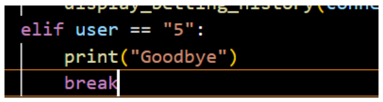
Name:Daniel Mark:

If your name matches with the person in our database, we will grant you access to the home screen of our app. Our original home screen login was simple, with just "bet" and "exit" options. However, we later added more features to the home screen, including the ability to update your account, view your balance, and see your account history. As this may help the user get satisfied with the features.



```
ile True:
  user = input("\nWhat would you like to do?\n1. Bet\n2. Update account\n3. See Balance\n4. Account Histo
  if user == "2":
```

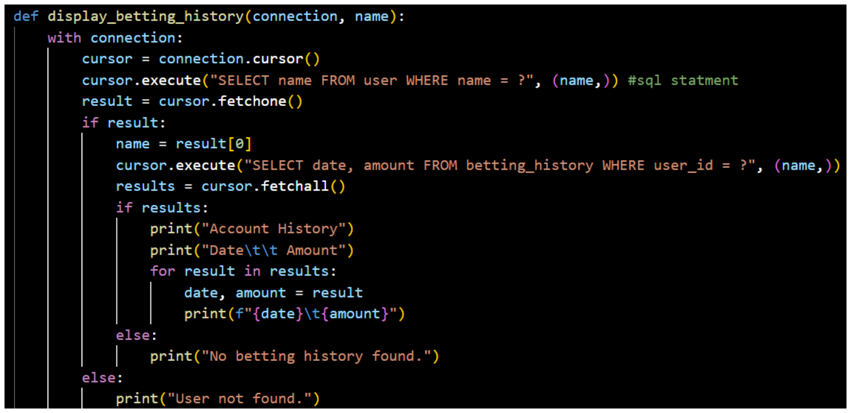
The function of the exit button is to exit the program.



```
        display_betting_history(conne
    elif user == "5":
        print("Goodbye")
        break
```

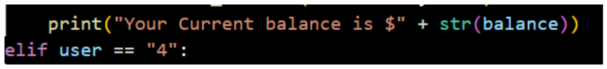
The function of the account history is to view the person's account history for example when they placed a bet or updated the account the data and how much money they put in it will show in the database. For example, Andrew Smith updates his account and adds 200 dollars. The program should update what date he placed and the amount he placed.

The function for the account history



```
def display_betting_history(connection, name):
    with connection:
        cursor = connection.cursor()
        cursor.execute("SELECT name FROM user WHERE name = ?", (name,)) #sql statment
        result = cursor.fetchone()
        if result:
            name = result[0]
            cursor.execute("SELECT date, amount FROM betting_history WHERE user_id = ?", (name,))
            results = cursor.fetchall()
            if results:
                print("Account History")
                print("Date\t\t Amount")
                for result in results:
                    date, amount = result
                    print(f"{date}\t{amount}")
            else:
                print("No betting history found.")
        else:
            print("User not found.")
```

This should output the history

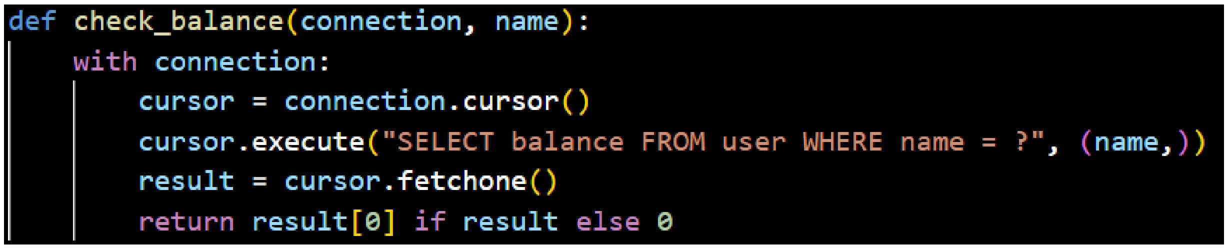


```
    print("Your Current balance is $" + str(balance))
elif user == "4":
```

The function for the show balance. It just shows the amount that the user has in its balance it works by connecting to the user id and then printing out whatever uses balance is. But it is connected to the check user's balance and just prints out the user's current balance.

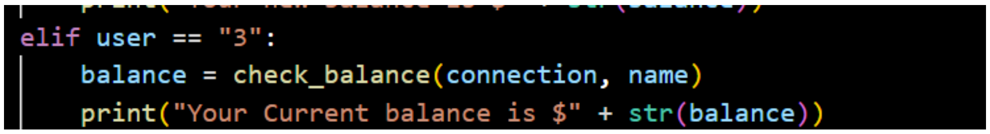
Function for the check user balance

Name:Daniel Mark:



```python
def check_balance(connection, name):
    with connection:
        cursor = connection.cursor()
        cursor.execute("SELECT balance FROM user WHERE name = ?", (name,))
        result = cursor.fetchone()
        return result[0] if result else 0
```

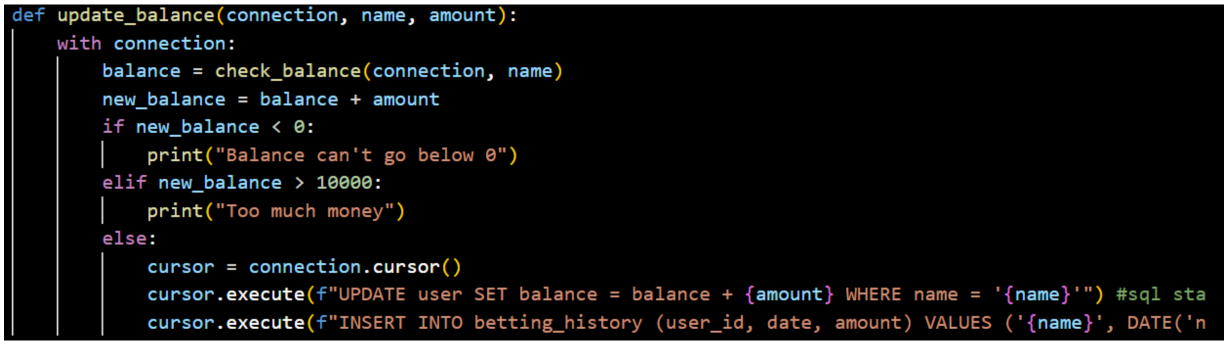
Prints out the user's balance



```python
elif user == "3":
    balance = check_balance(connection, name)
    print("Your Current balance is $" + str(balance))
```

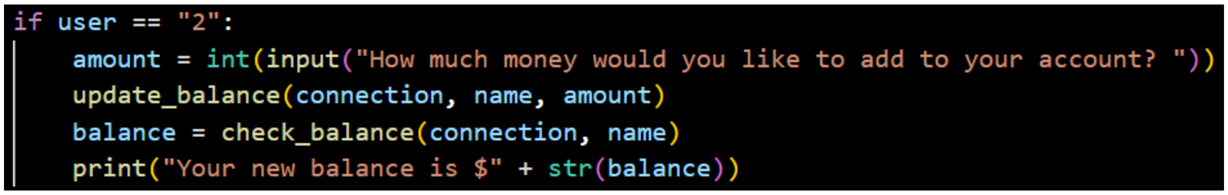
The function for the update account. The updated account is for the users to update their accounts. This is for when the users are running low on cash they could top up their account. The account gets updated when the user tops up the account.

This is the function for updating the user account



```python
def update_balance(connection, name, amount):
    with connection:
        balance = check_balance(connection, name)
        new_balance = balance + amount
        if new_balance < 0:
            print("Balance can't go below 0")
        elif new_balance > 10000:
            print("Too much money")
        else:
            cursor = connection.cursor()
            cursor.execute(f"UPDATE user SET balance = balance + {amount} WHERE name = '{name}'") #sql sta
            cursor.execute(f"INSERT INTO betting_history (user_id, date, amount) VALUES ('{name}', DATE('n
```

It asks the user how much they would like to update their account. It also outputs the new balance that the user inputted.

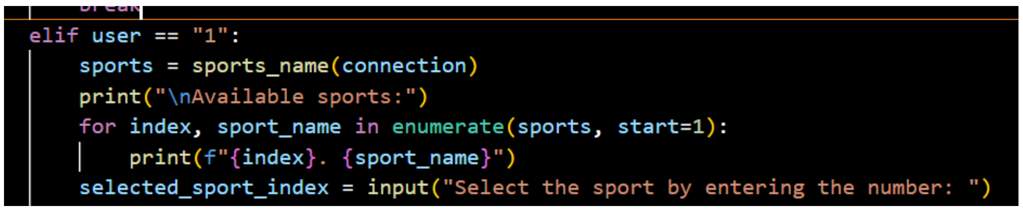


```python
if user == "2":
    amount = int(input("How much money would you like to add to your account? "))
    update_balance(connection, name, amount)
    balance = check_balance(connection, name)
    print("Your new balance is $" + str(balance))
```

The function of the betting system. Firstly when you click bet you will get taken to the available sports that the user can pick. After they have chosen the sport of their choice it will take them to the available teams for that kind of sport.

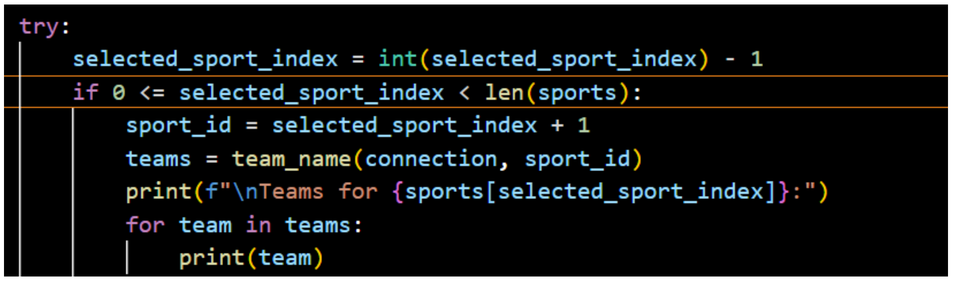
If the user inputs one it takes them to the available sports

Name:Daniel Mark:



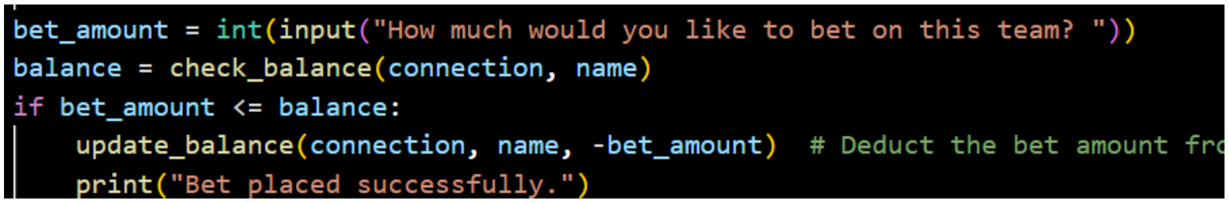
```
elif user == "1":
    sports = sports_name(connection)
    print("\nAvailable sports:")
    for index, sport_name in enumerate(sports, start=1):
        print(f"{index}. {sport_name}")
    selected_sport_index = input("Select the sport by entering the number: ")
```

After they have chosen the sport of their choice it will take them to the available teams for that kind of sport.



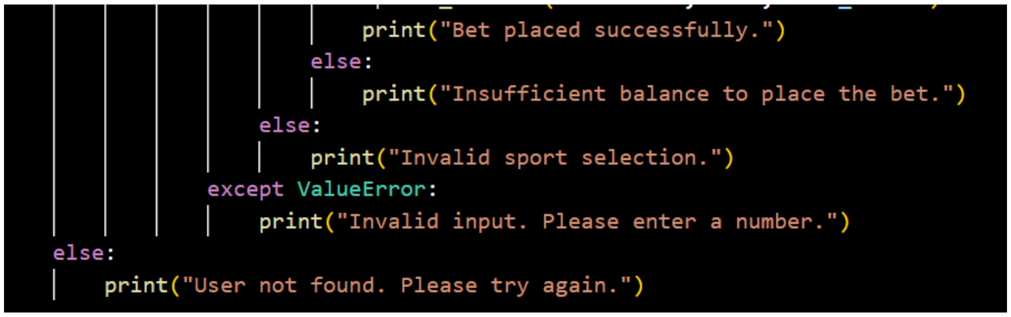
```
try:
    selected_sport_index = int(selected_sport_index) - 1
    if 0 <= selected_sport_index < len(sports):
        sport_id = selected_sport_index + 1
        teams = team_name(connection, sport_id)
        print(f"\nTeams for {sports[selected_sport_index]}:")
        for team in teams:
            print(team)
```

Lastly, the user has to enter the amount that they are willing to bet on that team, the user balance will get deducted depending on how much they put into the bet.



```
bet_amount = int(input("How much would you like to bet on this team? "))
balance = check_balance(connection, name)
if bet_amount <= balance:
    update_balance(connection, name, -bet_amount)  # Deduct the bet amount fro
    print("Bet placed successfully.")
```

The function for all the invalid codes. When a code or an invalid input gets typed in, it will print invalid input. if the user input in an invalid integer it will say invalid integer



```
                    print("Bet placed successfully.")
                else:
                    print("Insufficient balance to place the bet.")
            else:
                print("Invalid sport selection.")
        except ValueError:
            print("Invalid input. Please enter a number.")
else:
    print("User not found. Please try again.")
```

I made some significant improvements to the code by adding more features than originally planned. This should make the program more engaging for the user, as they will have access to more options instead of just the basic bet and exit functions.
Additionally, I addressed the issue of the program crashing due to unexpected files by implementing a system to detect invalid inputs instead. This makes the error message much clearer and easier for users to understand. To ensure that the program is in good working order, I extensively tested it and

Name:Daniel Mark:

resolved any issues that arose through the code.

Name:Daniel Mark:

Teacher Checklists:

AS91879- Develop a digital outcome to manage data

Credits: 4
NZQA: https://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2019/as91879.pdf

Achieved- Develop a digital outcome to manage data	**Evidence**	
using appropriate tools and techniques to structure, organise, query and present data for a purpose and end user		✓
applying appropriate data integrity and testing procedures	Hmm. Some strings in integer fields.	✓
describing relevant implications.		✓
Merit- Develop an informed digital outcome to manage data		
using information from testing procedures to improve the quality and functionality of the outcome		✓
structuring, organising and querying the data logically	See A criteria comment	-
addressing relevant implications.		✓
Excellence- Develop a refined digital outcome to manage data		
iterative improvement throughout the development and testing process		
presenting the data effectively for the purpose and to meet end-user requirements.		

Name:Daniel Mark:

Develop a computer program

Credits: 4 (Internal)
NZQA: http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2018/as91883.pdf

Achieved **Develop a computer program**	**Evidence**	
Wrote a program that performs a specific task using a suitable programming language		✓
Set out the program code clearly		✓
Documented the program with comments		✓
Tested and debugged to ensure that it works on a sample of expected cases		✓
Merit **Develop an informed computer program**		
Documented the program with variable names and comments that describe code function and behaviour		✓
Following conventions of the chosen programming language		✓
Tested and debugged the program in an organised way to ensure it works on expected and relevant boundary cases	Some bugs. Breaks with data check for bet number on integer conversion. And the comments haven;t been fixed.	
Excellence **Develop a refined computer program**		
Ensured the program is a well structured logical solution to the task		
Making the program flexible and robust		
Comprehensively tested and debugged the program		

Comments:

Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Achievement Standard.