

SPL-2 Final Project Report Report, 2024

ShotMaster

SE 505: Software Project Lab II

Submitted by

Md. Rayhan Islam Sefat

BSSE Roll No. : 1332

and

Reza Abdullah

BSSE Roll No. : 1335

Supervised by

Dr. AHMEDUL KABIR

(Associate Professor)

Institute of Information Technology



Date of submission: 13th June 2024

Signature of supervisor:

Table of Contents

1. Introduction	3
2. Background of the Project	4
2.1. The Evolution of Cricket Training	4
2.2. Technological Advancements in Sports Analytics	4
2.3. The Need for Shot Analysis	4
2.4. Existing Solutions and Their Limitations	4
3. Description of the Project	5
3.1. Sensor Integration	5
3.2. Shot Analysis	5
3.3. Session Analysis	5
3.4. Graph Visualization	5
3.5. Player-Coach Connection	5
4. Implementation	6
4.1. Frontend	6
4.1.1. Architecture	6
4.1.2. Key Components	6
4.1.2.1 Firebase options	6
4.1.2.2. Widgets	6
4.1.3. UI/UX Design	7
4.1.3.1. Themes	7
4.1.3.2. Interactions with Backend	7
4.2. Backend	7
4.2.1. Connecting with the database	7
4.2.2. Getting Data from Sensor	7
4.2.3. Flask	7
5. User Manual	9
5.1. Register and Login	9
5.1.1. Register	9
5.1.2. Login	10
5.2. Player Profile	11
5.3. Practice Sessions	13
5.3.1. Session Summary	14
5.3.2. Shot by Shot Data	15
5.4. Connect Your Coach	16
6. Challenges Faced	17
6.1. Connecting the sensor	17
6.2. Scalability and Accessibility	17
6.3. Data Privacy and Security	17

6.4. Validation and Accuracy	17
6.5. User Interface and Experience	17
7. Conclusion and Future Work	18
Appendix	18
References	26

1. Introduction

Cricket, often referred to as a gentleman's game, is a sport that demands not only physical prowess but also a deep understanding of technique and strategy. Among the numerous aspects that contribute to a player's success, the proficiency in executing various cricket shots stands out as a critical element. The ability to analyze and refine these shots can significantly enhance a player's performance, offering a competitive edge in both amateur and professional settings.

The advent of advanced technology and data analytics has opened new avenues for sports training and performance enhancement. In this context, we introduce **ShotMaster**, an innovative software designed to analyze cricket shots with precision and detail. ShotMaster leverages state-of-the-art computer vision and machine learning algorithms to provide comprehensive feedback on shot technique, biomechanics, and performance metrics.

This report details the development and functionality of ShotMaster, highlighting its key features, technological underpinnings, and potential impact on cricket training methodologies. By integrating high-resolution video analysis, motion capture, and data analytics, ShotMaster aims to revolutionize the way cricket shots are analyzed and improved. Through this project, we seek to bridge the gap between traditional coaching methods and modern technological solutions, offering players and coaches a powerful tool to elevate their game to the next level.

2. Background of the Project

For implementing this project, prior studies were necessary. These are given below:

2.1. The Evolution of Cricket Training

Cricket, a sport with a rich history dating back to the 16th century, has evolved significantly in terms of playing techniques and training methodologies. Traditionally, coaching relied heavily on the expertise and subjective judgment of experienced players and coaches. This method, while effective to an extent, often lacked the precision and objectivity needed to identify subtle flaws in a player's technique. As the game progressed, the demand for more accurate and detailed analysis grew, leading to the incorporation of technology in cricket training.

2.2. Technological Advancements in Sports Analytics

The past few decades have seen a rapid advancement in sports analytics, driven by the development of high-speed cameras, motion capture technology, and sophisticated data analysis tools. In cricket, these technologies have been employed to analyze various aspects of the game, from player fitness and injury prevention to match strategies and player performance. High-speed video analysis, in particular, has become a cornerstone in the assessment of bowling and batting techniques, allowing coaches to break down each movement frame by frame.

2.3. The Need for Shot Analysis

Among the various facets of cricket, shot selection and execution are crucial for a batsman's success. A well-executed shot not only requires impeccable timing but also proper body mechanics and technique. Despite its importance, analyzing and refining cricket shots has remained a complex task due to the dynamic nature of the sport. Traditional video analysis provides some insights, but it often falls short in delivering comprehensive, actionable feedback. There is a pressing need for a more integrated and precise solution that can assist players in perfecting their shot techniques.

2.4. Existing Solutions and Their Limitations

Several tools and software have been developed to aid in cricket training, including video analysis software and biomechanical modeling systems. However, many of these solutions are either too simplistic, providing only basic feedback, or too complex and expensive, making them inaccessible to a broader audience. Additionally, most existing tools do not integrate real-time analysis with comprehensive biomechanical insights, limiting their effectiveness in practical coaching scenarios.

3. Description of the Project

There are 4 main features of this application:

1. Sensor Integration
2. Shot Analysis
3. Session Analysis
4. Graph Visualization

3.1. Sensor Integration

An MPU6050 sensor will be connected to the cricket bat which will read a 3-axis gyroscope and a 3-axis accelerometer data in every single interval of time. It will detect the movements and rotations of the bat while a shot is being attempted.

3.2. Shot Analysis

For each shot 5 attributes will be calculated and saved to the database:

1. Bat speed
2. Impact speed
3. Bat lift angle
4. Bat face angle

3.3. Session Analysis

Players will conduct their practices in session by session modules. A practice session will consist of multiple shots played within a short period of time. The application will analyze each of the shots in a session. After the practice session is completed, it will calculate the average(mean), maximum value, minimum value of the shot parameters and display it to the frontend user interface.

3.4. Graph Visualization

The application will display a line-diagram for each player. It will be for average value each of the attributes based on session.

3.5. Player-Coach Connection

A player can be connected to a coach. The assigned coach will see the profile's details of the player. He can also access the session data of the player and provide feedback to the player. A player and their coach can exchange text messages among themselves.

4. Implementation

4.1. Frontend

4.1.1. Architecture

The application follows the **Model-View-Controller (MVC)** architecture pattern, which is an architectural/design pattern that separates an application into three main logical components Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. It was traditionally used for desktop graphical user interfaces (GUIs). Nowadays, MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. It is also used for designing mobile apps.

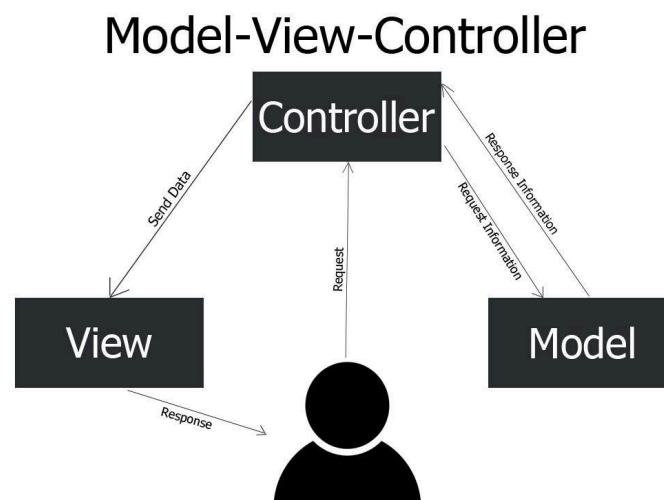


Fig: MVC architecture (source: medium.com)

4.1.2. Key Components

4.1.2.1 Firebase options

We have used Firebase, which is a comprehensive platform offered by Google for developing web and mobile applications. It provides various tools and services to help developers build high-quality apps, grow their user base, and earn more profit. It will make our application work according to the operating system of the device.

4.1.2.2. Widgets

Widgets are the building blocks of a Flutter application. We have utilized both stateless and stateful widgets to create a responsive and interactive user interface.

- **Stateless Widgets:** Used for components that do not require mutable state. Examples include buttons, labels, and static screens.
- **Stateful Widgets:** Used for components that maintain and update state. Examples include forms, dynamic lists, and interactive elements.

4.1.3. UI/UX Design

The app's design adheres to Material Design principles, ensuring a consistent and intuitive user experience across different devices.

4.1.3.1. Themes

We implemented custom themes to maintain a consistent look and feel. The ThemeData class allows customization of colors, fonts, and other visual properties.

4.1.3.2. Interactions with Backend

The frontend interacts with the backend through RESTful APIs. We used the http package for making network requests.

4.2. Backend

4.2.1. Connecting with the database

We have used MongoDB, which is a no SQL database, in our application. MongoDB is built on a scale-out architecture that has become popular with developers of all kinds for developing scalable applications with evolving data schemas. As a document database, MongoDB makes it easy for developers to store structured or unstructured data. It uses a JSON-like format to store documents.

4.2.2. Getting Data from Sensor

The MPU6050, connected via ESP32 WiFi device, is a popular sensor that includes a 3-axis accelerometer and a 3-axis gyroscope. It's commonly used in applications like motion tracking and stabilization systems. Interfacing the MPU6050 with a microcontroller (such as an Arduino) involves reading data from its registers via the I2C protocol. Below, I'll explain how to get data from the MPU6050 sensor using C++ code with an Arduino.

4.2.3. Flask

Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible

framework for new developers since you can build a web application quickly using only a single Python file.

We have used Flask at our application's backend. It has a rich library that has helped us to connect our backend with database, process data and communicate with the frontend.

5. User Manual

5.1. Register and Login

5.1.1. Register

First the user should register with a name, email and a valid password.

1:04

Register

Enter the credentials below to create your account

Name
Rayhan Sefat

Email
bsse1333@iit.du.ac.bd

Password
asdfghjkl

Re-enter Password
asdfghjkl

☒ I agree to ShotMaster [TERMS OF USE](#) and [PRIVACY POLICY](#)

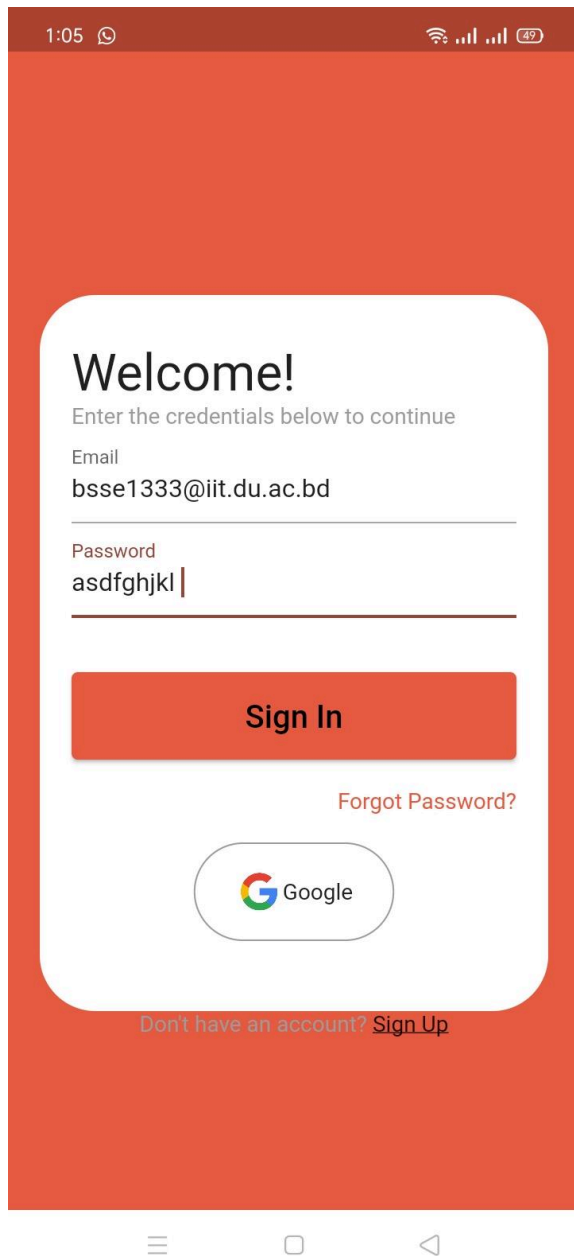
Register

Already have an account? [Sign In](#)

Fig: Register page

5.1.2. Login

Users must login with email and password.



The image shows a mobile application login screen. At the top, a status bar displays the time 1:05, a clock icon, and cellular signal strength. The main background is a solid orange color. A white rounded rectangle in the center contains the login form. The form starts with the heading 'Welcome!' followed by the instruction 'Enter the credentials below to continue'. There are two input fields: 'Email' with the text 'bsse1333@iit.du.ac.bd' and 'Password' with the text 'asdfghjkl |'. Below the password field is an orange 'Sign In' button. Underneath the button is a link 'Forgot Password?'. Below that is a Google sign-in button with the Google logo and the word 'Google'. At the bottom of the white box is a link 'Don't have an account? Sign Up'. The bottom of the screen shows the standard Android navigation bar with three icons: a hamburger menu, a square home button, and a triangle back button.

1:05

Welcome!


Enter the credentials below to continue

Email
bsse1333@iit.du.ac.bd

Password
asdfghjkl |

Sign In

[Forgot Password?](#)

 Google

[Don't have an account? Sign Up](#)

Fig: Login page

5.2. Player Profile

Players can access their own profile.

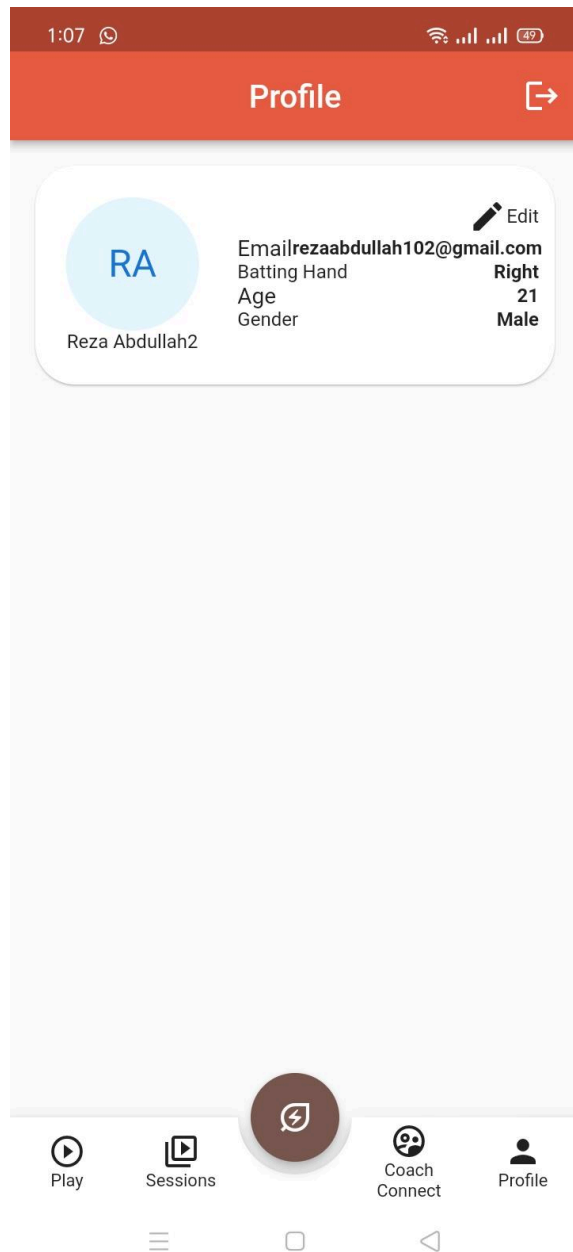


Fig: Player profile page

Profile will contain insights.

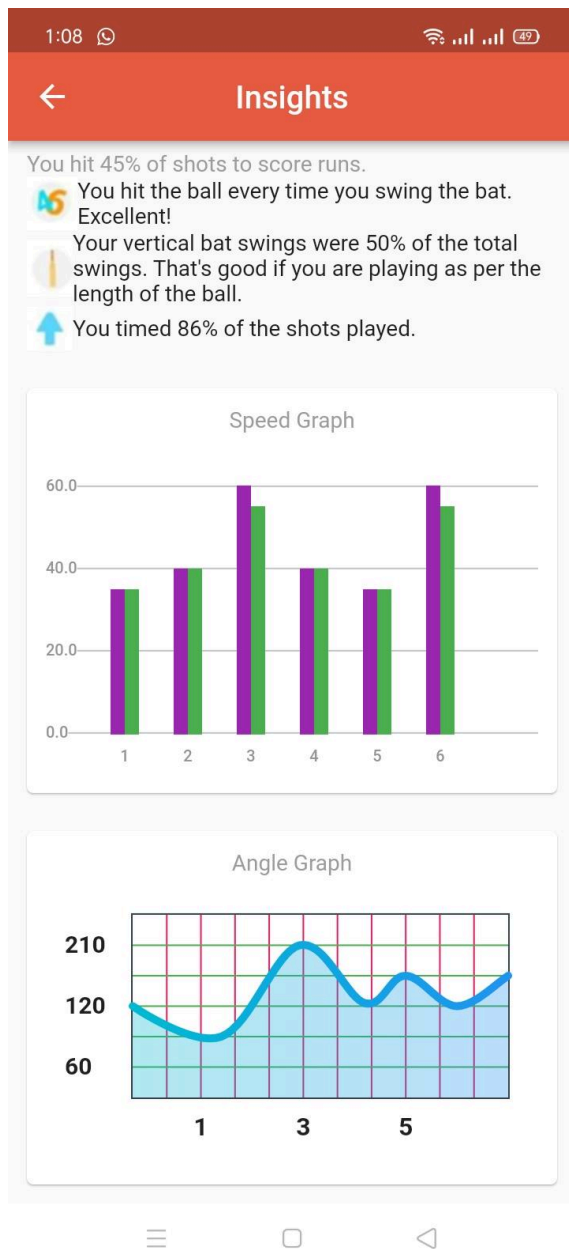


Fig: Profile insights

5.3. Practice Sessions

Select a session.

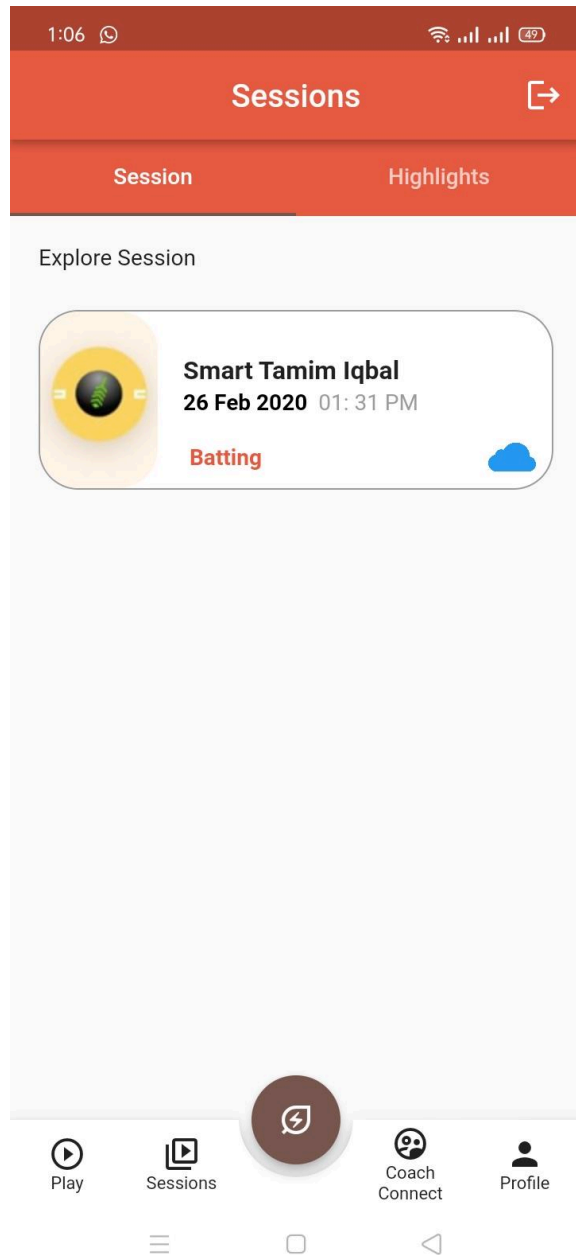


Fig: Select session page.

5.3.1. Session Summary

Session summary for each session is provided.

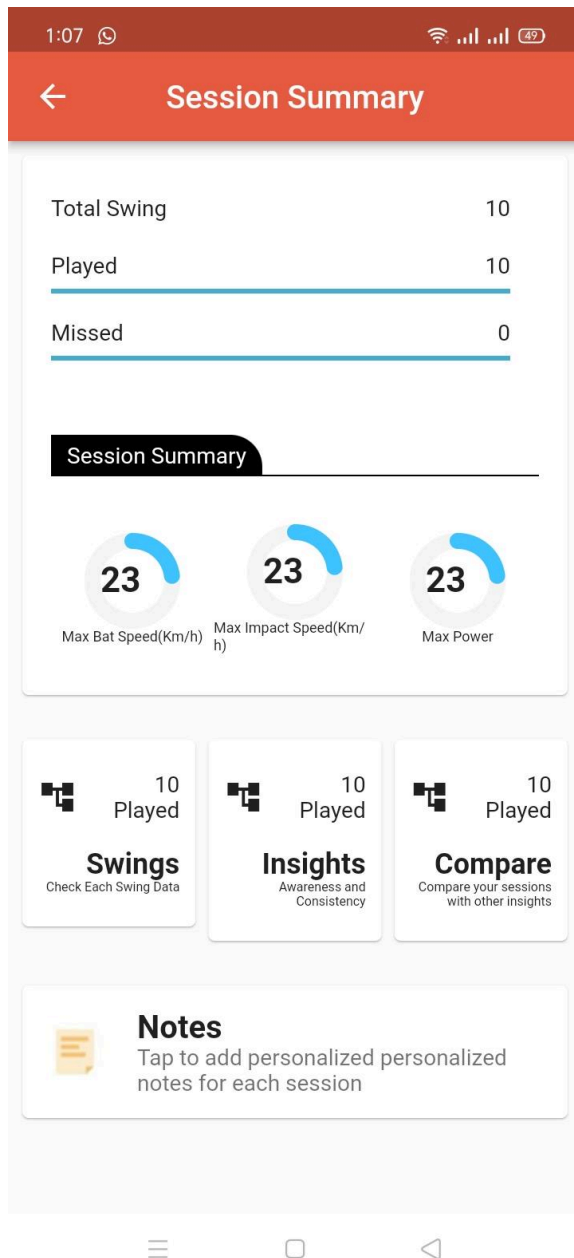


Fig: Session summary page.

5.3.2. Shot by Shot Data

We can see shot by shot data of a session.

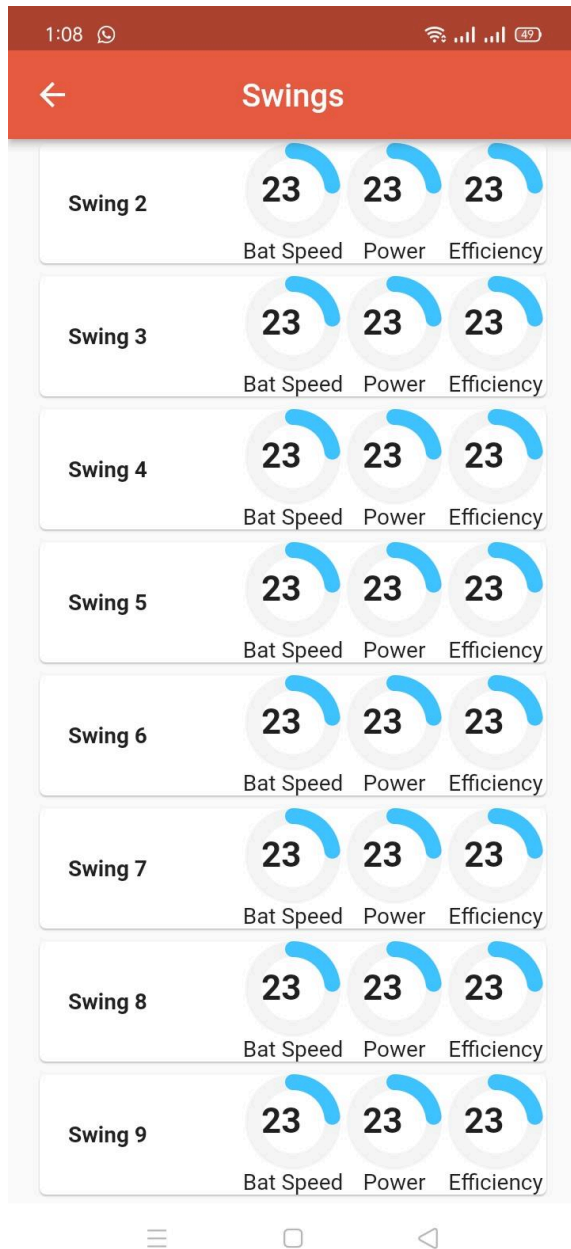


Fig: Shot by shot data

5.4. Connect Your Coach

Players can send connection requests to their coaches.

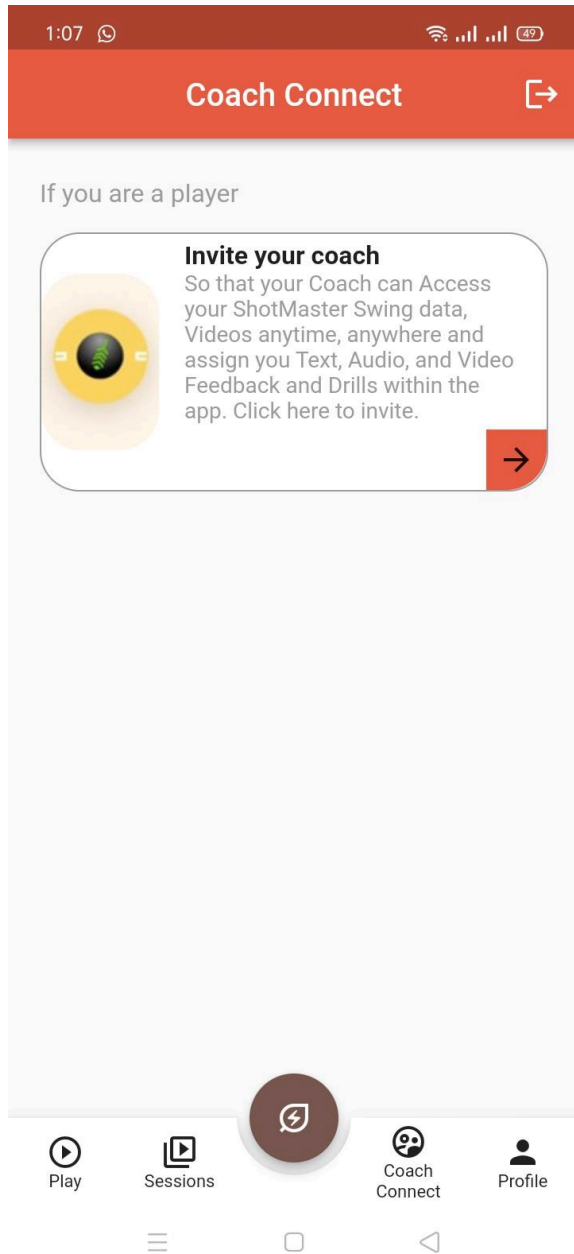


Fig: Coach connect page

5. Challenges Faced

6.1. Connecting the sensor

We had to connect a MPU6050 sensor with the bat so that the sensor could read the motionic and rotational data of the bat while a shot was being played. Connecting the bat and sensor with proper networking support was our goal.

6.2. Scalability and Accessibility

Ensuring ShotMaster was scalable and accessible to a wide range of users, from grassroots players to professional athletes, posed logistical challenges. The software needed to function effectively on various devices and under different conditions.

6.3. Data Privacy and Security

Handling sensitive data, including video footage of players, necessitated stringent data privacy and security measures to protect user information.

6.4. Validation and Accuracy

Validating the accuracy and reliability of the software's analysis was essential to ensure it provided meaningful feedback that users could trust.

6.5. User Interface and Experience

Designing an intuitive and user-friendly interface that catered to both amateur players and professional coaches was crucial. The software needed to present complex data in an accessible manner without overwhelming users.

6. Conclusion and Future Work

ShotMaster represents a significant advancement in cricket training, leveraging computer vision and machine learning to provide detailed, real-time feedback on shot techniques. The development process addressed challenges in data collection, algorithm integration, real-time processing, and user interface design, resulting in a sophisticated yet accessible tool for players at all levels.

By offering precise biomechanical analysis and actionable insights, ShotMaster bridges traditional coaching and modern technology, enhancing training and player performance. Its user-friendly design ensures broad adoption, potentially elevating the overall standard of cricket.

This project highlights the successful integration of sports science and technology, promising ongoing improvements and future enhancements. ShotMaster stands as a testament to innovation in sports, paving the way for advanced performance optimization and excellence in cricket training.

Appendix

```
1. class DefaultFirebaseOptions {  
    static FirebaseOptions get currentPlatform {  
        if (kIsWeb) {  
            throw UnsupportedError(  
                'DefaultFirebaseOptions have not been configured for web - '  
            );  
        }  
        switch (defaultTargetPlatform) {  
            case TargetPlatform.android:  
                return android;  
            case TargetPlatform.iOS:  
                return ios;  
            case TargetPlatform.macOS:  
                return macOS;  
            case TargetPlatform.windows:  
                return windows;  
            case TargetPlatform.linux:  
                return linux;  
        }  
    }  
}
```

Fig: Implementation of the Firebase options

```

@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'ShotMaster',
    debugShowCheckedModeBanner: false,
    theme: ThemeData(
      colorScheme: ColorScheme.fromSeed(seedColor: Color(0xff19DF0E)),
      useMaterial3: true,
    ),
    home: WelcomeScreen(),
  );
}

```

2.

Fig: 'WelcomeScreen' widget code

```

static final ThemeData coreTheme = ThemeData(
  primaryColor: Colors.yellow,
  accentColor: Colors.orange,
  textTheme: TextTheme(
    bodyText1: TextStyle(color: Colors.black),
    bodyText2: TextStyle(color: Colors.grey),
  ),
);

```

3.

Fig: Implementation of core theme

```
4. TextFormField(  
    controller: cpass,  
    decoration: InputDecoration(  
        hintText: 'Re-enter Password',  
        labelText: 'Re-enter Password',  
    ),  
    validator: (value) {  
        if (value == null || value.isEmpty) {  
            return 'Please enter your password';  
        }  
        if (value.length < 8) {  
            return 'Password must contain 8 characters';  
        }  
        return null;  
    },  
),
```

Fig: Validate the password

```
5. # MongoDB connection  
client = MongoClient('mongodb://localhost:27017/')  
db = client['sensor_data']  
collection = db['data']  
  
@app.route('/data', methods=['POST'])  
def receive_data():  
    data = request.json  
    mongo.db.sensor_data.insert_one(data)  
    return jsonify({"message": "Data received and stored successfully"})
```

Fig: Code snippet to connect database

```

@app.route('/create_db_from_csv')
def create_db_from_csv():
    # Drop existing database if it exists
    mongo.db.client.drop_database('myDatabase')

    # Read CSV file and insert data into MongoDB
    with open(csv_file_path, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            # Insert each row into MongoDB
            mongo.db.shots.insert_one(row)
    return 'Data inserted into MongoDB from CSV successfully!'

@socketio.on('connect')
def handle_connect():
    print('Client connected')

@socketio.on('disconnect')
def handle_disconnect():
    print('Client disconnected')

```

6.

Fig: Code snippet to connect database

```

_id: ObjectId('66374dc6811d1b081b2def37')
Accel_X: "-4.3591757327903835"
Accel_Y: "126.68163630168277"
Displacement: "126.75661477469797"

```

```

_id: ObjectId('66374dc7811d1b081b2def38')
Accel_X: "3.0010539577448476"
Accel_Y: "-135.31254488234543"
Displacement: "135.34582050212722"

```

```

_id: ObjectId('66374dc7811d1b081b2def39')
Accel_X: "-4.496955891145196"
Accel_Y: "-158.53623667964948"
Displacement: "158.6000030038863"

```

7.

Fig: Data on MongoDB compass

```

void setup()
{
    Serial.begin(115200);

    // Connect to WiFi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }
    Serial.println("Connected to WiFi");

    // Start the MPU9250 sensor
    Wire.begin();
    imu.begin();
}

```

8.

Fig: Connect sensor via ESP32

```

String payload = "{\"accel_x\": " + String(imu.getAccelX_mss()) + ",";
payload += "\"accel_y\": " + String(imu.getAccelY_mss()) + ",";
payload += "\"accel_z\": " + String(imu.getAccelZ_mss()) + ",";
payload += "\"gyro_x\": " + String(imu.getGyroX_rads()) + ",";
payload += "\"gyro_y\": " + String(imu.getGyroY_rads()) + ",";
payload += "\"gyro_z\": " + String(imu.getGyroZ_rads()) + ",";
payload += "\"mag_x\": " + String(imu.getMagX_uT()) + ",";
payload += "\"mag_y\": " + String(imu.getMagY_uT()) + ",";
payload += "\"mag_z\": " + String(imu.getMagZ_uT()) + "}";

```

9.

Fig: Read sensor data code

10.

```
def calculate_bat_speed(self):
    # Calculate the bat speed using the acceleration data
    speeds = []
    for data in self.sensor_data_list:
        speed = math.sqrt(data.ax**2 + data.ay**2 + data.az**2)
        speeds.append(speed)

    # Calculate the average of every three consecutive speeds
    average_speeds = []
    for i in range(len(speeds) - 2):
        avg_speed = (speeds[i] + speeds[i + 1] + speeds[i + 2]) / 3
        average_speeds.append(avg_speed)

    return max(average_speeds) # Maximum average speed of three consecutive speeds
```

Fig: Backend code snippet

11.

```
def calculate_impact_speed(self):
    # Calculate the impact speed using the maximum acceleration value
    impact_speeds = []
    for data in self.sensor_data_list:
        impact_speed = math.sqrt(data.ax**2 + data.ay**2 + data.az**2)
        impact_speeds.append(impact_speed)
    return max(impact_speeds) # Maximum speed during the shot

def calculate_bat_lift_angle(self):
    # Calculate the bat lift angle using the gyroscope data
    angles = []
    for data in self.sensor_data_list:
        angle = math.atan2(data.gy, data.gz) * (180 / math.pi)
        angles.append(angle)
    return max(angles) - min(angles) # Difference between maximum and minimum angles
```

Fig: Backend code snippet

12.

```

def calculate_bat_face_angle(self):
    # Calculate the bat face angle using the gyroscope data
    angles = []
    for data in self.sensor_data_list:
        angle = math.atan2(data.gx, data.gz) * (180 / math.pi)
        angles.append(angle)
    return max(angles) - min(angles) # Difference between maximum and minimum angles

def calculate_downswing_angle(self):
    # Calculate the downswing angle using the gyroscope data
    angles = []
    for data in self.sensor_data_list:
        angle = math.atan2(data.gx, data.gy) * (180 / math.pi)
        angles.append(angle)
    return max(angles) - min(angles) # Difference between maximum and minimum angles

```

Fig: Backend code snippet

13.

```

@property
def average_bat_speed(self):
    return self.calculate_average_bat_speed()

@property
def average_impact_speed(self):
    return self.calculate_average_impact_speed()

@property
def average_bat_lift_angle(self):
    return self.calculate_average_bat_lift_angle()

@property
def average_bat_face_angle(self):
    return self.calculate_average_bat_face_angle()

@property
def average_downswing_angle(self):
    return self.calculate_average_downswing_angle()

```

Fig: Backend code snippet

Data Flow: ESP32 sends data(mpu6050 and piezo sensor) -> MongoDB stores data -> Python backend retrieves and analyzes data -> Flask/FastAPI serves analysis results and graphs -> Flutter app fetches and displays results.

Technologies: ESP32, MongoDB, Python (Pandas, Matplotlib), Flask/FastAPI, Flutter, HTTP.
Eita add kor

References

- [1] C. M. Bishop, "Pattern Recognition and Machine Learning". New York, NY, USA: Springer, 2006.
- [2] K. P. Murphy, "Machine Learning: A Probabilistic Perspective". Cambridge, MA, USA: MIT Press, 2012.
- [3] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning". Cambridge, MA, USA: MIT Press, 2016.
- [4] 03 March 2024, Why Use MongoDB?, MongoDB,
<https://www.mongodb.com/resources/products/fundamentals/why-use-mongodb#:~:text=MongoDB%20is%20built%20on%20a,like%20format%20to%20store%20documents>.
- [5] 15 April 2024, How To Make a Web Application Using Flask in Python 3, DigitalOcean,
<https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3>
- [6] 10 June 2024, Calculating speed of tennis racket on impact with ball using mpu6050, Arduino Forum,
<https://forum.arduino.cc/t/calculating-speed-of-tennis-racket-on-impact-with-ball-using-mpu6050/640812/3>