



## **Computer Network Assignment Report #3**

*Name: Asad Abbas Naqvi*

*Registration Number: 200901024*

*Submission to: Sir Faran Mehmood*

*Date: 14-05-23*

# Cricket Match Forecast Application

## Overview:

The application for this assignment named “Cricket Match Forecast” is a dynamic and user-friendly cricket match prediction platform developed using Flask, a powerful Python web framework. It empowers cricket enthusiasts to participate in a vibrant community-driven prediction system where they can submit their own predictions and explore predictions made by fellow users. By leveraging web scraping techniques and a SQLite database, the application fetches live player data and facilitates informed decision-making for accurate predictions.

The primary objective of this interactive platform is to provide cricket fans with an engaging environment to showcase their prediction skills and engage in the exciting world of cricket match forecasting. Users can submit their predictions by selecting a player and entering their anticipated score for a specific match. Additionally, they can seamlessly navigate through the interface to browse and interact with predictions submitted by other users, casting votes to express their opinions.

With the help of web scraping capabilities, the application retrieves real-time player data from the popular Cricbuzz website. By offering access to this dynamic information, users can make well-informed predictions based on the players' performance and track the accuracy of their forecasts. The application's intuitive user interface streamlines the prediction submission process, ensures a seamless browsing experience, and encourages active community participation.

## Backend and Frontend:

The backend of the application, powered by Flask, handles the routing and logic for the different endpoints, while the SQLite database efficiently stores and retrieves prediction data. Combining the strengths of HTML, CSS, and JavaScript, along with the Bootstrap framework for responsive design and jQuery for AJAX requests, the frontend delivers an aesthetically pleasing and intuitive user experience with no extra design features for simplicity.

## Usage:

Initialize the application by running it from code runner extension in VSCode or python command using CMD. When the application is running, users can access the web interface by visiting the root URL (<http://127.0.0.1:5000>). The index page (index.html) will be rendered, displaying a form to submit predictions and a list to display existing predictions.

```
PS C:\Users\Asad\Desktop\Semester 6\CN\Assignment 3.1
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 190-894-256
```

Users can fill in their name, the Cricbuzz scorecard match link, select a player, and enter a predicted score in the form. After submitting the form, the prediction will be added to the database, and the list of predictions will be updated to include the new prediction.

### Fetching live match link from Cricbuzz:

The link of live or previous matches from Cricbuzz can be found through their website by visiting this link: [Live Cricket Score, Schedule, Latest News, Stats & Videos | Cricbuzz.com](https://www.cricbuzz.com/live-cricket-scores) (<https://www.cricbuzz.com/live-cricket-scores>)

# Live Cricket Score

[Live](#) [Recent](#) [Upcoming](#)

League

## T20 BLAST 2023

### Hampshire vs Surrey, South Group

May 31 • 11:00 PM at Southampton, The Rose Bowl

<b>HAM</b>	<b>156-4 (20 Ovs)</b>	
<b>SUR</b>	<b>57-2 (7.3 Ovs)</b>	>
Surrey need 100 runs in 75 balls		

[Live Score](#) | [Scorecard](#) | [Full Commentary](#) | [News](#)

### Warwickshire vs Northamptonshire, North Group

May 31 • 10:30 PM at Northampton, County Ground

<b>WARK..</b>	<b>202-6 (20 Ovs)</b>	
<b>NHNT..</b>	<b>159-5 (17.3 Ovs)</b>	>
Northamptonshire need 44 runs in 15 balls		

[Live Score](#) | [Scorecard](#) | [Full Commentary](#) | [News](#)

### Glamorgan vs Middlesex, South Group

May 31 • 8:30 PM at Northwood, Merchant Taylors' School Ground

<b>GLAM</b>	<b>238-3 (20 Ovs)</b>	
<b>MDX</b>	<b>209-5 (20 Ovs)</b>	>
Glamorgan won by 29 runs		

[Live Score](#) | [Scorecard](#) | [Full Commentary](#) | [News](#)

Then click “Scorecard”. This will open another webpage showing all the detailed score information.

## Hampshire vs Surrey, South Group - Live Cricket Score, Commentary

Series: T20 Blast 2023

Venue: The Rose Bowl, Southampton

Date & Time: May 31, 07:00 PM LOCAL

[Commentary](#)

**[Scorecard](#)**

[Squads](#)

[Highlights](#)

[Full Commentary](#)

[Live Blog](#)

[Points Table](#)

Surrey need 80 runs in 60 balls

Surrey Innings		77-2 (10 Ov)				
Batter		R	B	4s	6s	SR
<a href="#">Will Jacks</a>	batting	49	32	2	5	153.12 ▶
<a href="#">Laurie Evans</a>	c Ben McDermott b C Wood	13	11	3	0	118.18 ▶
<a href="#">Sam Curran</a>	c James Fuller b Mason Crane	8	8	1	0	100.00 ▶
<a href="#">Tom Curran</a>	batting	5	9	0	0	55.56 ▶
Extras		2 (b 0, lb 1, w 1, nb 0, p 0)				
Total		77 (2 wkts, 10 Ov)				

Yet to Bat [Jamie Smith \(wk\)](#) , [Sunil Narine](#) , [Jamie Overton](#) , [Tom Lawes](#) , [Gus Atkinson](#) , [Chris Jordan \(c\)](#) , [Sean Abbott](#)

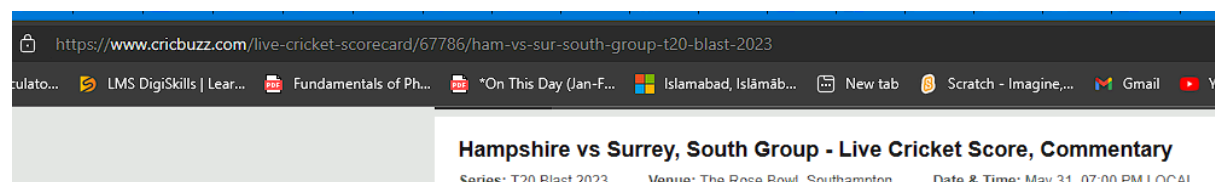
### Fall of Wickets

24-1 ([Laurie Evans](#), 3.3), 54-2 ([Sam Curran](#), 6.3)

Bowler	O	M	R	W	NB	WD	ECO
<a href="#">Chris Wood</a>	2	0	14	1	0	0	7.00 ▶
<a href="#">Nathan Ellis</a>	2	0	24	0	0	1	12.00 ▶
<a href="#">James Fuller</a>	2	0	14	0	0	0	7.00 ▶
<a href="#">Mason Crane</a>	2	0	17	1	0	0	8.50 ▶
<a href="#">Liam Dawson</a>	2	0	7	0	0	0	3.50 ▶

Powerplays	Overs	Runs
Mandatory	0.1-6	52

Now copy the URL of this webpage:



<https://www.cricbuzz.com/live-cricket-scorecard/67786/ham-vs-sur-south-group-t20-blast-2023>

Now paste the URL of the desired match scorecard into the match link textbox.

## Cricket Match Forecast

Name:

Cricbuzz Scorecard Match Link:

Player:

--Select Player--

Predicted Score:

Submit Prediction

## Predictions

Ahmed: Ishant (45)	Upvote	Downvote	Ridiculous	1 Upvotes	1 Downvotes	1 Ridiculous
Asad: Shivam Dube (76)	Upvote	Downvote	Ridiculous	1 Upvotes	1 Downvotes	1 Ridiculous
Ali: Rachin Ravindra (56)	Upvote	Downvote	Ridiculous	1 Upvotes	1 Downvotes	1 Ridiculous
	Upvote	Downvote	Ridiculous	1 Upvotes	1 Downvotes	1 Ridiculous

Users can enter their name and choose from a list of batting players from the given match using the dropdown menu.

## Cricket Match Forecast

Name:

Asad

Cricbuzz Scorecard Match Link:

<https://www.cricbuzz.com/live-cricket-scorecard/67786/ham-vs-sur-south-group-t20-blast-2023>

Player:

--Select Player--

- Select Player--
- Will Jacks (53)
- Laurie Evans (13)
- Sam Curran (8)
- Tom Curran (11)
- Sunil Narine (7)
- Jamie Smith (wk) (4)
- Extras (2)
- Total (98)
- Chris Wood (1)
- Nathan Ellis (0)
- James Fuller (0)
- Mason Crane (1)

As we can see that this player has not got to play their innings yet, we can predict their score

Player:

Nathan Ellis (0)

Predicted Score:

32

Submit Prediction

Users can also view and interact with existing predictions from other players and other matches. Each prediction in the list will show the user's name, predicted score, and vote buttons (upvote, downvote, and ridiculous). Clicking on these buttons will cast a vote for the respective prediction.

Player:

Nathan Ellis (0)

Predicted Score:

32

Submit Prediction

## Predictions

Ahmed: Ishant (45)	Upvote	Downvote	Ridiculous	1 Upvotes	1 Downvotes	1 Ridiculous
Asad: Shivam Dube (76)	Upvote	Downvote	Ridiculous	1 Upvotes	1 Downvotes	1 Ridiculous
Ali: Rachin Ravindra (56)	Upvote	Downvote	Ridiculous	1 Upvotes	1 Downvotes	1 Ridiculous
Muhammad: Mohammad Wasim Jr (87)	Upvote	Downvote	Ridiculous	1 Upvotes	11 Downvotes	1 Ridiculous
Shuja: Ruturaj Gaikwad (34)	Upvote	Downvote	Ridiculous	0 Upvotes	0 Downvotes	3 Ridiculous
Abbas: Yashasvi Jaiswal (69)	Upvote	Downvote	Ridiculous	0 Upvotes	0 Downvotes	17 Ridiculous
Abbas: Ashwin (75)	Upvote	Downvote	Ridiculous	5 Upvotes	5 Downvotes	0 Ridiculous
Ahsan: Robert Yates (56)	Upvote	Downvote	Ridiculous	1 Upvotes	16 Downvotes	1 Ridiculous
AhsanAil: Tom Taylor (3)	Upvote	Downvote	Ridiculous	0 Upvotes	0 Downvotes	0 Ridiculous
Asad: Nathan Ellis (34)	Upvote	Downvote	Ridiculous	0 Upvotes	0 Downvotes	0 Ridiculous

The prediction table updates dynamically. The user who got the most amount of downvotes and ridiculous, but their prediction becomes true wins.

### Code:

The application utilizes the following technologies:

1. Flask: A Python web framework used for backend development, routing, and managing endpoints.
2. HTML, CSS, and JavaScript: Core web development technologies used for creating the frontend interface and implementing interactivity.
3. Bootstrap: A CSS framework that provides pre-designed templates and components for responsive web design.
4. jQuery: A JavaScript library used for simplifying HTML document traversal, event handling, and making AJAX requests.
5. SQLite: A lightweight relational database management system used for storing and retrieving prediction data.
6. BeautifulSoup: A Python library used for web scraping and parsing HTML or XML documents.
7. Requests: A Python library used for making HTTP requests and handling responses.

Here are the code snippets.

app.py

```
from flask import Flask, render_template, request, jsonify
from bs4 import BeautifulSoup
import requests
import sqlite3

app = Flask(__name__)

def init_db():
    conn = sqlite3.connect('predictions.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS predictions (id INTEGER PRIMARY
KEY, user TEXT, match_link TEXT, player TEXT, predicted_score INTEGER,
upvotes INTEGER, downvotes INTEGER, ridiculous_votes INTEGER)''')
    conn.commit()
```



```

        conn.close()

init_db()

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/get_live_players', methods=['GET'])
def get_live_players():
    match_link = request.args.get('match_link')
    if match_link:
        response = requests.get(match_link)
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find the scorecard container
        scorecard_container = soup.select_one('div[id^="innings_"]')

        players = []
        if scorecard_container:
            # Web Scrape data to find all the rows containing player names
            # and their runs from the live scorecard portion of Cricbuzz website
            player_rows = scorecard_container.select('div.cb-col.cb-col-
100.cb-scrd-itms')

            for row in player_rows:
                player_name = row.select_one('div:nth-child(1)')
                player_runs = row.select_one('div.text-right.text-bold')

                if player_name and player_runs:
                    player = {}
                    player['name'] = player_name.text.strip()
                    player['runs'] = player_runs.text.strip()
                    players.append(player)
            return jsonify(players)
        else:
            return jsonify([])

@app.route('/add_prediction', methods=['POST'])
def add_prediction():
    data = request.json
    conn = sqlite3.connect('predictions.db')
    c = conn.cursor()
    c.execute("INSERT INTO predictions (user, match_link, player,
predicted_score, upvotes, downvotes, ridiculous_votes) VALUES (?, ?, ?, ?,
?, ?, ?)", (data['user'], data['match_link'], data['player'],
data['predicted_score'], 0, 0, 0))

```

```

        conn.commit()
        conn.close()
        return jsonify({'status': 'success'})

@app.route('/cast_vote', methods=['POST'])
def cast_vote():
    data = request.json
    conn = sqlite3.connect('predictions.db')
    c = conn.cursor()

    if data['vote_type'] == 'upvote':
        c.execute("UPDATE predictions SET upvotes = upvotes + 1 WHERE id = ?", (data['prediction_id'],))
    elif data['vote_type'] == 'downvote':
        c.execute("UPDATE predictions SET downvotes = downvotes + 1 WHERE id = ?", (data['prediction_id'],))
    elif data['vote_type'] == 'ridiculous':
        c.execute("UPDATE predictions SET ridiculous_votes = ridiculous_votes + 1 WHERE id = ?", (data['prediction_id'],))

    conn.commit()
    conn.close()
    return jsonify({'status': 'success'})

@app.route('/get_predictions', methods=['GET'])
def get_predictions():
    conn = sqlite3.connect('predictions.db')
    c = conn.cursor()
    c.execute("SELECT * FROM predictions")
    predictions = c.fetchall()
    conn.close()
    return jsonify(predictions)

if __name__ == '__main__':
    app.run(debug=True)

```

Index.html:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<title>Cricket Match Forecast</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></
script>
</head>
<body>
  <div class="container">
    <h1 class="mt-5">Cricket Match Forecast</h1>
    <form id="prediction-form" class="mt-4">
      <div class="form-group">
        <label for="user">Name:</label>
        <input type="text" class="form-control" id="user"
name="user" required>
      </div>
      <div class="form-group">
        <label for="match_link">Cricbuzz Scorecard Match
Link:</label>
        <input type="url" class="form-control" id="match_link"
name="match_link" required>
      </div>
      <div class="form-group">
        <label for="player">Player:</label>
        <select class="form-control" id="player" name="player"
required disabled>
          <option value="">--Select Player--</option>
        </select>
      </div>
      <div class="form-group">
        <label for="predicted_score">Predicted Score:</label>
        <input type="number" class="form-control"
id="predicted_score" name="predicted_score" required disabled>
      </div>
      <button type="submit" class="btn btn-primary" id="submit-
prediction" disabled>Submit Prediction</button>
    </form>
    <h2 class="mt-5">Predictions</h2>
    <div class="list-group" id="predictions"></div>
  </div>

  <script>
    $(document).ready(function() {
      function loadPredictions() {
        $.getJSON('/get_predictions', function(predictions) {
          $('#predictions').empty();

```

```

        predictions.forEach(function(prediction) {
            $('#predictions').append('<div class="list-group-item d-flex justify-content-between align-items-center">' + prediction[1] +
            ': ' + prediction[3] + ' (' + prediction[4] + ')' +
            '<div class="btn-group">' +
            '<button class="btn btn-success btn-sm upvote" data-id="' + prediction[0] + '">Upvote</button>' +
            '<button class="btn btn-danger btn-sm downvote" data-id="' + prediction[0] + '">Downvote</button>' +
            '<button class="btn btn-warning btn-sm ridiculous" data-id="' + prediction[0] + '">Ridiculous</button>' +
            '<span class="badge badge-primary badge-pill ml-2">' + prediction[5] + ' Upvotes</span>' +
            '<span class="badge badge-danger badge-pill ml-2">' + prediction[6] + ' Downvotes</span>' +
            '<span class="badge badge-warning badge-pill ml-2">' + prediction[7] + ' Ridiculous</span>' +
            '</div>' +
            '</div>');
        });
    });
}

$('#match_link').on('input', function() {
    let matchLink = $(this).val();
    if (matchLink) {
        $('#player').empty().prop('disabled', false);
        $('#predicted_score').prop('disabled', false);
        $('#submit-prediction').prop('disabled', false);
        $('#player').append('<option value="">--Select Player--
</option>');

        $.ajax({
            type: 'GET',
            url: '/get_live_players',
            data: { match_link: matchLink },
            success: function(players) {
                players.forEach(function(player) {
                    $('#player').append('<option value="' +
player.name + '">' + player.name + ' (' + player.runs + ')</option>');
                });
            },
            error: function() {
                alert('Error loading players');
            }
        });
    } else {
        $('#player').empty().prop('disabled', true);
    }
});

```

```

        $('#predicted_score').prop('disabled', true);
        $('#submit-prediction').prop('disabled', true);
    }
});

$('#prediction-form').submit(function(e) {
    e.preventDefault();

    let predictionData = {
        user: $('#user').val(),
        match_link: $('#match_link').val(),
        player: $('#player').val(),
        predicted_score: $('#predicted_score').val()
    };

    $.ajax({
        type: 'POST',
        url: '/add_prediction',
        contentType: 'application/json',
        data: JSON.stringify(predictionData),
        success: function(response) {
            if (response.status === 'success') {
                loadPredictions();
                $('#prediction-form')[0].reset();
                $('#player').empty().prop('disabled', true);
                $('#predicted_score').prop('disabled', true);
                $('#submit-prediction').prop('disabled', true);
            } else {
                alert('Error adding prediction');
            }
        },
        error: function() {
            alert('Error adding prediction');
        }
    });
});

function castVote(prediction_id, vote_type) {
    $.ajax({
        type: 'POST',
        url: '/cast_vote',
        contentType: 'application/json',
        data: JSON.stringify({ prediction_id: prediction_id,
vote_type: vote_type }),
        success: function(response) {
            if (response.status === 'success') {
                loadPredictions();
            } else {

```

```

        alert('Error casting vote');
    },
    error: function() {
        alert('Error casting vote');
    }
});
}

$(document).on('click', '.upvote', function() {
    let prediction_id = $(this).data('id');
    castVote(prediction_id, 'upvote');
});

$(document).on('click', '.downvote', function() {
    let prediction_id = $(this).data('id');
    castVote(prediction_id, 'downvote');
});

$(document).on('click', '.ridiculous', function() {
    let prediction_id = $(this).data('id');
    castVote(prediction_id, 'ridiculous');
});

loadPredictions();
});
</script>
</body>
</html>

```

Moreover, the predictions file saves the database as we use SQLite.

## Conclusion:

In conclusion, this cricket match prediction platform merges the capabilities of web scraping, database management, and web development technologies. It provides a captivating and interactive space for cricket enthusiasts to demonstrate their prediction prowess, engage with fellow fans, and stay up-to-date with live player statistics. The application offers an immersive and enjoyable experience, amplifying the thrill of cricket by merging data-driven insights with the excitement of forecasting.