DOUGLAS COLLEGE

Winter 2025

Final Report: Sports Management Portal for Injury Prevention, Performance, and Athlete Management Using Machine Learning

Prepared for

CSIS4495 Applied Research Project
Section 002

Presented to

Padmapriya Arasanipalai Kandhadai
CSIS4495 Section 002

Presented by

Juan Carlos Katigbak
300366535

Table of Contents

Introduction Page 01

Summary of the Research Project Page 04

Changes to the Proposal Page 07

Project Completion Timeline Page 10

Technology Stack Page 11

Implemented Features Page 13

Reflections/ Discussions Page 37

Work Date/Hours Logged Page 39

Concluding Remarks Page 45

References Page 46

Appendix A: Installation Guide Page 50

Appendix B: User Guide Page 51

Appendix C: User Feedback Survey Page 61

Appendix D: Jupyter Notebook Page 66

Introduction

I. Domain, Overview of Background, and Context of the Research

Sports injuries are a normal thing in everyday life, arising from interactions between modifiable factors (e.g., training load, strength) and non-modifiable factors (e.g., age, injury history) that fluctuate over time. Sports injuries happen to the so-called “weekend warrior” trying to squeeze in some form of exercise which could be some form of sports activity in an attempt to improve fitness in spite of such a busy schedule. The “weekend warrior” is usually at a high risk for injuries that are just waiting to happen. Then you have the athletes, both amateur and professional, who play their respective sports at a high-level which can be demanding and taxing to their bodies to the point of exhaustion and that too can lead to injuries. It is a fact of life indeed in spite of all the conditioning, training, and nutrition present nowadays to be able to avoid such sports injuries.

With the advent of Artificial intelligence (AI) and Machine Learning (ML), sports injuries can be predicted which can lead to injury prevention. The application of AI and ML in sports has grown significantly in recent years which can lead to advantages for both athletes and coaches in their respective sport. AI has been a boon in performance forecasting and injury prevention in various sports. Examples include optimizing training regimens and evaluating injury risks through advanced algorithms.

II. Framing the Problem

Here are some questions that the researcher has come up with to frame the problem:

1. How can ML improve injury prediction and performance optimization in sports?
2. What methods can provide data-driven insights for athletes?

Injuries significantly impact athletes’ performance thus a reliable injury prediction system can reduce risks and improve overall athlete performance.

III. Relevant Literature and Research

In a study done by researchers Robin Owen, Julian A. Owen, and Seren L. Evans[1], they mentioned that preventing injuries is important to be successful in sports and a lot of investment is made to be able to do this. Despite all of these, injuries still remain a significant challenge. Millions of athletes around the world suffer from injuries every year even if there are scientifically backed prevention strategies. The researchers point out that this sport injury prediction is something that can still be improved and this is where Artificial Intelligence (AI) comes into play. AI offers a promising path forward to provide a comprehensive view of the complexity of sports injuries. Recent advancements of AI and Machine Learning (ML) have become a pivotal part of sports medicine research which provides a better analysis of huge quantities of data to be able to make predictive models of injury. AI and ML can handle imbalanced datasets common in sports injury studies and account for risk factors, assessing their role in predicting injuries.

The same AI and ML technologies are used in one of the major sports leagues in the world which is based in North America and this is the National Basketball Association or the NBA for short. In another study done by researchers Yining Lu, Ayoosh Pareek, Ophelie Z. Lavoie-Gagne, Enrico M. Forlenza, Bhavik H. Patel, Anna K. Reinholz, Brian Forsythe, and Christopher L. Camp[2], they wanted to know about time-loss lower extremity muscle strains (LEMSs) in the NBA and to determine the validity of a ML model in predicting injury risk. Their study involved using various ML models some of which outperformed other ML models by means of achieving high accuracy and reliability. The ML models that did well provided actionable insights which can help physicians assess individual injury risks and inform prevention strategies through a web-based application. The findings highlight the potential of ML to revolutionize injury prevention and management in sports, though further validation and improvement are needed for broader application.

Research Journals and Studies

1. Owen, R., Owen, J. A., & Evans, S. (2024). Artificial intelligence for sport injury prediction. In Artificial Intelligence for Sport Injury Prediction. Springer.

2. Lu, Y., Pareek, A., Lavoie-Gagne, O. Z., Forlenza, E. M., Patel, B. H., Reinholz, A. K., Forsythe, B., & Camp, C. L. (2022). Machine learning for predicting lower extremity muscle strain in National Basketball Association athletes. The Orthopaedic Journal of Sports Medicine, 10(7), 23259671221111742.

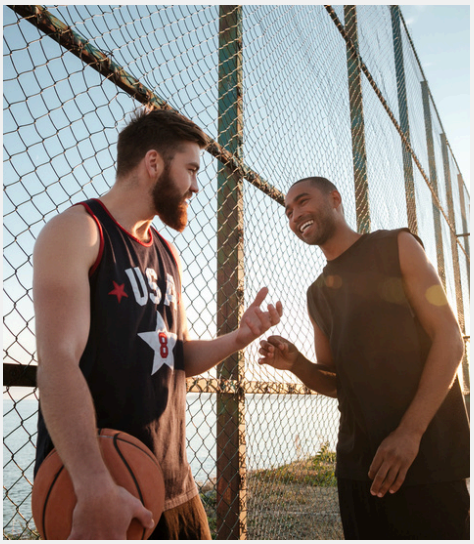
IV. Hypotheses, Assumptions, and Potential Benefit of Research

An integrated sports management portal powered by ML will significantly improve the accuracy of injury prediction and provide insights to optimize athlete performance. The portal will leverage algorithms to analyze both modifiable (e.g., training load, strength) and non-modifiable factors (e.g., age, injury history), enabling athletes to make informed decisions.

Assumptions include the availability of high-quality datasets such as those from the NBA containing relevant variables like athlete demographics, performance metrics, and injury histories. It also assumes that the adoption of the portal by athletes will align with the growing reliance on data-driven approaches in sports.

The potential benefits of this research are immense. First of all, the system is expected to reduce athlete downtime by accurately predicting injury risks and recommending preventive measures tailored to individual needs. Furthermore, the portal will improve training outcomes by enabling people like coaches to adjust regimens based on real-time performance data and risk assessments. Lastly, the research will contribute to the development of a user-friendly and customizable platform, empowering athletes to track their progress and coaches to manage multiple profiles efficiently.

By addressing the gaps in current injury prevention and performance tracking tools, this research aims to advance sports management technology and contribute to safer and more efficient athletic practices. In the long term, the portal could serve as a model for broader adoption across various sports and training environments.



Summary of the Research Project

I. Research Design and Objectives

Design

Develop a web-based sports management portal utilizing machine learning to predict injury risks and deliver data-driven insights tailored specifically for basketball athletes.

Objectives

a. Create a database to store athlete profiles and performance metrics.
b. Train a machine learning model to predict injury risks using historical data.
c. Build interactive dashboards for visualizing injuries and injury risk level.

II. Methodology

Data Collection

Sources were data from the NBA (National Basketball Association) and Pro Sports Transactions websites focused on NBA team being the Los Angeles Lakers with the current Los Angeles Lakers players' historical performance statistics (which includes preseason, regular season, playoffs, a play-in, and an in-season tournament) and synthetic data slightly based from the Lakers' dataset. Features included age, height, weight, basketball position (if guard, forward, or center), games played, minutes played, field goals attempted, three point filed goals attempted, free throws attempted, steals, blocks, fouls, total injuries, and most common injury.

Data Analysis

Use machine learning algorithms/models for injury risk prediction.

Justification

ML models done using NBA data such as in the study of Yining Lu, Ayoosh Pareek, Ophelie Z. Lavoie-Gagne, Enrico M. Forlenza, Bhavik H. Patel, Anna K. Reinholz, Brian Forsythe, and Christopher L. Camp[2] show that this proposed research project can be done as well as make it more holistic by means of including performance and training with injury prediction thus making the application broader.

III. Technology Stack

Operating System/Platform

MacOS and Windows 11 operating systems were used interchangeably in the research and creation of the app.

Programming Languages/Frameworks/Libraries/Database

The **backend** was done by means of ***Django*** **web framework using Python language**. For the **frontend**, it was ***Bootstrap 5*** **using HTML language and** ***Chart.js*** **using JavaScript and JSON data passed from Django views**. For **building the injury risk prediction model**, it was ***scikit-learn*** **library**. **Data processing** was done using ***Pandas and Numpy*** **libraries.** ***Django Auth*** is another which is a **Django built-in user registration, login, and logout system.** ***Joblib*** was used to **save the trained injury risk prediction model.** ***Pillow,*** an **image processing library,** was used so that profile pictures of players can be added in the database. Finally, ***SQLite*** was used for the **database**.

Paid Subscriptions/Services Used in the Research/Creation of the App

ChatGPT Plus (AI tool) - for helping the researcher, who is a beginner in the world of software engineering and data analytics, to be able to do research on how to do the app using AI for ideas generation, algorithm benchmarking, prediction, creating the kanban board, coding (with comments), debugging, report creation (including ReadMe file), survey creation, and data analysis.

Canva Pro (digital branding) - for helping the researcher come up with a nice looking template for Progress Reports, Midterm Report, and Final Report (which includes photos of basketball players) as well as the logo/branding for the app (called PlayAnalytics) and the Powerpoint presentation for the defense.

Amazon Prime (e-commerce) - for helping the researcher get a book called **Phyton crash course 3rd edition: A hands-on, project-based introduction to programming by Eric Matthes from No Starch Press** and this book taught the researcher all about the Django web framework especially how to make a database which served as the foundation of the Profiles page of the app. Two other books were also purchased which were **Complete Conditioning for Basketball written by NBSCA (National Basketball Strength and Conditioning Association) and Strength Training for Basketball written by NSCA (National Strength and Conditioning Association)**; they were supposed to be used as guidance for the Training page of the app which was scrapped due to inability to find a NBA dataset regarding this but will be part of a future feature when the researcher continues further work on the app.

Fantasy Math (sports data science tool) - for helping the researcher with getting ideas on how to build data visualizations based on this website created by **Nathan Braun** who also has a book called **Learn to Code with Basketball 3rd Edition** which the researcher purchased as well to learn all about data analytics. This also gave ideas to the researcher on how to do basketball-related data analytics which were implemented initially during the Proof of Concept for the Midterm Report but was also scrapped since the data visualizations had to be more for the non-data analytics basketball athlete.

DataCamp (e-learning) - for making the researcher learn about how to do data analytics by means of online lessons.

Udemy (e-learning) - another online learning portal for making the researcher learn about how to do data analytics.

Free Subscriptions/Services Used in the Research/Creation of the App

GitHub (online repository for IT portfolios) - online repository used for the submission of all the requirements of the PlayAnalytics app.

YouTube (video sharing platform) - used as a free online platform to access video-based learning for Django, machine learning, data visualization, and full-stack development.

Jupyter Notebook (interactive data science and scientific computing) - used for machine learning model training, data preprocessing, algorithm benchmarking, and visual data analysis for the app.

Douglas College Technology Stack

Office 365 (productivity suite) - **Microsoft Planner** was used to make a kanban board to be able to track the four (4) Phases of the development of the sports management portal. **Microsoft Excel** was used for creating/fixing/editing/cleaning the datasets used via CSV and the creation of Work Date/Hours Logged of all the work the researcher spent time on in building the app. **Microsoft Outlook** was used for email correspondence between the researcher and the advisor/client. **Microsoft Forms** was used for getting a user feedback survey about the app.

IV. Expected Results

The expected results are:

1. Predicting injury risks of basketball athletes
2. Insights and logging notes of basketball performance/injuries/training that basketball athletes and coaches can use

Changes to the Proposal

1. Changes in Technology Stack

The technology stack used during the midterm submission of my initial version of the app was a lighter version compared to what it is now. The initial version looked like this:

Django - Web framework for backend development
Bootstrap 5 - Frontend UI and styling
SQLite - Lightweight database
Matplotlib and Seaborn - Data visualization

The data visualization created was for data analysts and not for the actual users who were supposed to be the basketball athletes/coaches. Upon guidance from the advisor/client, the researcher was able to make use of this technology stack for the final iteration:

Django - Main web framework for routes, views, models, and templates
Bootstrap 5 and Chart.js - Frontend UI and styling via django-bootstrap5 and rendering interactive charts dynamically especially in the Dashboard and Prediction pages
SQLite - Local database used via Django ORM (Object-Relational Mapping)
Django Auth - Django built-in user registration, login, and logout system
scikit-learn - Trained injury risk prediction model
joblib - saved/loaded machine learning model and injury risk label encoder
Pandas - created DataFrames for basketball player stats
Numpy - used indirectly by scikit-learn/Pandas
Pillow - handles image uploads like the profile picture of the basketball athlete in the Profiles page

The sports management portal now had the basketball athletes/coaches in mind so the dashboard/prediction pages with the data visualizations are more easily understood by non-data analytics users. A profiles page is there for the basketball athletes and coaches to check on profiles of a basketball athlete's performance and injury history with a note taking feature to be able to keep a journal of all the activities of the basketball athlete.

The first version of the app, succeeding iterations, and final version called PlayAnalytics all used the following to be able to research and create the app:

ChatGPT Plus (AI tool) - Assisted in every step of the app's creation: ideas generation, algorithm benchmarking, prediction, creating the kanban board, coding (with comments), debugging, report creation (including ReadMe file), survey creation, and data analysis.
Canva Pro (digital branding) - Used to design professional-looking progress reports (which includes photos of basketball players), the PlayAnalytics logo, and the final presentation deck
Amazon Prime (e-commerce) – Enabled access to key physical resources like Python Crash Course (3rd ed.) that supported app functionality and planning
Fantasy Math (sports data science tool) – Served as a source of inspiration for basketball data visualizations and predictive analysis logic, especially during the prototyping and midterm phase
DataCamp (e-learning) – Provided structured online courses for learning foundational data analytics and machine learning concepts
Udemy (e-learning) – Supplemented DataCamp by offering additional practical tutorials and project-based learning for data science
GitHub (online repository for IT portfolios) – Used as the centralized version-controlled repository for the PlayAnalytics app source code and documentation submission.
YouTube (video sharing platform) - for teaching researcher how to integrate Machine Learning data visualization with Django especially from the channel KenBroTech.
Office 365 (productivity suite) – Microsoft Planner was used for kanban-style project tracking; Microsoft Excel for cleaning/preparing CSV datasets and daily work logs; Microsoft Outlook for advisor communication and milestone updates; Microsoft Forms for creating/distributing user feedback survey to gather insights about the app.
Jupyter Notebook (interactive data science and scientific computing) - used for machine learning model training, data preprocessing, algorithm benchmarking, and visual data analysis for the app.

2. Changes in Timelines

The researcher wasted no time with doing PHASE 3: Build Minimum Viable Product and Testing right away after the Midterm submission because the two cards from PHASE 2: Crash Course Coding, Some Implementation, and Data Collecting, which were (1) Finalizing NBA Data composed of Performance and Injuries and (2) Finalizing Machine Learning Algorithms to predict injuries based on NBA Performance data, were supposed to have been finished during PHASE 2 had to be moved to PHASE 3. The reason for this was because the researcher spent a lot of time with finding unique NBA datasets (for statistical performance and injury history) that is different from the other NBA datasets in Kaggle and other sources. The researcher also did a lot of time cleaning the datasets that he found which he thought might be best for training a model based on those datasets. Adding to that was the researcher's lack of experience with machine learning algorithms making him overthink as to which algorithm to use best for predicting basketball injury risk. Other than the researcher having to do a second Proof of Concept as per the suggestion of the advisor/client that the initial app has to be for non-data analytics basketball athletes, the timeline between PHASE 3 and PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client was unchanged even with a last minute requirement of getting a survey regarding user feedback of the app and the researcher having to scrap the minimum viable product that he already had due to having too many errors. The researcher then had to do his process backwards to make the new MVP which is now called PlayAnalytics.

Project Completion Timeline

PHASE	DURATION	MILESTONES
PHASE 1: RESEARCH AND PLANNING	1.5 weeks	1. Determining topic of research for Applied Research Project 2. Researching the topic and coming up with 1st draft of project proposal 3. Finalizing project proposal
PHASE 2: CRASH COURSE CODING, SOME IMPLEMENTATION, AND DATA COLLECTING	3 weeks	1. Crash Courses on Coding with Some Implementation 2. Midterm Report and Implementation
PHASE 3: BUILD MINIMUM VIABLE PRODUCT AND TESTING	5 weeks	1. Finalizing NBA Data composed of Performance and Injuries 2. Finalizing Machine Learning Algorithms to predict injuries based on NBA Performance data 3. Building athlete profiles and injury prediction
PHASE 4: PLAYANALYTICS (SPORTS MANAGEMENT PORTAL) PRESENTATION TO CLIENT	2 weeks	1. Conducting Testing/Refinement 2. Gather user feedback 3. Ready for presentation to client

Technology Stack

This is also discussed in Page 4 - Summary of the Research Project. Just the same, the following was used in the research and creation of the PlayAnalytics app:

Operating System/Platform

MacOS and Windows 11 operating systems were used interchangeably in the research and creation of the app.

Programming Languages/Frameworks/Libraries/Database

The **backend** was done by means of ***Django*** **web framework using Python language**. For the **frontend**, it was ***Bootstrap 5*** **using HTML language and** ***Chart.js*** **using JavaScript and JSON data passed from Django views**. For **building the injury risk prediction model**, it was ***scikit-learn*** **library**. **Data processing** was done using ***Pandas and Numpy*** **libraries.** ***Django Auth*** is another which is a **Django built-in user registration, login, and logout system.** ***Joblib*** was used to **save the trained injury risk prediction model.** ***Pillow,*** an **image processing library,** was used so that profile pictures of players can be added in the database. Finally, ***SQLite*** was used for the **database**.

Paid Subscriptions/Services Used in the Research/Creation of the App

ChatGPT Plus (AI tool) - for helping the researcher, who is a beginner in the world of software engineering and data analytics, to be able to do research on how to do the app using AI for ideas generation, algorithm benchmarking, prediction, creating the kanban board, coding (with comments), debugging, report creation (including ReadMe file), survey creation, and data analysis.

Canva Pro (digital branding) - for helping the researcher come up with a nice looking template for Progress Reports, Midterm Report, and Final Report (which includes photos of basketball players) as well as the logo/branding for the app (called PlayAnalytics) and the Powerpoint presentation for the defense.

Amazon Prime (e-commerce) - for helping the researcher get a book called **Phyton crash course 3rd edition: A hands-on, project-based introduction to programming by Eric Matthes from No Starch Press** and this book taught the researcher all about the Django web framework especially how to make a database which served as the foundation of the Profiles page of the app. Two other books were also purchased which were **Complete Conditioning for Basketball written by NBSCA (National Basketball Strength and Conditioning Association) and Strength Training for Basketball written by NSCA (National Strength and Conditioning Association)**; they were supposed to be used as guidance for the Training page of the app which was scrapped due to inability to find a NBA dataset regarding this but will be part of a future feature when the researcher continues further work on the app.

Fantasy Math (sports data science tool) - for helping the researcher with getting ideas on how to build data visualizations based on this website created by **Nathan Braun** who also has a book called **Learn to Code with Basketball 3rd Edition** which the researcher purchased as well to learn all about data analytics. This also gave ideas to the researcher on how to do basketball-related data analytics which were implemented initially during the Proof of Concept for the Midterm Report but was also scrapped since the data visualizations had to be more for the non-data analytics basketball athlete.

DataCamp (e-learning) - for making the researcher learn about how to do data analytics by means of online lessons.

Udemy (e-learning) - another online learning portal for making the researcher learn about how to do data analytics.

Free Subscriptions/Services Used in the Research/Creation of the App

GitHub (online repository for IT portfolios) - online repository used for the submission of all the requirements of the PlayAnalytics app.

YouTube (video sharing platform) - used as a free online platform to access video-based learning for Django, machine learning, data visualization, and full-stack development.

Jupyter Notebook (interactive data science and scientific computing) - used for machine learning model training, data preprocessing, algorithm benchmarking, and visual data analysis for the app.

Douglas College Technology Stack

Office 365 (productivity suite) - **Microsoft Planner** was used to make a kanban board to be able to track the four (4) Phases of the development of the sports management portal. **Microsoft Excel** was used for creating/fixing/editing/cleaning the datasets used via CSV and the creation of Work Date/Hours Logged of all the work the researcher spent time on in building the app. **Microsoft Outlook** was used for email correspondence between the researcher and the advisor/client. **Microsoft Forms** was used for getting a user feedback survey about the app.

Implemented Features

Feature 1: Responsive Web Design

Feature Description

Responsive Web Design ensures that the PlayAnalytics web application looks and functions well across different devices — from large desktop monitors to tablets and mobile phones. This allows athletes, coaches, or researchers to interact with the platform comfortably on any screen size.

Design & Implementation

This feature was implemented using the Bootstrap 5 mobile-first framework. Key Bootstrap components and utilities were used to ensure flexible layouts and UI elements that adapt to the screen width.

- **Navbar:** The top navigation bar collapses into a hamburger menu on mobile using Bootstrap's navbar-expand-md and navbar-toggler.
- **Grid Layout:** Profile listings, player details, and prediction forms use Bootstrap's grid system (col-md-6, col-sm-12) to reorganize layout structure for smaller screens.
- **Image Responsiveness:** Player profile pictures use class="rounded-circle" and object-fit: cover; in combination with inline style attributes for responsive scaling.
- **Spacing and Readability:** Responsive margin (mb-4, mt-5) and text size classes (fs-4, fw-bold) are used to improve readability on mobile.

Code Implementation

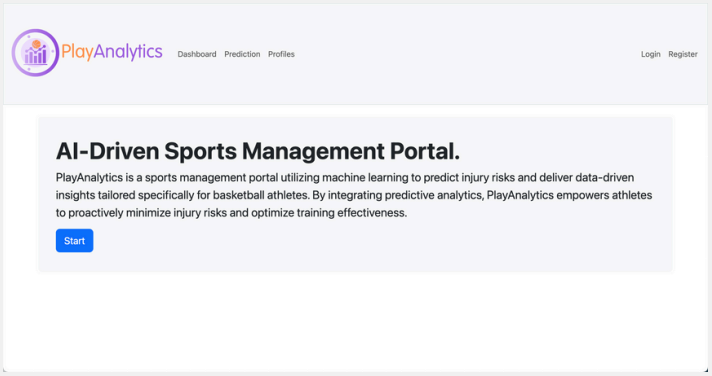
1. Navbar (in base.html):

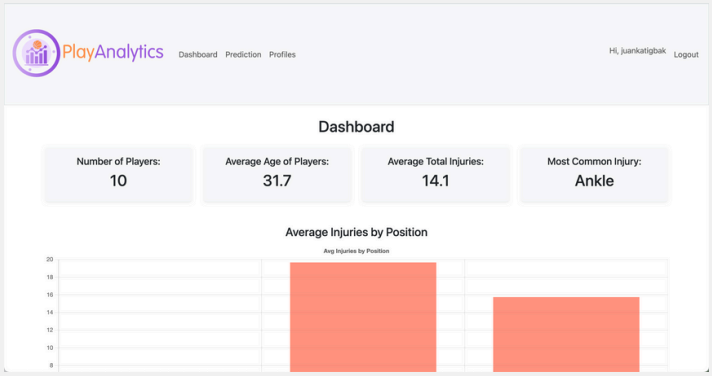
```
<nav class="navbar navbar-expand-md navbar-light bg-light mb-4 border">
  <div class="container-fluid">
    <a class="navbar-brand" href="{% url 'player_profiles:index' %}">
      <img src="{% static 'images/logo.svg' %}" alt="Logo">
    </a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
            data-bs-target="#navbarCollapse" aria-controls="navbarCollapse"
            aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarCollapse">
      <!-- nav links -->
    </div>
  </div>
</nav>
```

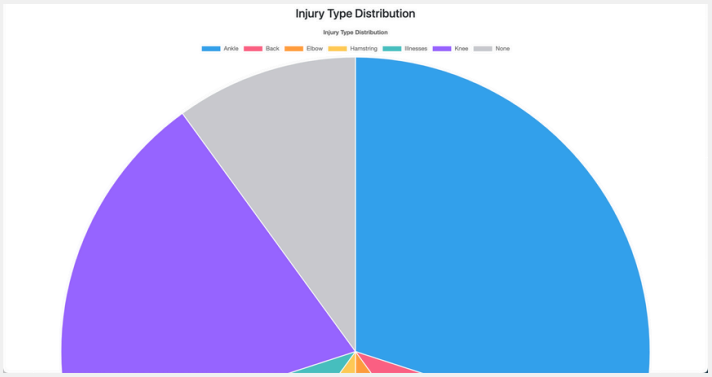
2. Responsive Profile Layout (in profile_list.html):

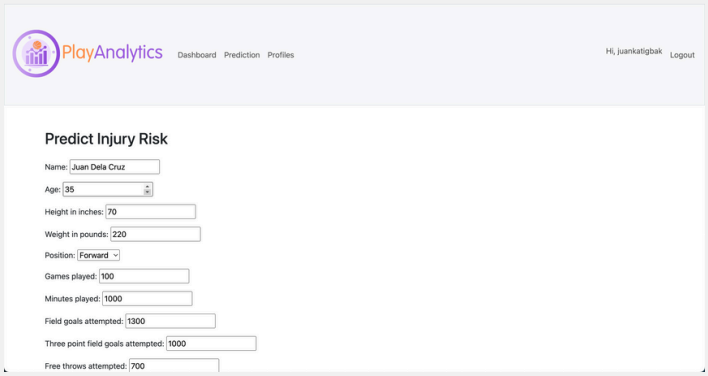
```
<div class="d-flex align-items-center gap-3">
  {% if profile.profile_picture %}
    <img src="{{ profile.profile_picture.url }}" alt="{{ profile.name }}"
         class="rounded-circle" style="width: 50px; height: 50px; object-fit: cover;">
  {% else %}
    <img src="{% static 'images/default_profile.png' %}" alt="Default Profile"
         class="rounded-circle" style="width: 50px; height: 50px; object-fit: cover;">
  {% endif %}
</div>
```

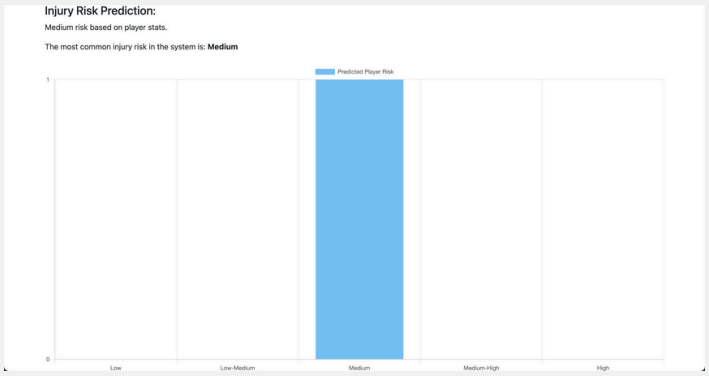
Screenshots
1. Desktop View

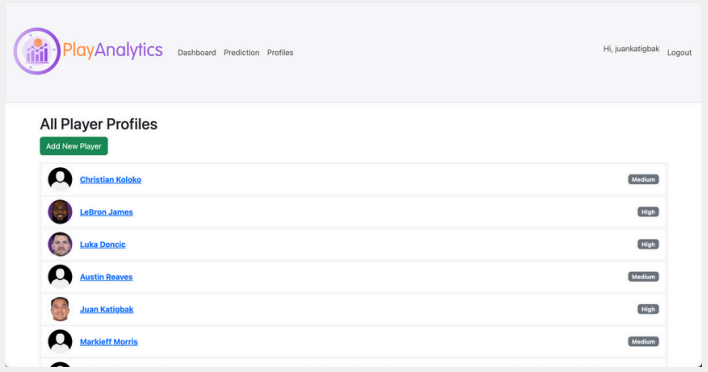


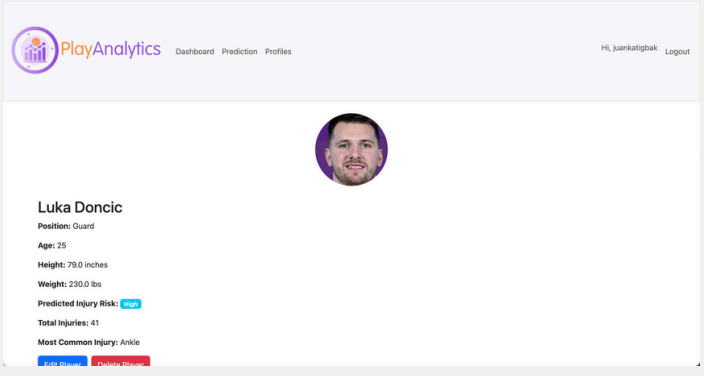


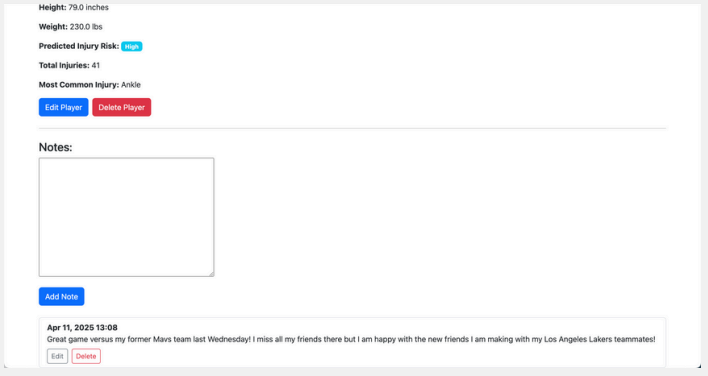




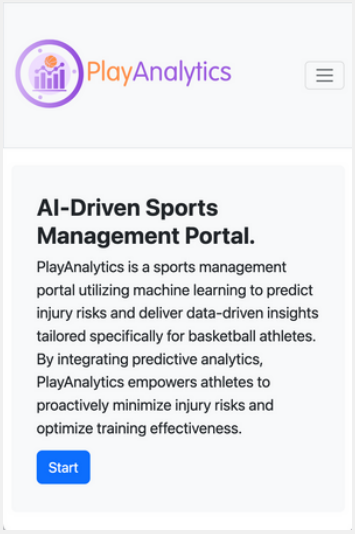


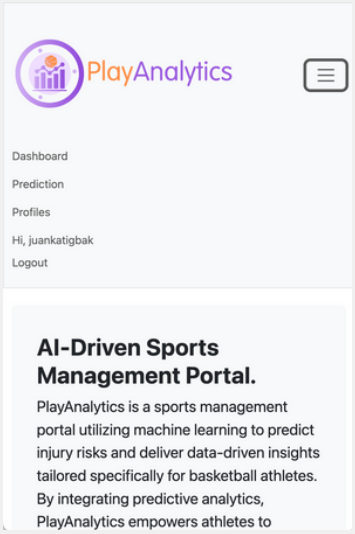


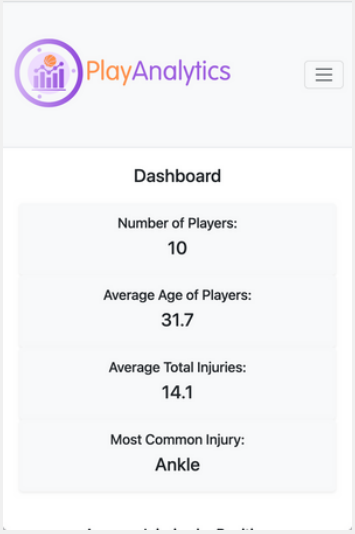


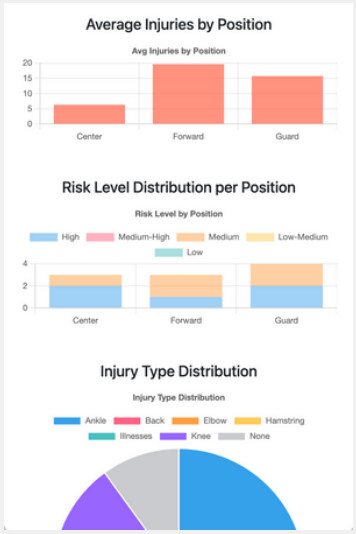


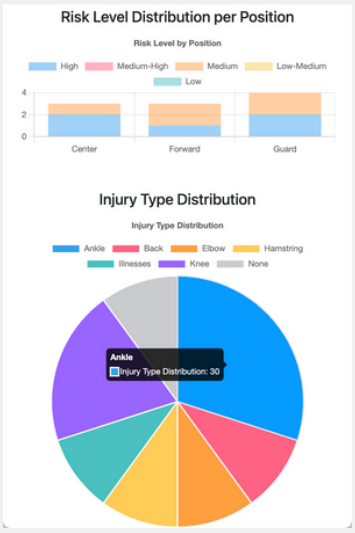
2. Mobile View

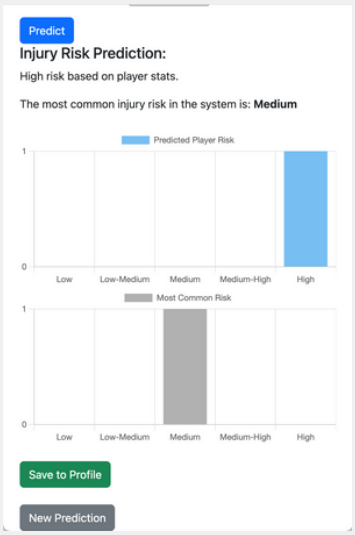


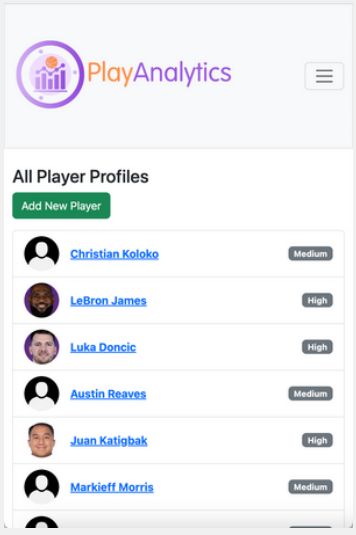


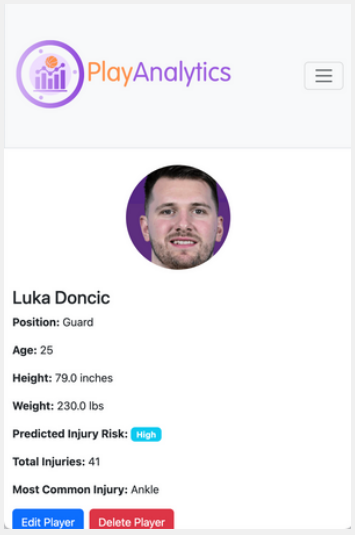


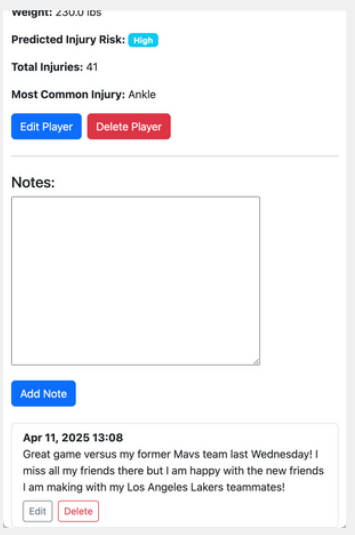












Feature 2: User Registration and Login

Feature Description

The User Authentication System enables secure user registration, login, and logout functionality. It ensures that only registered users can access key features of the PlayAnalytics portal such as adding player data, predicting injuries, and viewing the dashboard. This keeps the system secure and personalized.

Design & Implementation

Django's built-in authentication system (django.contrib.auth) was used as the foundation. A custom registration form was created using UserCreationForm to simplify onboarding. Upon successful login, users are redirected to the Dashboard. Unauthorized users attempting to access restricted pages are redirected to the login page.

- **Registration Page:** Allows new users to create an account using a custom CustomUserCreationForm.
- **Login Page:** Authenticates user credentials and redirects to the dashboard or the requested page.
- **Logout:** Safely logs the user out and returns to the landing page.
- **Access Restriction:** Decorators like @login_required protect all critical views.

Code Implementation

1. Custom User Form (users/forms.py):

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class CustomUserCreationForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'password1', 'password2']
```

2. Views (users/views.py):

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from .forms import CustomUserCreationForm

# Register a new user
def register_view(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('dashboard:dashboard_view')
    else:
        form = CustomUserCreationForm()
    return render(request, 'users/register.html', {'form': form})

# Login existing user
def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            next_url = request.GET.get('next') or 'dashboard:dashboard_view'
            return redirect(next_url)
        else:
            messages.error(request, 'Invalid credentials')
    return render(request, 'users/login.html')

# Logout view
def logout_view(request):
    logout(request)
    return redirect('landing')
```

3. Templates (register.html):

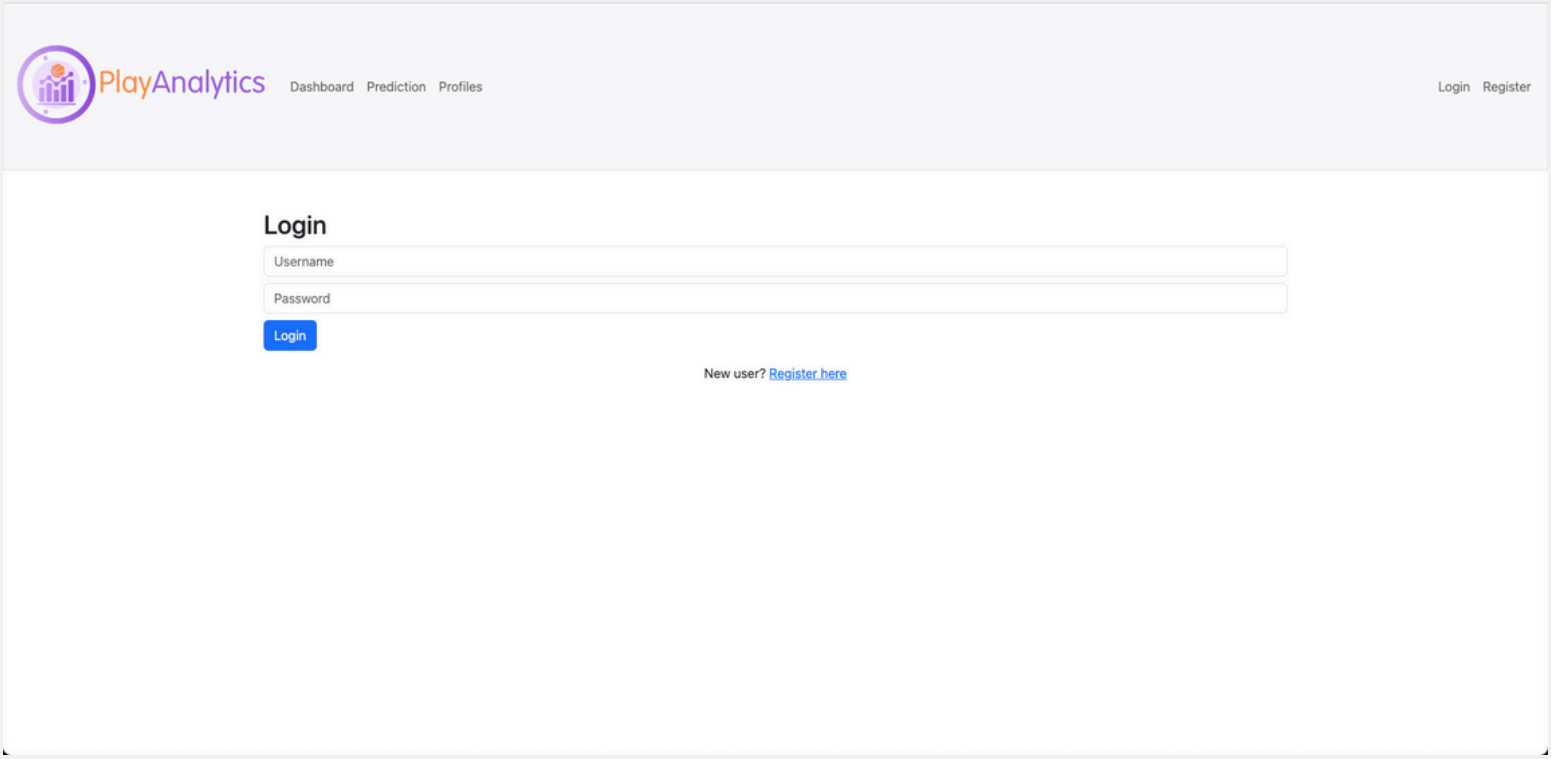
```
<h2>Register</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit" class="btn btn-success">Register</button>
</form>
```

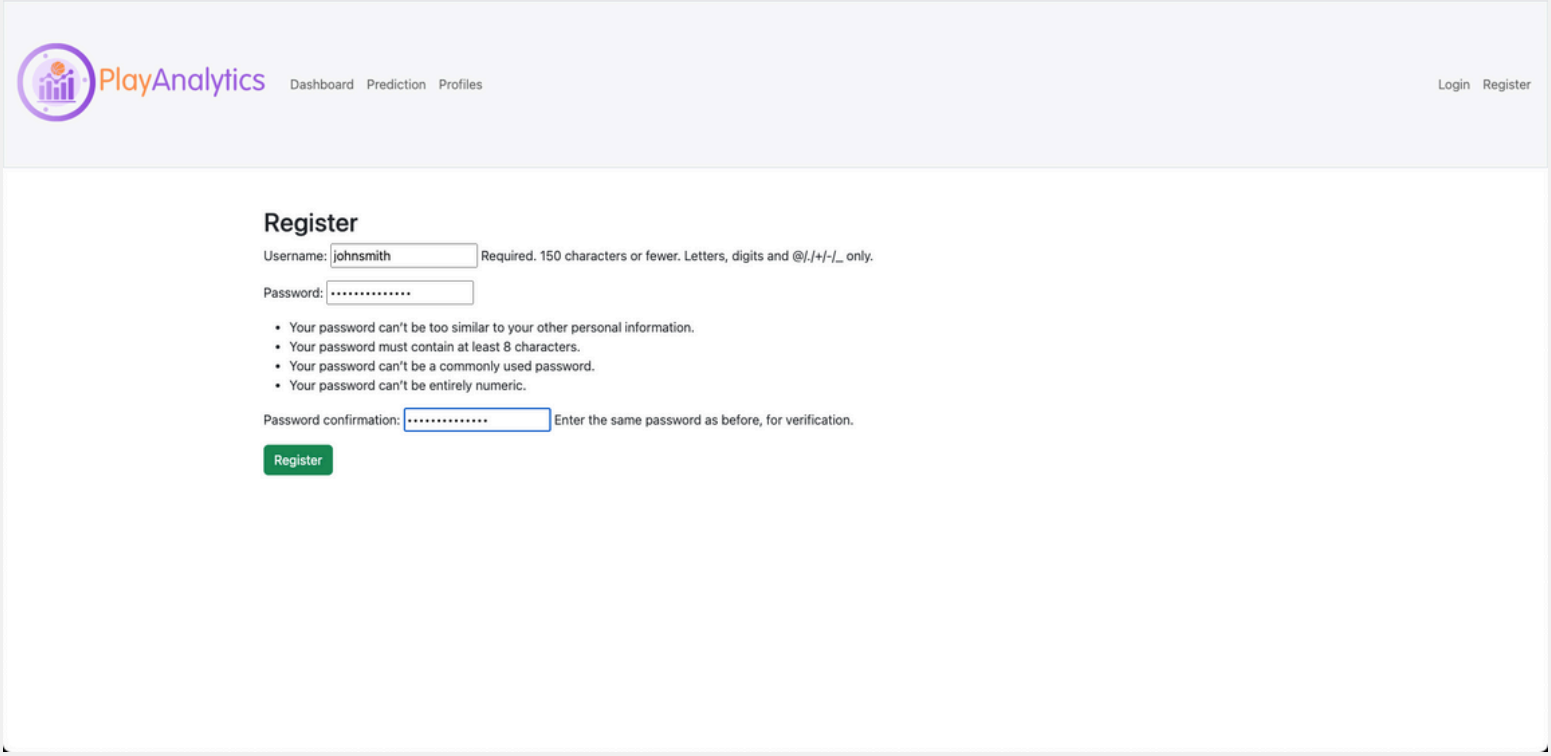
4. Templates (login.html)

```
<h2>Login</h2>
<form method="post">
  {% csrf_token %}
  <input type="text" name="username" class="form-control mb-2" required>
  <input type="password" name="password" class="form-control mb-2" required>
  <button type="submit" class="btn btn-primary">Login</button>
</form>
```

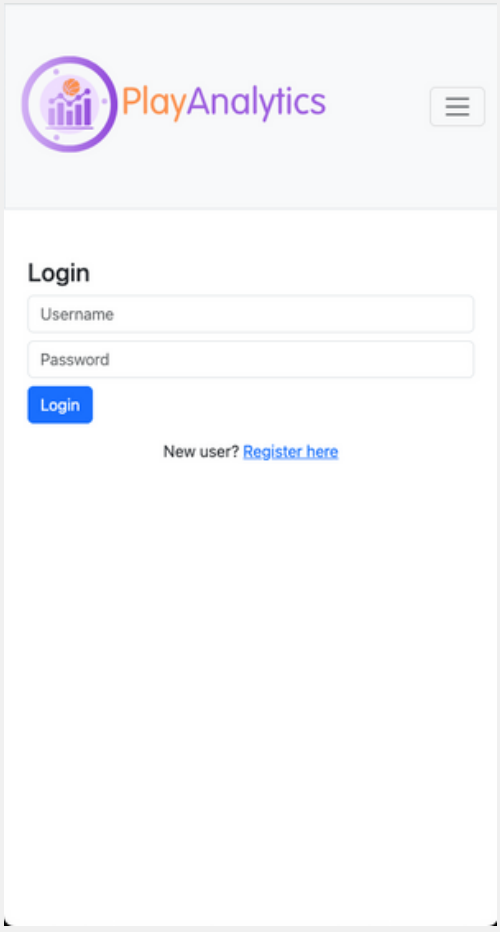
Screenshots

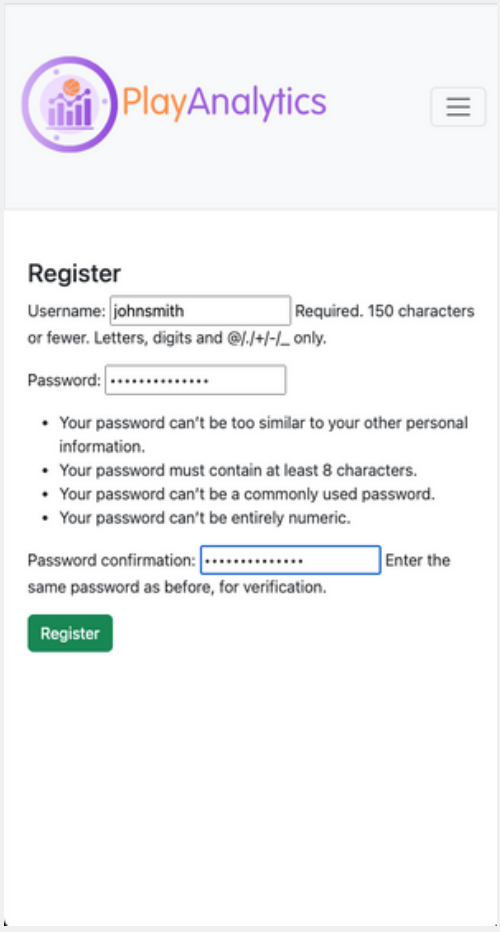
1. Desktop View





2. Mobile View





Feature 3: Dashboard Page

Feature Description

The Dashboard Page of PlayAnalytics provides users with real-time insights into athlete performance and injury trends through dynamic visualizations and summary indicators. It serves as the app's data analysis hub, allowing users to monitor the predicted injury risks of all saved players.

Design & Implementation

This feature was designed using Chart.js for dynamic data visualization and Bootstrap 5 for layout styling and responsiveness. The dashboard updates automatically as new players are added to the system.

- **Summary Indicators:** Top stats like Total Players, High Risk Players, and Most Common Injury appear in prominent card components using Bootstrap.
- **Bar Chart:** Displays the distribution of average injuries by player position.
- **Pie Chart:** Shows the percentage of different injury types across all saved profiles.
- **Real-Time Updates:** Charts and indicators reflect the latest state of the database each time a user returns to the dashboard.

All player data is fetched from the PlayerProfile model using Django ORM (Object-Relational Mapping) queries. Data is passed from dashboard/views.py to dashboard/dashboard.html as JSON using Django's JsonResponse and embedded directly in <script> tags to generate Chart.js charts.

Code Implementation

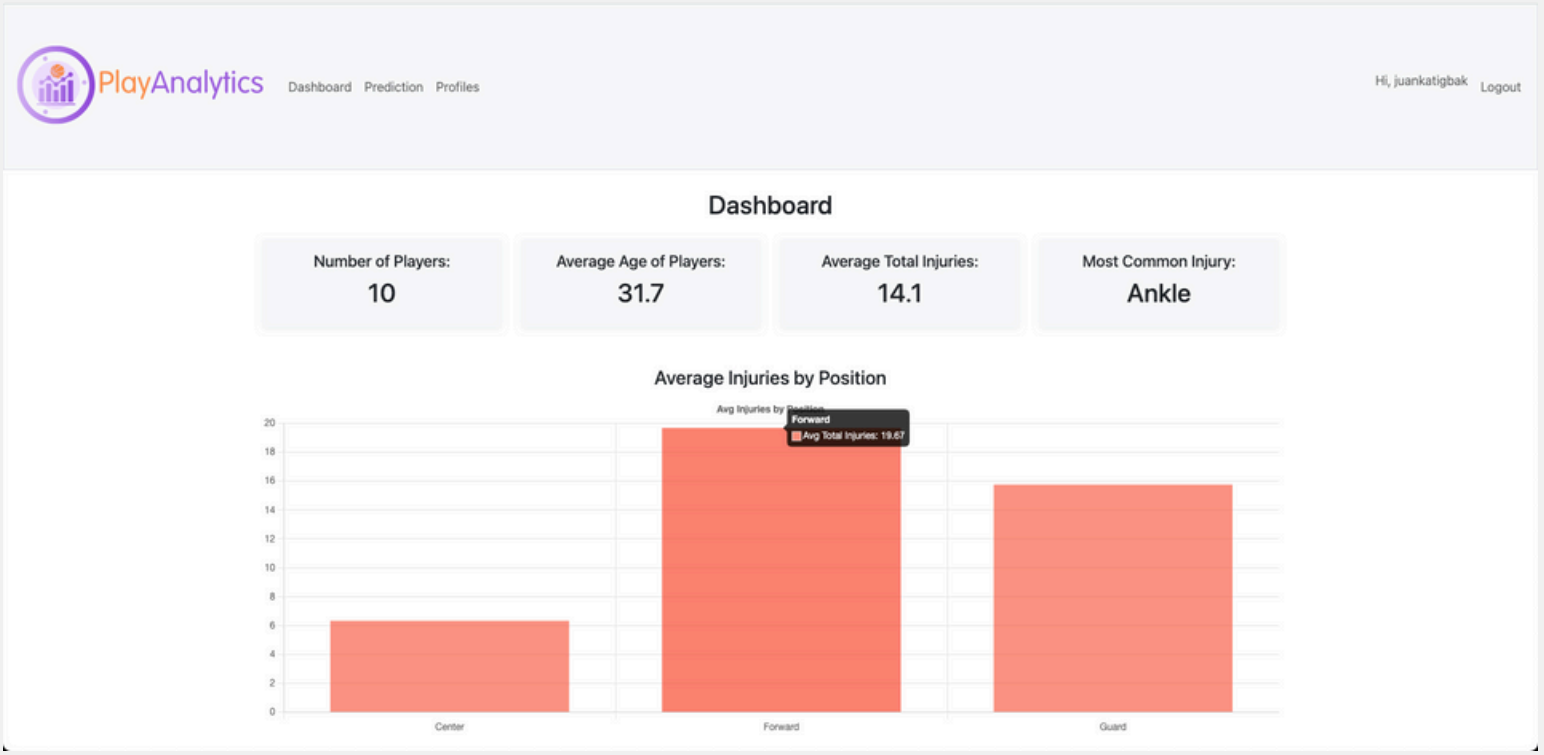
1. Django View (dashboard/views.py):

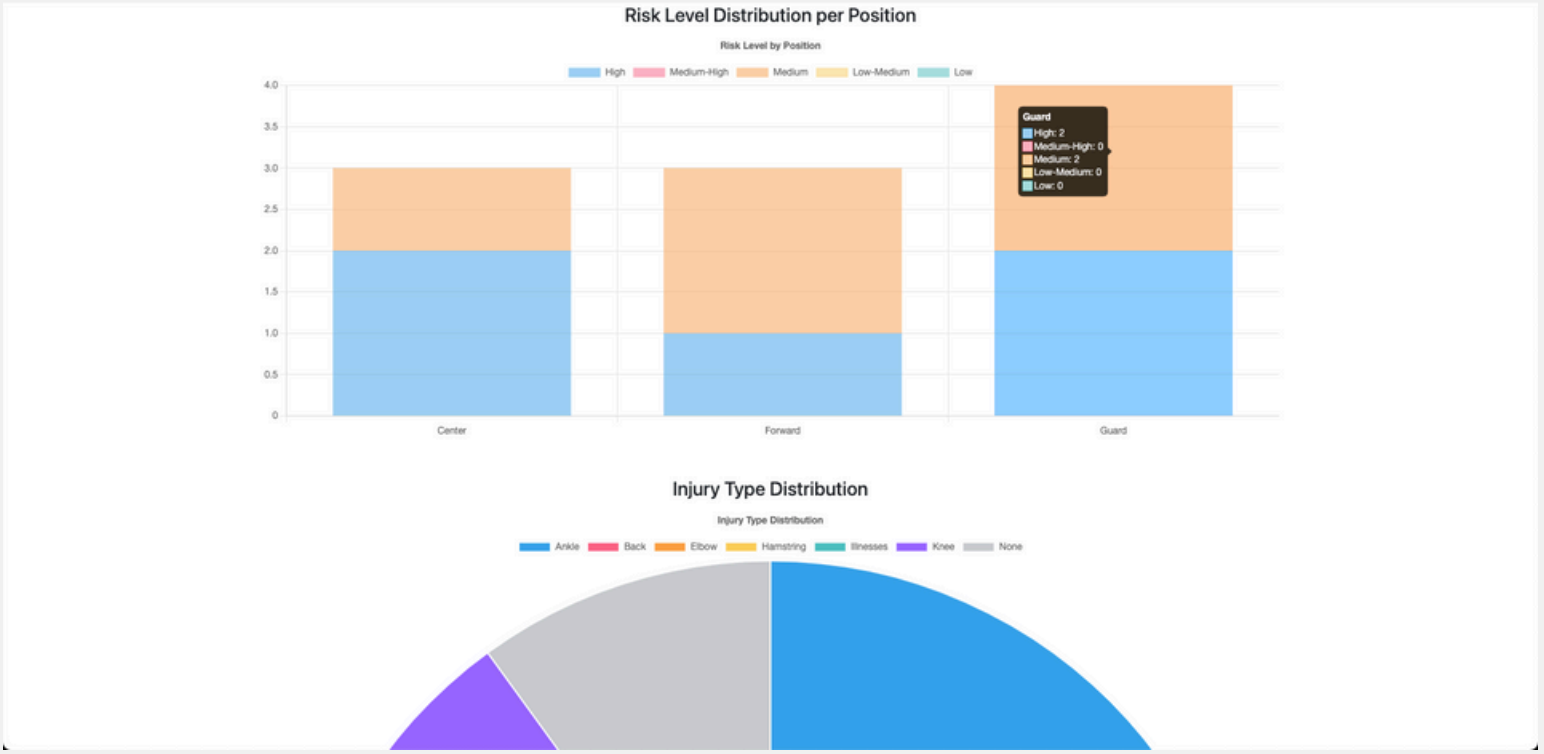
```
from django.shortcuts import render
from player_profiles.models import PlayerProfile
from collections import Counter

def dashboard_view(request):
    players = PlayerProfile.objects.all()
    total_players = players.count()
    high_risk_count = players.filter(predicted_risk_label="High").count()
    most_common_injury = Counter(players.values_list('most_common_injury', flat=True)).most_common(1)[0][0]

    return render(request, 'dashboard/dashboard.html', {
        'total_players': total_players,
        'high_risk_count': high_risk_count,
        'most_common_injury': most_common_injury,
        'injury_data': players,
    })
```

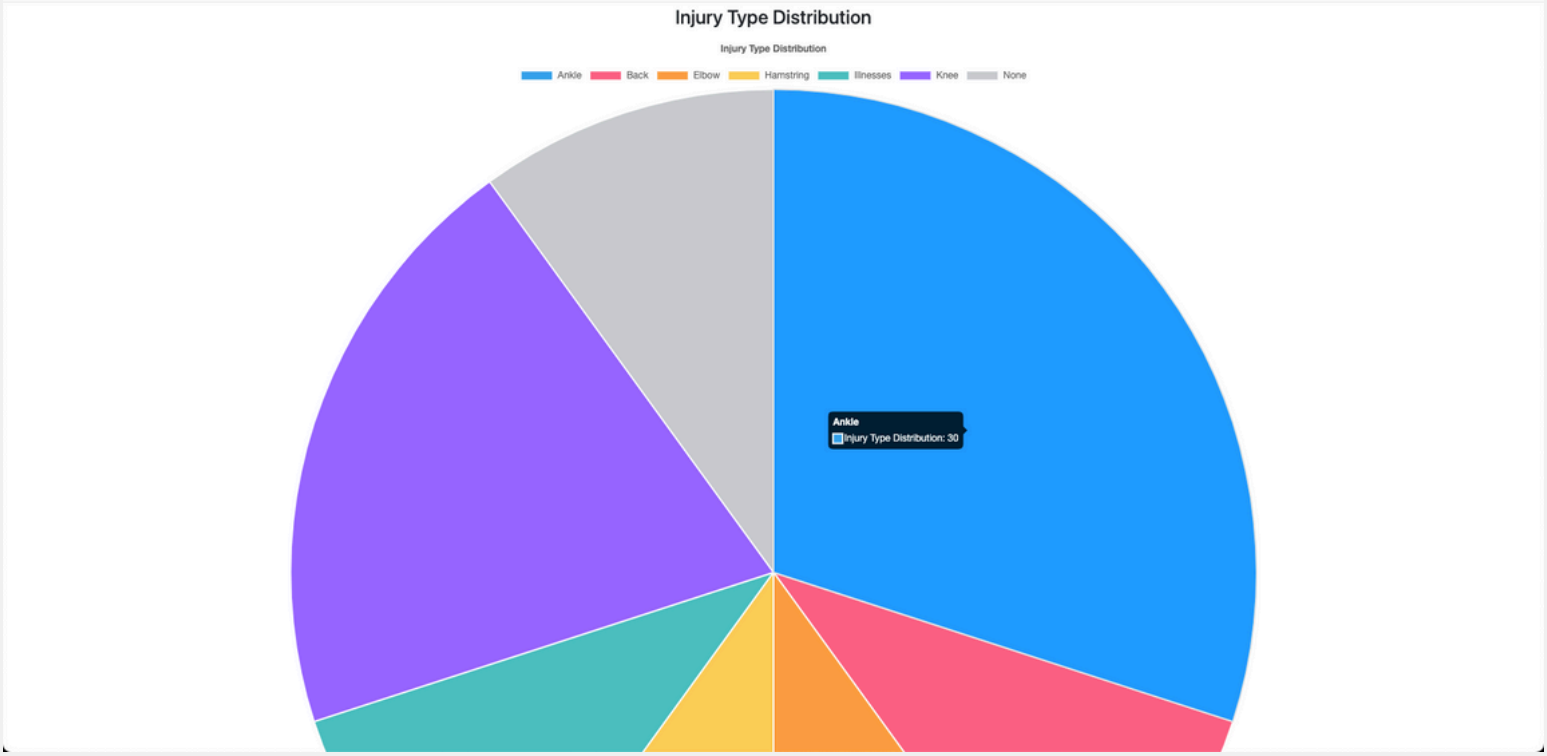
2. HTML Template with Chart.js (dashboard/dashboard.html):

```
<div class="row text-center mb-4">
  <div class="col-md-4">
    <div class="card p-3">
      <h5>Total Players</h5>
      <p class="fw-bold">{{ total_players }}</p>
    </div>
  </div>
  <div class="col-md-4">
    <div class="card p-3">
      <h5>High Risk Players</h5>
      <p class="fw-bold text-danger">{{ high_risk_count }}</p>
    </div>
  </div>
  <div class="col-md-4">
    <div class="card p-3">
      <h5>Most Common Injury</h5>
      <p class="fw-bold">{{ most_common_injury }}</p>
    </div>
  </div>
</div>

<div class="row mt-4">
  <div class="col-md-6">
    <canvas id="injuriesByPositionChart"></canvas>
  </div>
  <div class="col-md-6">
    <canvas id="injuryTypeDistributionChart"></canvas>
  </div>
</div>

<script>
  // Example: Injury Type Pie Chart
  const ctx = document.getElementById('injuryTypeDistributionChart').getContext('2d');
  new Chart(ctx, {
    type: 'pie',
    data: {
      labels: {{ labels|safe }},
      datasets: [{
        data: {{ values|safe }},
        backgroundColor: ['#ff6384', '#36a2eb', '#cc65fe', '#ffce56', '#2ecc71']
      }]
    }
  });
</script>
```

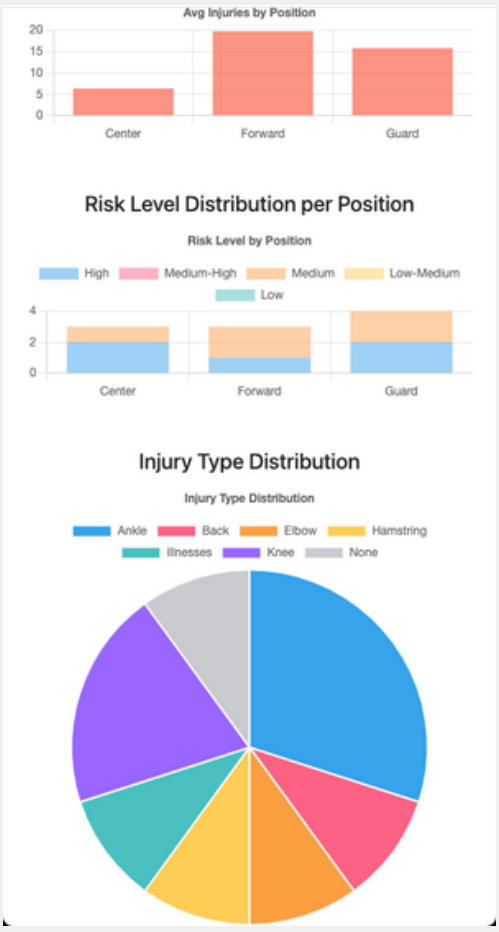
Screenshots
1. Desktop View







2. Mobile View



Feature 4: Injury Risk Prediction Page

Feature Description

The Prediction Page allows users to input basketball performance statistics and injury history to predict a player's injury risk level using a trained machine learning model. This AI-powered feature provides athletes and coaches with valuable insights into injury probability and assists in proactive performance planning.

Design & Implementation

The form uses Django's ModelForm to collect stats such as minutes played, field goals attempted, past injuries, etc. Upon submission, the data is processed and fed into a Random Forest model built using scikit-learn. The model outputs an injury risk label such as Low, Low-Medium, Medium, Medium-High, or High, which is visualized via Chart.js as a bar chart.

- **Stat Input Form:** Collects performance metrics and injury history.
- **AI Prediction:** Uses a pre-trained ML model (rf_injury_model.joblib) and label_encoder.joblib.
- **Visualization:** Dual bar charts show the predicted risk and compare it with the most common risk from other saved profiles.
- **Actions:** Users can save the prediction to their profile or create a new prediction.

Code Implementation

1. Prediction Form (predictor/forms.py):

```
from django import forms
from player_profiles.models import PlayerProfile

class PlayerPredictionForm(forms.ModelForm):
    class Meta:
        model = PlayerProfile
        fields = [
            'name', 'age', 'height_in_inches', 'weight_in_pounds',
            'position', 'games_played', 'minutes_played',
            'field_goals_attempted', 'three_point_field_goals_attempted',
            'free_throws_attempted', 'steals', 'blocks', 'fouls',
            'total_injuries', 'most_common_injury'
        ]
```

2. Prediction Logic (predictor/views.py):

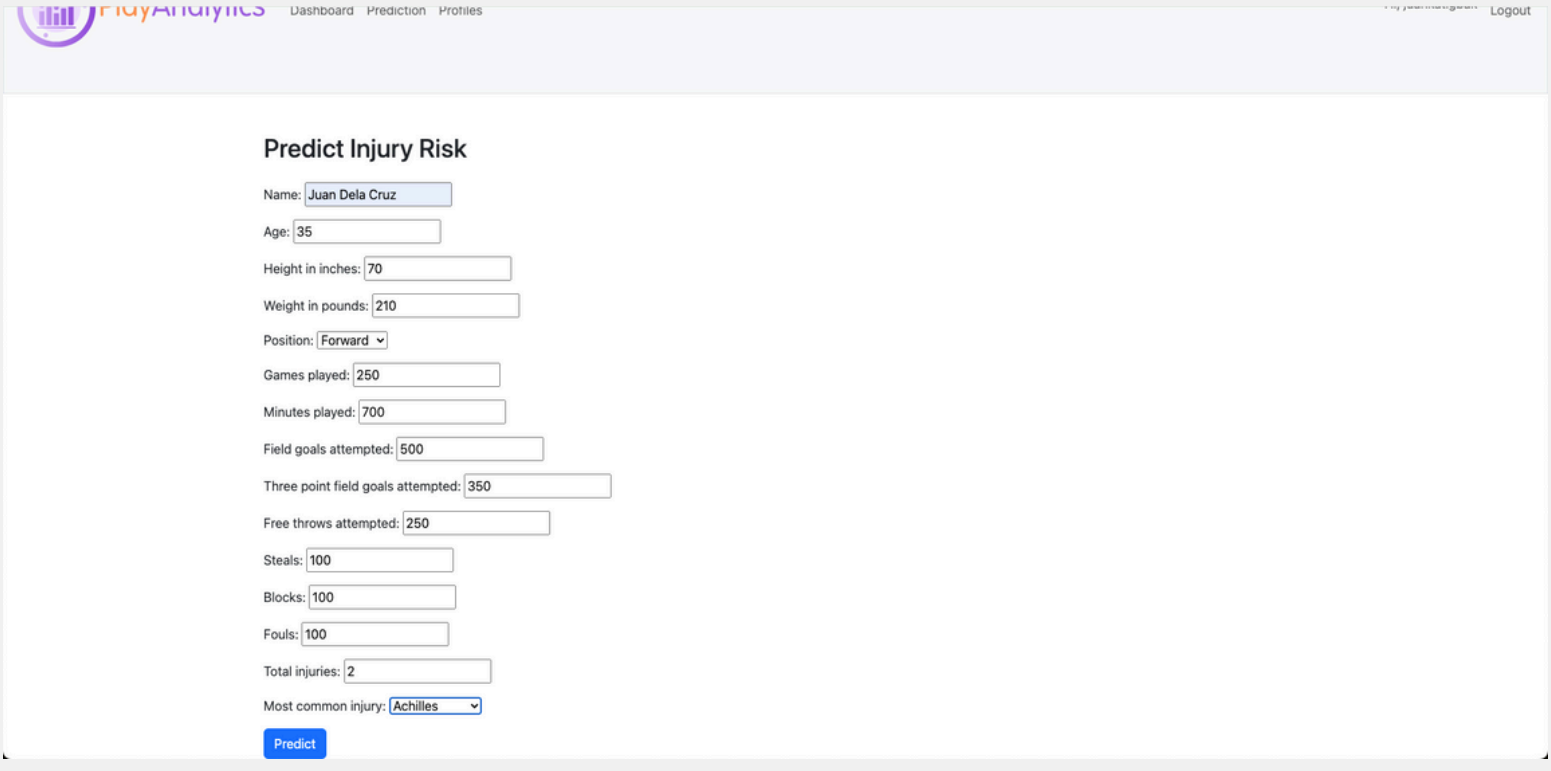
```
import pandas as pd
import joblib
from .forms import PlayerPredictionForm
from player_profiles.models import PlayerProfile
from .utils import get_most_common_risk_label

model = joblib.load('predictor/rf_injury_model.joblib')
label_encoder = joblib.load('predictor/risk_label_encoder.joblib')

@login_required
def predict_injury(request):
    if request.method == 'POST':
        form = PlayerPredictionForm(request.POST)
        if form.is_valid():
            player = form.save(commit=False)
            input_data = pd.DataFrame([{
                'Age': player.age,
                'Height_in_Inches': player.height_in_inches,
                'Weight_in_Pounds': player.weight_in_pounds,
                'Games_Played': player.games_played,
                'Minutes_Played': player.minutes_played,
                'Field_Goals_Attempted': player.field_goals_attempted,
                'Three_Point_Field_Goals_Attempted': player.three_point_field_goals_attempted,
                'Free_Throws_Attempted': player.free_throws_attempted,
                'Steals': player.steals,
                'Blocks': player.blocks,
                'Fouls': player.fouls,
            }])
            prediction_index = model.predict(input_data)[0]
            predicted_label = label_encoder.inverse_transform([prediction_index])[0]
            player.predicted_risk_label = predicted_label

            request.session['temp_player_data'] = {
                **form.cleaned_data,
                'predicted_risk_label': predicted_label
            }

            return render(request, 'predictor/predict.html', {
                'form': form,
                'predicted_risk': predicted_label,
                'most_common_risk': get_most_common_risk_label(),
                'show_results': True,
            })
    else:
        form = PlayerPredictionForm()

    return render(request, 'predictor/predict.html', {'form': form})
```

3. Prediction Page (predict.html):

```
<!-- Prediction Bar Chart Rendering -->
<canvas id="predictedRiskChart"></canvas>
<canvas id="mostCommonRiskChart"></canvas>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
 const predictedRiskData = {
  labels: ['Low', 'Low-Medium', 'Medium', 'Medium-High', 'High'],
  datasets: [{
   label: 'Predicted Player Risk',
   data: ['Low', 'Low-Medium', 'Medium', 'Medium-High', 'High'].map(
    r => r === "{{ predicted_risk }}" ? 1 : 0),
   backgroundColor: 'rgba(54, 162, 235, 0.7)'
  }]
 };
 new Chart(document.getElementById('predictedRiskChart'), {
  type: 'bar',
  data: predictedRiskData,
  options: {
   scales: { y: { beginAtZero: true, ticks: { stepSize: 1 } } }
  }
 });
</script>
```

Screenshots

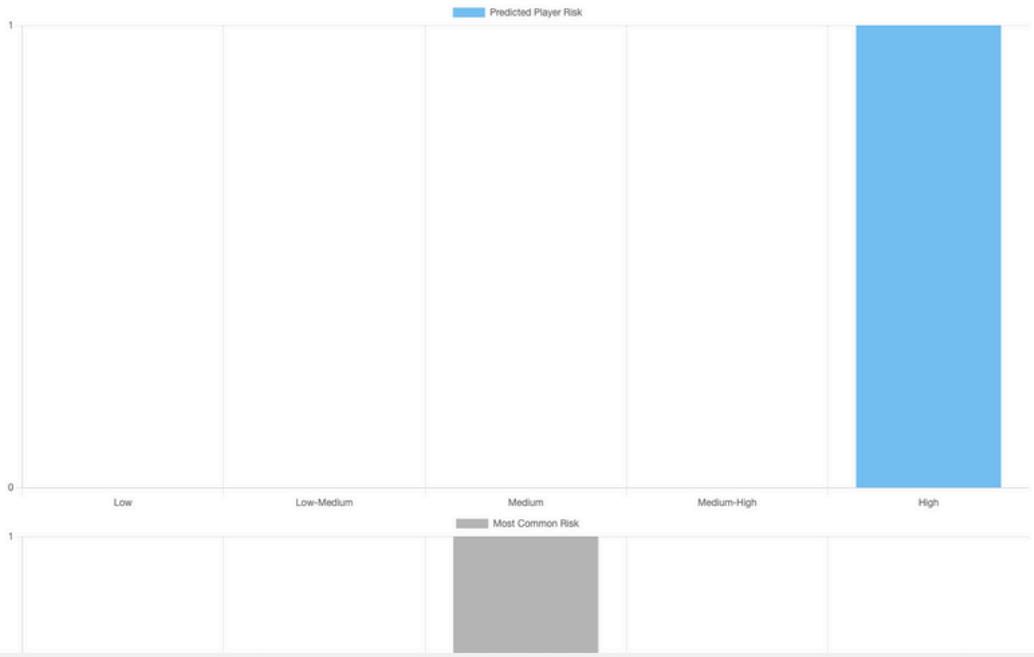
1. Desktop View

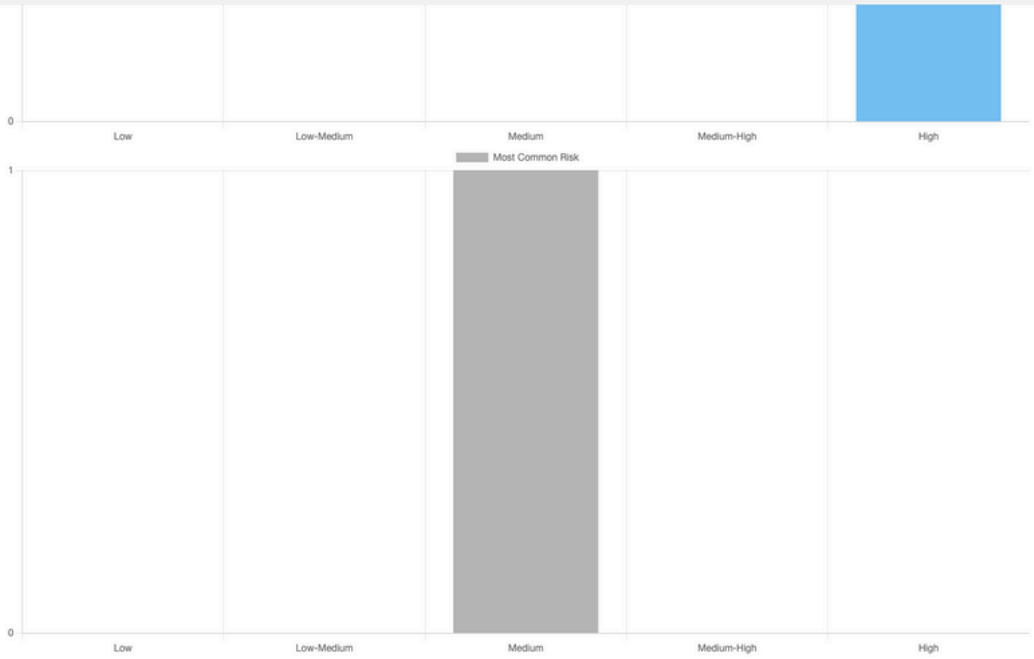


Injury Risk Prediction:

High risk based on player stats.

The most common injury risk in the system is: **Medium**

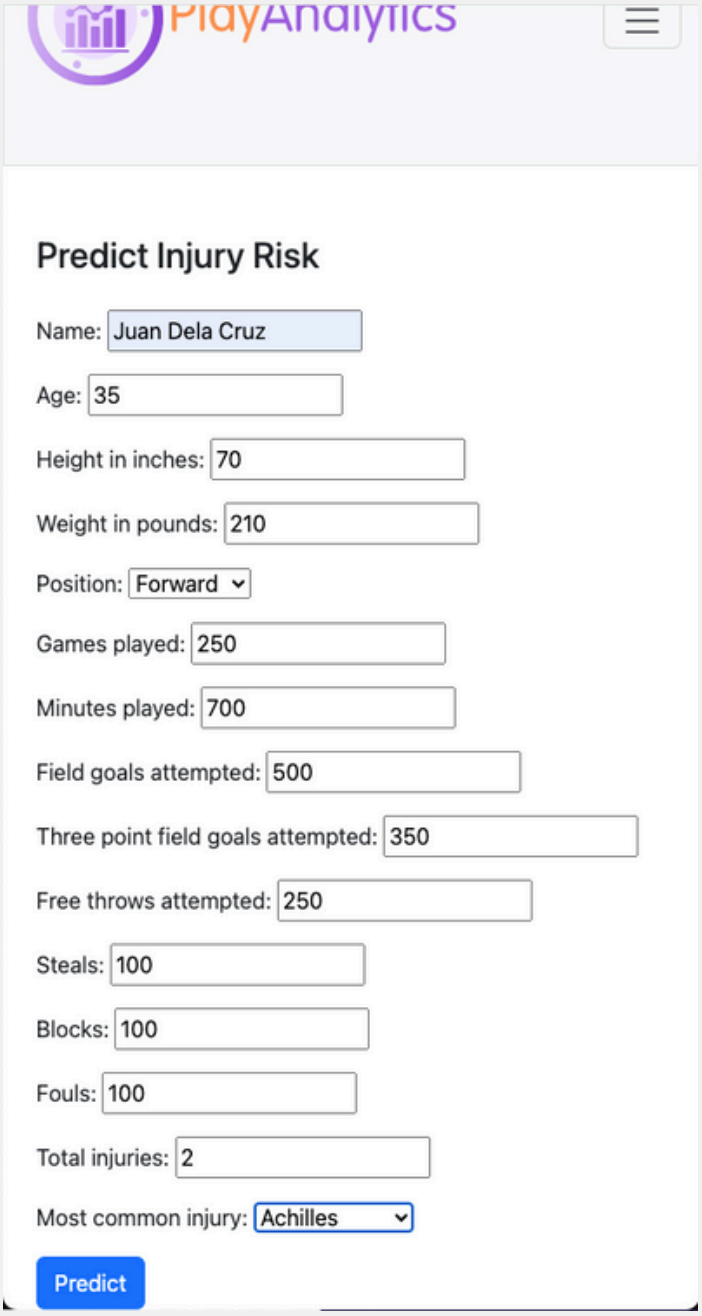




Save to Profile

New Prediction

2. Mobile View



Feature 5: Player Profile Page

Feature Description

The Player Profile Page is a centralized hub where users can view, update, and manage individual player information. It displays detailed statistics, injury risk prediction results, and allows users to upload a profile picture and maintain journal-style notes for each player. This feature adds a personal layer to the app by turning predictions into persistent, editable records — making it valuable for tracking injury history and performance over time.

Design & Implementation

The profile page combines Django model data with Bootstrap for clean, responsive layout and interactive elements. Each profile entry supports: (1) Editable profile information, (2) Profile picture upload via Pillow, (3) Add, edit, and delete personal notes, (4) Display predicted injury risk, (5) Display most common injury type & stats.

- **Bootstrap Layout:** Flex and grid utilities provide clean layout.
- **Rounded Profile Image:** Shown conditionally if image exists.
- **Notes Section:** Shows timestamped notes in reverse chronological order.
- **Edit/Delete Buttons:** For both player profiles and notes.

Code Implementation

1. Model Setup (models.py):

```
class PlayerProfile(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
    ...
    predicted_risk_label = models.CharField(max_length=50, blank=True, null=True)
    profile_picture = models.ImageField(upload_to='profile_pictures/', blank=True, null=True)

class PlayerNotes(models.Model):
    player = models.ForeignKey(PlayerProfile, on_delete=models.CASCADE, related_name='notes')
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
```

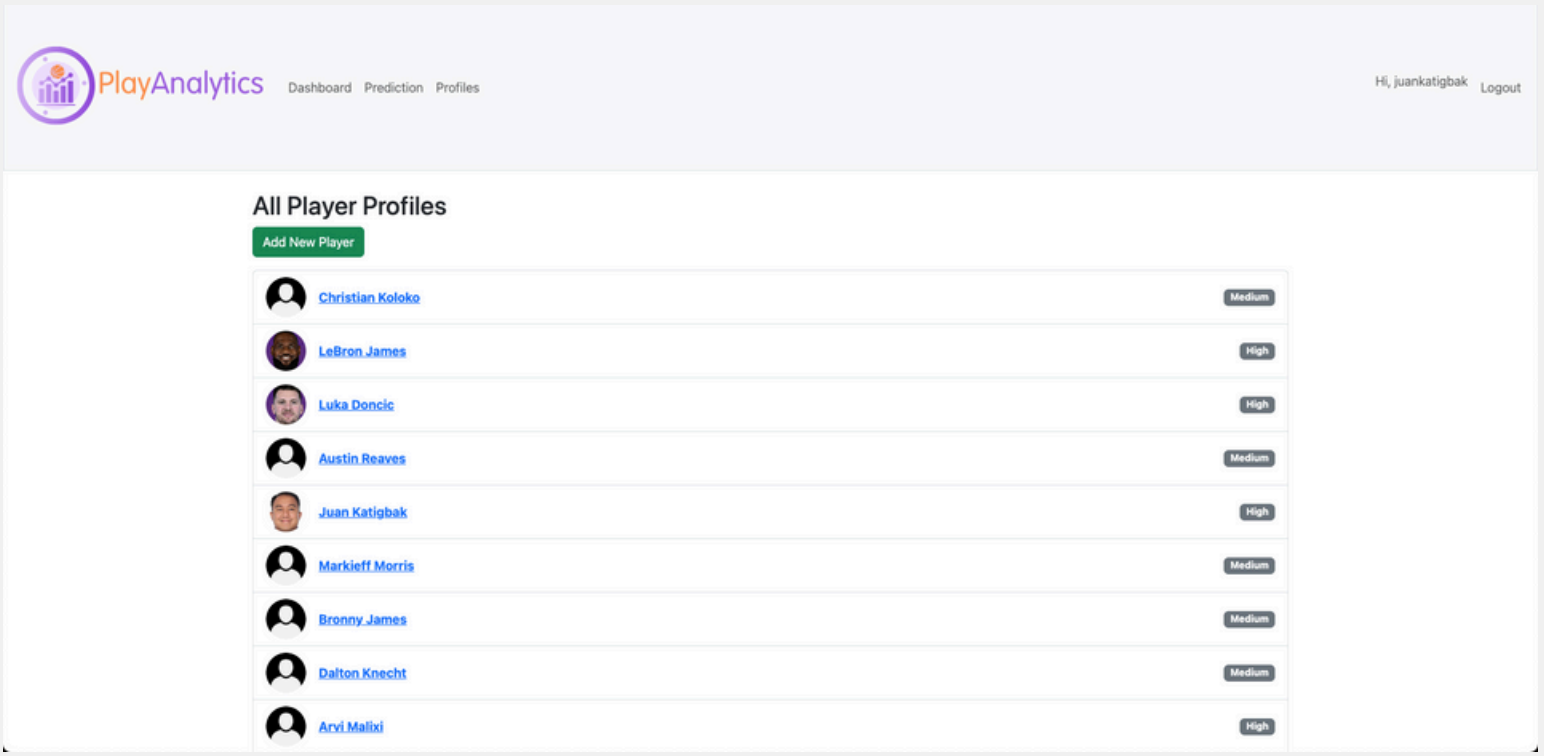
2. View Logic (views.py):

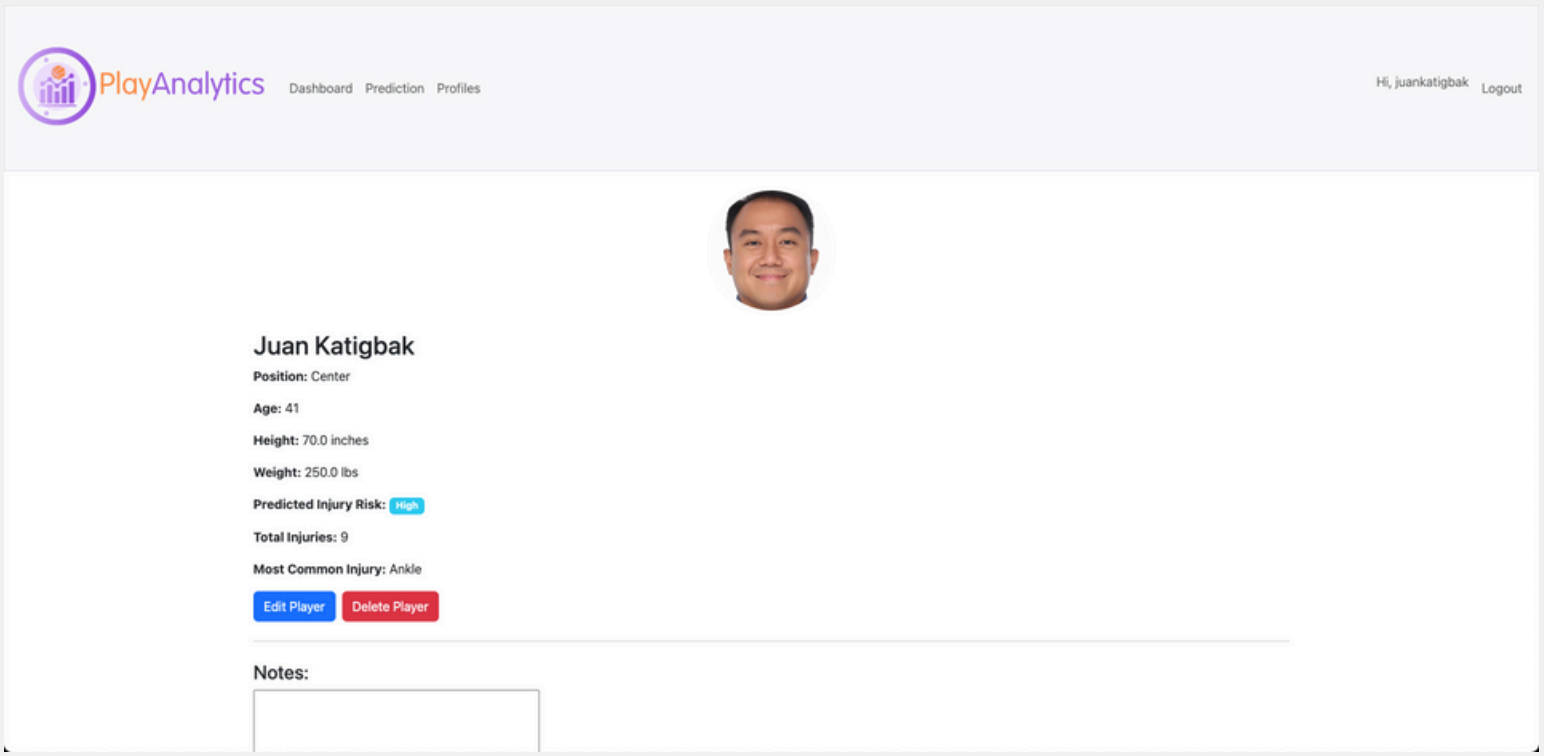
```
@login_required
def profile_detail(request, pk):
    profile = get_object_or_404(PlayerProfile, pk=pk)
    notes = profile.notes.all().order_by('-created_at')

    if request.method == 'POST':
        note_form = PlayerNoteForm(request.POST)
        if note_form.is_valid():
            note = note_form.save(commit=False)
            note.player = profile
            note.save()
            return redirect('player_profiles:profile_detail', pk=pk)
    else:
        note_form = PlayerNoteForm()

    return render(request, 'player_profiles/profile_detail.html', {
        'profile': profile,
        'notes': notes,
        'note_form': note_form
    })
```

3. Template Code (profile_detail.html):

```
{% if profile.profile_picture %}
  <img src="{{ profile.profile_picture.url }}" class="rounded-circle" style="width: 150px;">
{% else %}
  <img src="{% static 'images/default_profile.png' %}" class="rounded-circle" style="width: 150px;">
{% endif %}

<h2>{{ profile.name }}</h2>
<p><strong>Injury Risk:</strong> <span class="badge bg-info">{{ profile.predicted_risk_label|default:"None" }}</span></p>

<!-- Notes Section -->
<form method="post">{% csrf_token %}
  {{ note_form.text }}
  <button type="submit" class="btn btn-primary">Add Note</button>
</form>

<ul class="list-group mt-3">
  {% for note in notes %}
    <li class="list-group-item">
      <strong>{{ note.created_at|date:"M d, Y H:i" }}</strong><br>
      {{ note.text }}
      <a href="{% url 'player_profiles:edit_note' note.id %}" class="btn btn-sm btn-outline-secondary">Edit</a>
      <form action="{% url 'player_profiles:delete_note' note.id %}" method="post">{% csrf_token %}
        <button type="submit" class="btn btn-sm btn-outline-danger">Delete</button>
      </form>
    </li>
  {% endfor %}
</ul>
```

Screenshots
1. Desktop View





PlayAnalytics Dashboard Prediction Profiles Hi, juankatigbak Logout

Edit Juan Katigbak's Profile

Change picture:

Choose File No file chosen

Name: Juan Katigbak

Age: 41

Height in inches: 70.0

Weight in pounds: 250.0

Position: Center

Games played: 500

Minutes played: 3000.0

Field goals attempted: 1000

Three point field goals attempted: 500

Change picture:

Choose File No file chosen

Name: Juan Katigbak

Age: 41

Height in inches: 70.0

Weight in pounds: 250.0

Position: Center

Games played: 500

Minutes played: 3000.0

Field goals attempted: 1000

Three point field goals attempted: 500

Free throws attempted: 250

Steals: 100

Blocks: 180

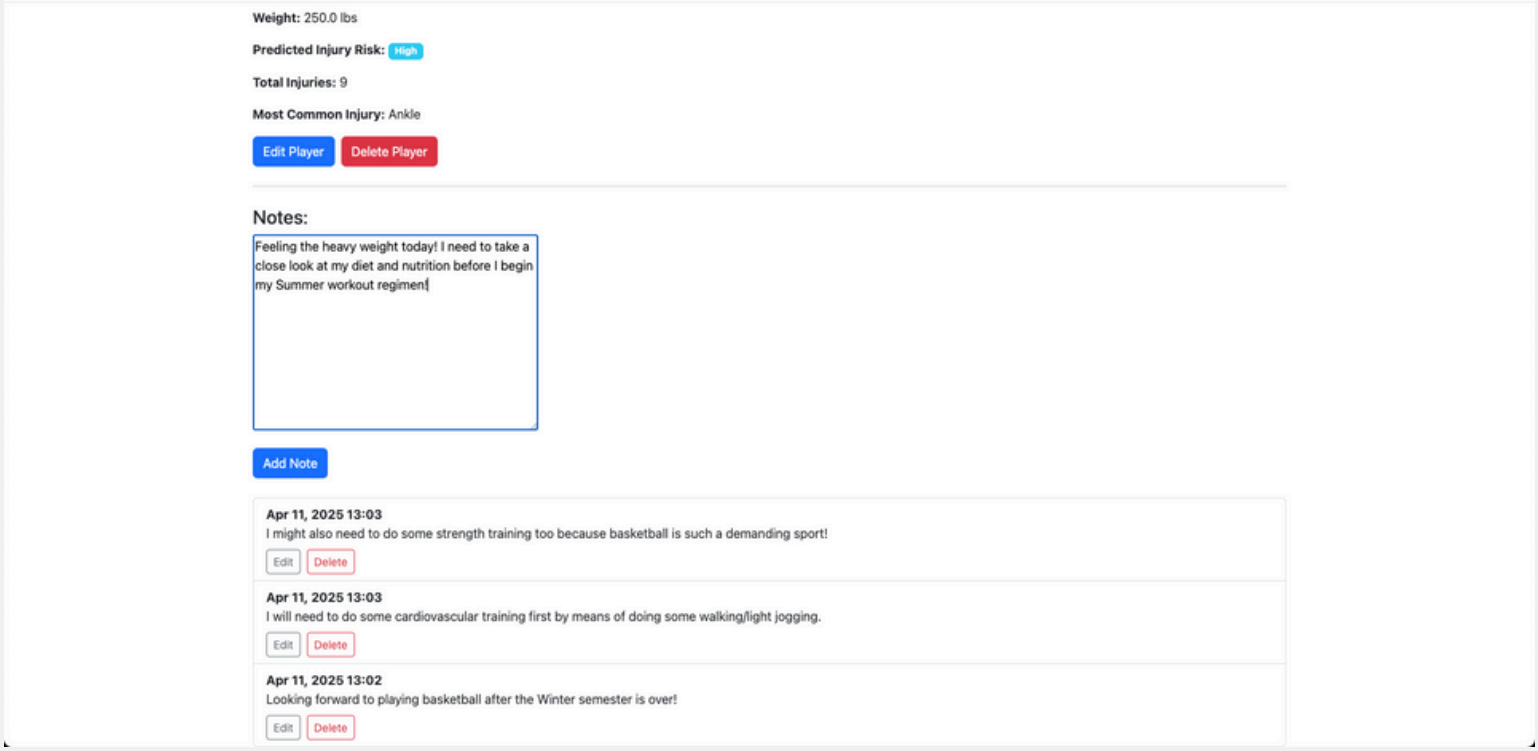
Fouls: 100

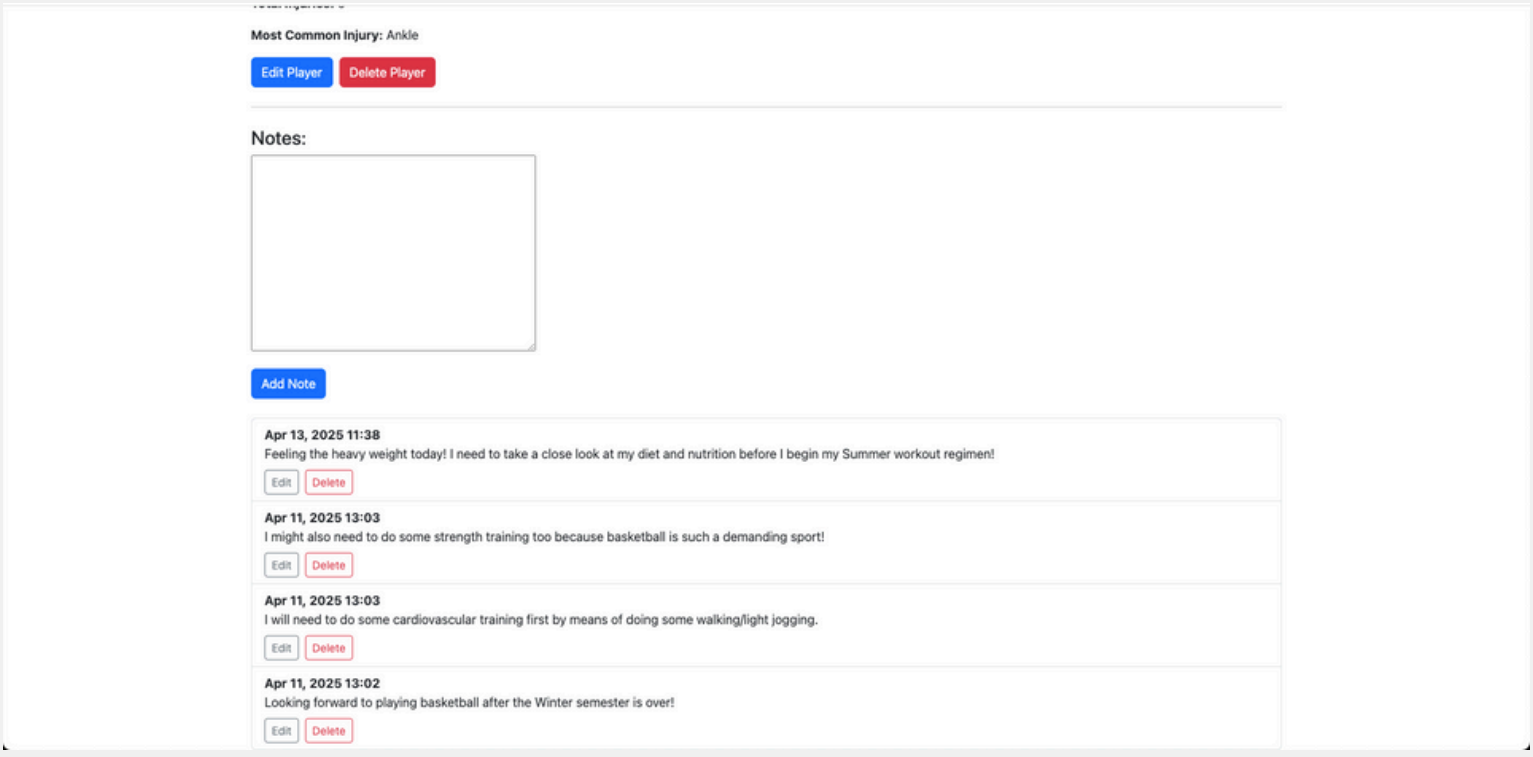
Predicted risk label: High

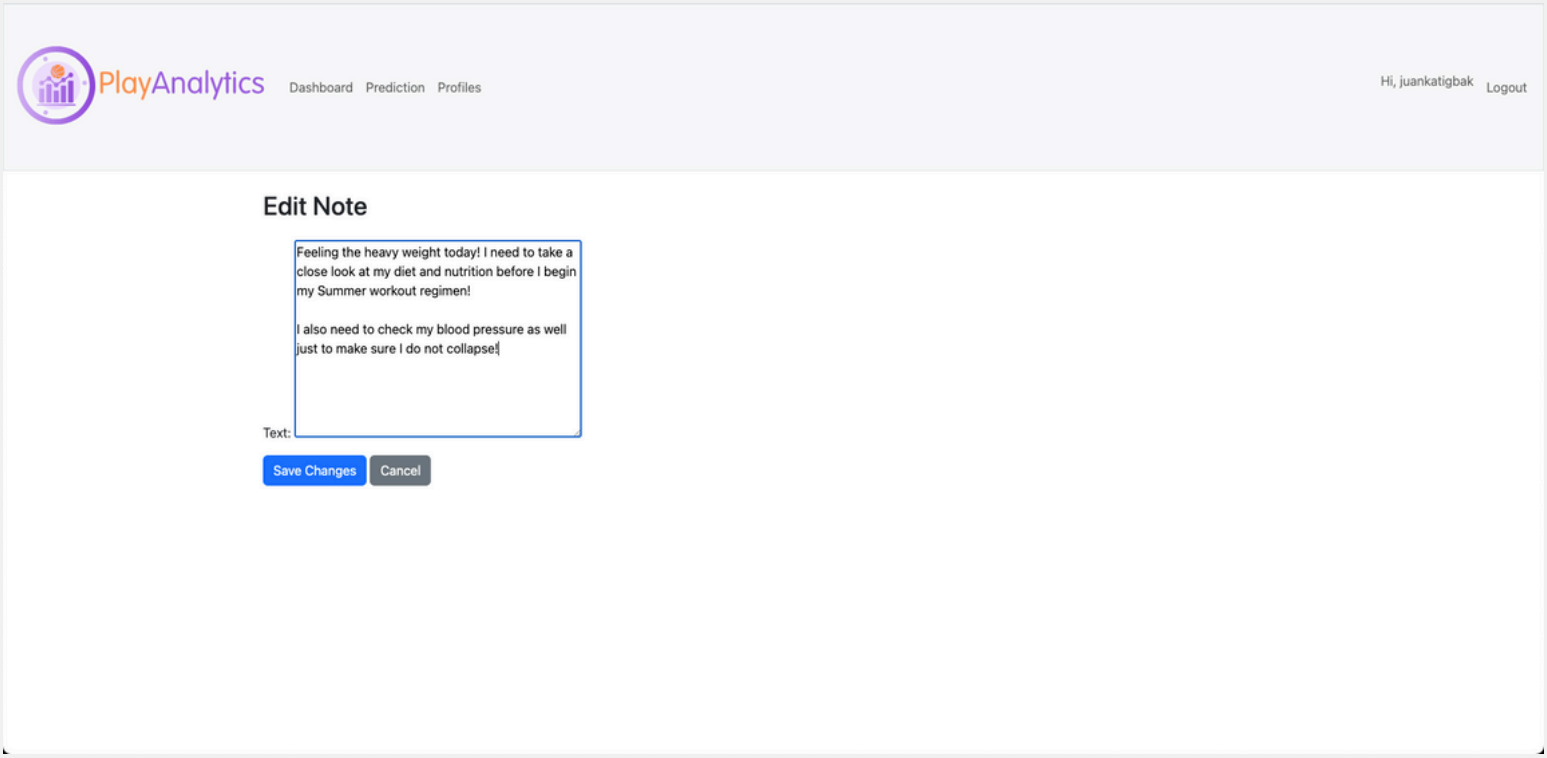
Total injuries: 9

Most common injury: Ankle

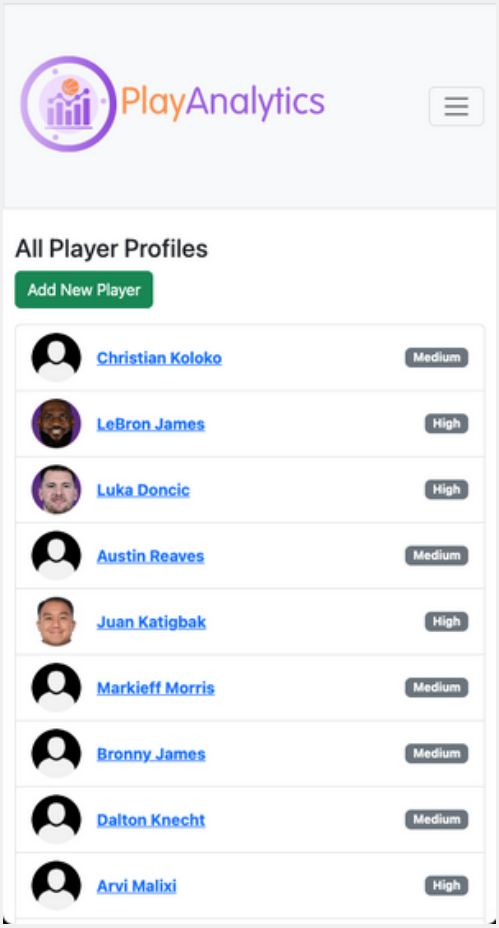
Save Changes Cancel

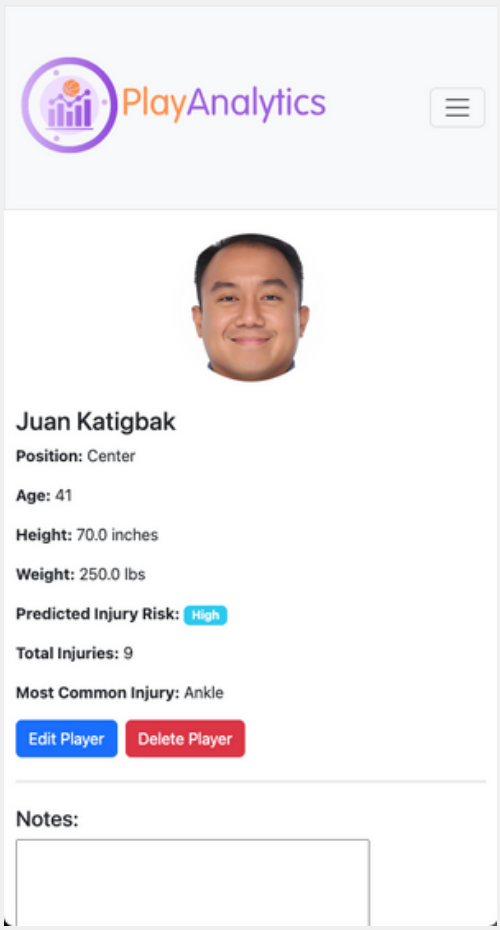

Weight: 250.0 lbs
Predicted Injury Risk: High
Total Injuries: 9
Most Common Injury: Ankle
Edit Player
Delete Player
Notes:
Feeling the heavy weight today! I need to take a close look at my diet and nutrition before I begin my Summer workout regimen!
Add Note
Apr 11, 2025 13:03
I might also need to do some strength training too because basketball is such a demanding sport!
Edit
Delete
Apr 11, 2025 13:03
I will need to do some cardiovascular training first by means of doing some walking/light jogging.
Edit
Delete
Apr 11, 2025 13:02
Looking forward to playing basketball after the Winter semester is over!
Edit
Delete

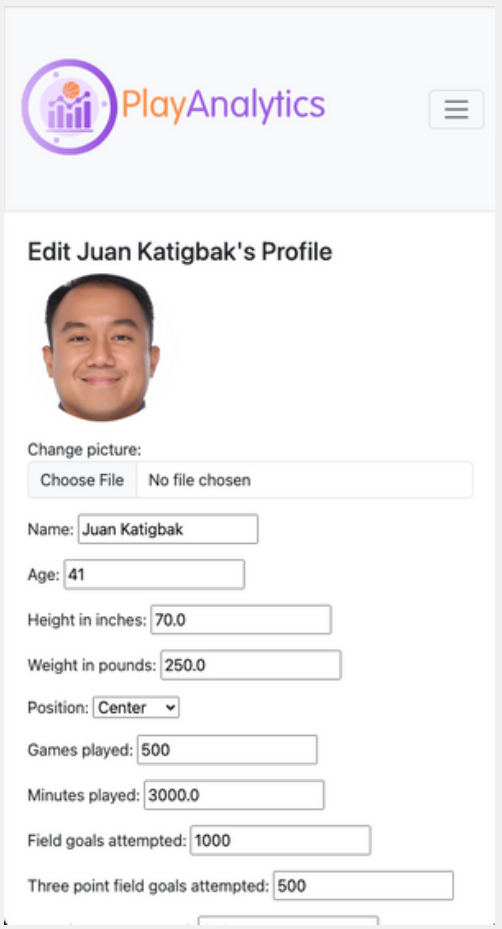

Most Common Injury: Ankle
Edit Player
Delete Player
Notes:
Add Note
Apr 13, 2025 11:38
Feeling the heavy weight today! I need to take a close look at my diet and nutrition before I begin my Summer workout regimen!
Edit
Delete
Apr 11, 2025 13:03
I might also need to do some strength training too because basketball is such a demanding sport!
Edit
Delete
Apr 11, 2025 13:03
I will need to do some cardiovascular training first by means of doing some walking/light jogging.
Edit
Delete
Apr 11, 2025 13:02
Looking forward to playing basketball after the Winter semester is over!
Edit
Delete

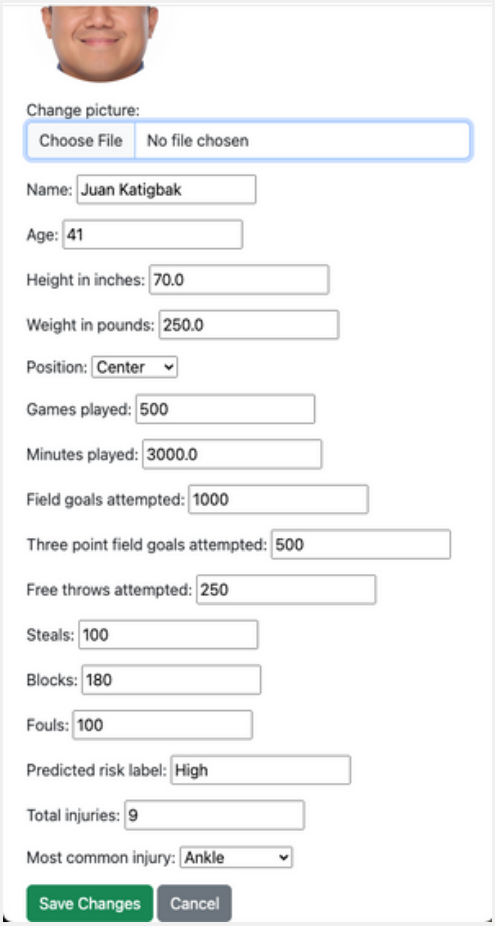


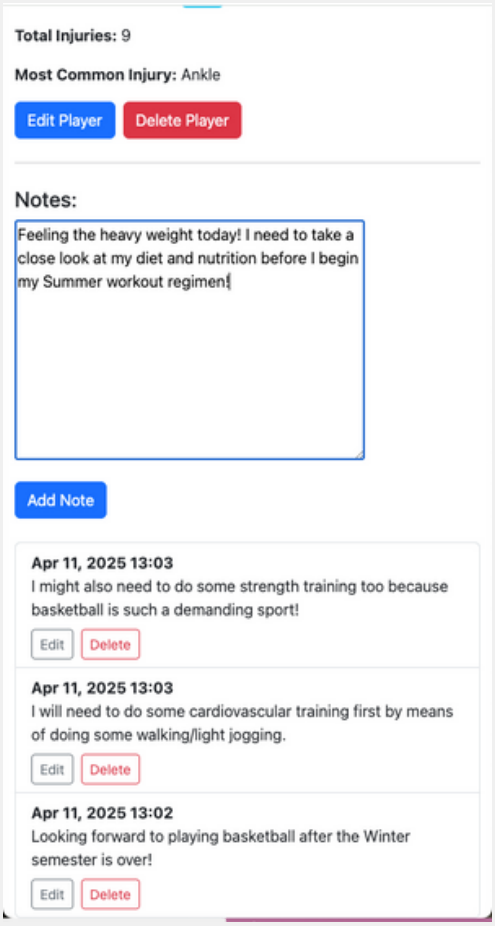
2. Mobile View

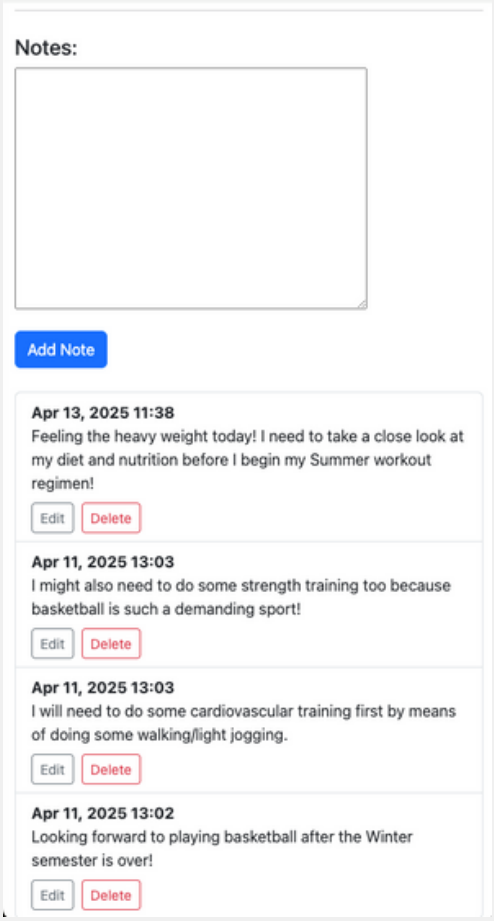


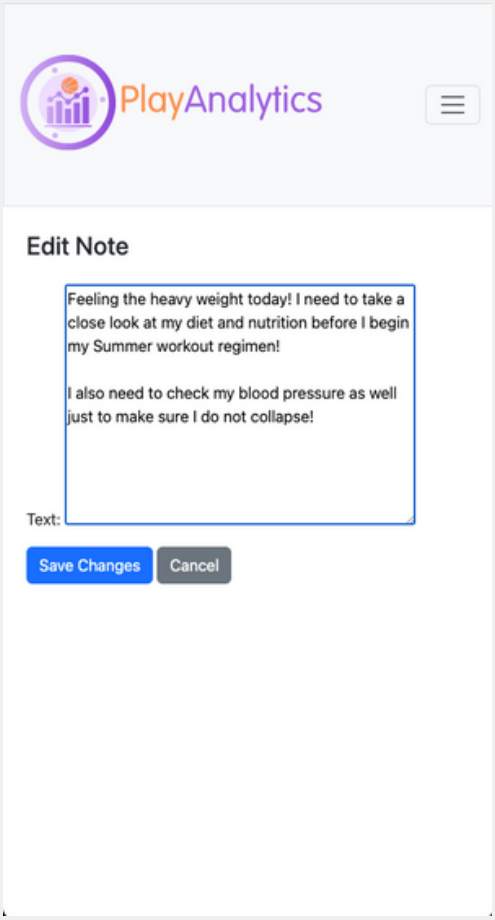





Change picture:
Choose File
No file chosen
Name: Juan Katigbak
Age: 41
Height in inches: 70.0
Weight in pounds: 250.0
Position: Center
Games played: 500
Minutes played: 3000.0
Field goals attempted: 1000
Three point field goals attempted: 500
Free throws attempted: 250
Steals: 100
Blocks: 180
Fouls: 100
Predicted risk label: High
Total injuries: 9
Most common injury: Ankle
Save Changes
Cancel


Total Injuries: 9
Most Common Injury: Ankle
Edit Player
Delete Player
Notes:
Feeling the heavy weight today! I need to take a close look at my diet and nutrition before I begin my Summer workout regimen!
Add Note
Apr 11, 2025 13:03
I might also need to do some strength training too because basketball is such a demanding sport!
Edit
Delete
Apr 11, 2025 13:03
I will need to do some cardiovascular training first by means of doing some walking/light jogging.
Edit
Delete
Apr 11, 2025 13:02
Looking forward to playing basketball after the Winter semester is over!
Edit
Delete


Notes:
Add Note
Apr 13, 2025 11:38
Feeling the heavy weight today! I need to take a close look at my diet and nutrition before I begin my Summer workout regimen!
Edit
Delete
Apr 11, 2025 13:03
I might also need to do some strength training too because basketball is such a demanding sport!
Edit
Delete
Apr 11, 2025 13:03
I will need to do some cardiovascular training first by means of doing some walking/light jogging.
Edit
Delete
Apr 11, 2025 13:02
Looking forward to playing basketball after the Winter semester is over!
Edit
Delete


PlayAnalytics
Edit Note
Feeling the heavy weight today! I need to take a close look at my diet and nutrition before I begin my Summer workout regimen!
I also need to check my blood pressure as well just to make sure I do not collapse!
Text:
Save Changes
Cancel

Reflections/Discussions

The researcher thought that the PlayAnalytics app provided extensive learning opportunities for him in both software engineering and data analytics domains. At first the researcher was hesitant with the idea laid out to him by the advisor/client that he should make a full stack development of an app from scratch, complete with a frontend, a backend, a database, and a machine learning component to think that he let her know that all he wanted to do was just data visualization via machine learning of basketball data based on the benchmarks provided by whatever algorithm was there. In hindsight, it was a good thing that he pushed through with the challenge laid out to him by the advisor/client because he wanted to get into sports data analytics and what better way to get into it than by trying to predict something like basketball injury risks based on player performance?

With the researcher taking on the challenge, he began the journey of making the PlayAnalytics app by doing crash course studying on Python and Machine Learning because he was not able to take Fundamentals of Machine Learning in Data Science CSIS3290 yet. Adding to that was the fact that the researcher had only been exposed to C#, HTML, CSS, JavaScript, PHP, and Java programming languages for the most part in Douglas College with Python being the newest and different because of the machine learning aspect. The researcher also had difficulty during PHASE 2: Crash Course Coding, Some Implementation, and Data Collecting, with the notables tasks being (1) Finalizing NBA Data composed of Performance and Injuries and (2) Finalizing Machine Learning Algorithms to predict injuries based on NBA Performance data. The researcher spent significant time sourcing unique and relevant NBA datasets related to player performance and injury history. Most of the available datasets on platforms such as Kaggle were limited or unsuitable, leading to him making his own dataset focusing on current Los Angeles Lakers by encoding his own data which was different from other researchers and it proved a tedious process. Afterwards, the researcher did some Jupyter Notebook preprocessing by selecting 11 features from the dataset that he made which represented player workload and on-court activity like minutes played, field goals attempted, fouls, and blocks. With some guidance from ChatGPT, the researcher then decided on the 11 features, computed for performance load based on some of the features which could show how some performance activities can cause injury, and created a heuristic label assignment which looked like this:

```python
# 4. Heuristic label assignment
def assign_risk_label(row):
    injuries = row["Total_Injuries"]
    perf_load = row["performance_load"]
    injury_type = str(row["Most_Common_Injury"]).lower()
    if injuries <= 2 and injury_type in ["eye", "illness", "none"]:
        return "Low"
    elif 3 <= injuries <= 5 and perf_load < 10000:
        return "Low-Medium"
    elif 3 <= injuries <= 5 and perf_load >= 10000:
        return "Medium"
    elif 6 <= injuries <= 10 and perf_load >= 15000:
        return "Medium-High"
    elif injuries > 10 and perf_load >= 20000:
        return "High"
```

The researcher was also unsure if this was the right way to do it due to limited prior experience in machine learning. The researcher faced difficulty determining the optimal algorithm for accurately predicting basketball injury risk, causing delays and overthinking during PHASE 2. The dataset that the researcher made was also overfitted due to initially having just 12 players (based on the current Los Angeles Lakers players) and the researcher decided to use synthetic data based on the stats of the original 12 and came up with 1000 players. Data was still overfitted because it revealed significant class imbalance:

```
predicted_risk_label
High           630
Medium         297
Medium-High     73
Name: count, dtype: int64
```

The researcher also had difficulty coming up with a minimum viable product since the first proof of concept was more targeted toward data analysts and not basketball athletes. Following advisor/client feedback, the researcher had to redefine and rebuild the MVP by first making a sketch by hand for the second proof of concept, shifting the target user base to non-data analytics basketball athletes. This resulted in scrapping initial implementations and working backward from the final desired outcome due to many errors that took place. There was also the problem of the prediction just predicting “High” injury risk even for players that had “Low” injury risk due to the overfitting problem of the dataset which had the current Los Angeles Lakers players having “High” injury risk because of their performance statistics and injury histories. The solution that the researcher did was to put in the code for the Prediction page a heuristic model as a backup in case the model continued “High”. This resulted in an either “Medium” or “High” injury risk label prediction when the user would click on “Predict” to predict his/her performance and injuries input.

The most satisfying part of the project for the researcher was successfully integrating a full-stack development framework with all these new technologies that he had never learned until now. The researcher can safely say that he now knows how to integrate software engineering and data analytics together to make any app that he would want to make now and in the future.

Work Date/Hours Logged

Juan Carlos Katigbak		
Date	**Number of Hours**	**Description of work done**
January 16, 2025	2	Determining topic of research for Applied Research Project
January 17, 2025	2	Initial research on topic
January 22, 2025	4	Further research on topic
January 23, 2025	3	Create 1st draft of project proposal
January 24, 2025	2	Create Github repository with file "README.md" (filled up information in there) and folders "Implementation" (with file "index.html" just to brush up on HTML code), "Misc" (with research journals related to topic including file "References.docx"), and "ReportsAndDocuments" (with files "JuanCarlosK_Proposal.pdf" and "Work Date and Hours Logs for Student) - Added all of these to repo
January 26, 2025	3	Ready project proposal for submission
February 4, 2025	2	Accomplished DataCamp's "Introduction to Python" course and added to repo, under ReportsAndDocuments folder, a screenshot of my Statement of Accomplishment for this course complete with stopwatch log with the filename "Proof of Crash Course with Python via DataCamp.png" together with a pdf file named "datacamp Statement of Accomplishment - Intro to Python.pdf". Added also to repo under ReportsAndDocuments folder a screenshot of my old kanban card called "Crash Courses on Coding with Some Implementation - 21 hours.png" because I updated the kanban card to a new one to reflect the new schedule of when I could possibly accomplish this.
February 5, 2025	2	Stopped at Section 4: Python Crash Course, 10. Python Crash Course - Part 1 of Udemy's "Python for Data Science and Machine Learning Bootcamp" course and added to repo, under ReportsAndDocuments folder, a screenshot of this which includes a stopwatch log with the filename "Proof of Udemy - Python Crash Course Part 1 of 6.png". Also added folder named "Repo for Python Data Science and Machine Learning Bootcamp" and e-book with filename "Learn to Code with Basketball 3rd Edition.pdf" to Misc folder as references for my Python/ML learning to use for implementing my project.
February 7, 2025	3	Did some non-coding research such as looking for extra research on my topic as well as what possible programming language to use. Fixed and updated Repo folders: 1. Misc folder - contains a Books folder where I placed the e-book with filename "Learn to Code with Basketball 3rd Edition.pdf" as well as added 3 new books (2 books about Basketball Strength and Conditioning and 1 book about Python Crash Course hands-on and project-based), contains a Research Journals and Studies folder where I placed all my research journals and added a new journal file "Strength and Conditioning Practices of NBA Strength and Conditioning Coaches.pdf", updated the References document to reflect this; 2. ReportsAndDocuments - contains a Proof of Crash Courses to Learn Code folder which has all my screenshot proofs of the courses that I have been taking including repo Python code and study notes of the courses.
	3.5	Did a Proof of Concept with zipped folder named "learning_log.zip" containing a learning log app using Django Python web framework as a proof of concept for my project and this is found under Proof of Concept Part 1 of 4 folder and added it to Repo under Implementation folder. The Implementation folder also contains a folder called "NEW - Supporting Docs Proof of Concept Part 1 of 4" which contains files with names "1. Server Code.pdf", "2. Shell Code.pdf", "Proof of Concept Part 1 of 4.png" which is a screenshot that contains a stopwatch log. I give acknowledgement to one of my book references [Matthes, E. (2023). Phyton crash course 3rd edition: A hands-on, project-based introduction to programming. No Starch Press.] for providing me with a step-by-step approach how I can create this full stack for my project.

February 8, 2025	1	Finished "JuanCarlosK_ProgressReport1.pdf", uploaded it on Blackboard and added it to Repo
February 14, 2025	1	Continued Proof of Concept with zipped folder named "learning_log.zip" containing a learning log app using Django Python web framework as a proof of concept for my project and this is found under NEW - Proof of Concept Part 2 of 4 folder and added it to Repo under Implementation folder. The Implementation folder also contains a folder called "NEW - Supporting Docs Proof of Concept Part 2 of 4" which contains file "Proof of Concept Part 2 of 4.png" which is a screenshot that contains a stopwatch log. I give acknowledgement to one of my book references [Matthes, E. (2023). Phyton crash course 3rd edition: A hands-on, project-based introduction to programming. No Starch Press.] for providing me with a step-by-step approach how I can create this full stack for my project.
February 17, 2025	8	Finished Proof of Concept Part 2 of 4 with zipped folder named "learning_log.zip" containing a learning log app using Django Python web framework as a proof of concept for my project and this is found under FINISHED - Proof of Concept Part 2 of 4 folder and added it to Repo under Implementation folder. The Implementation folder also contains a folder called "UPDATED - Supporting Docs Proof of Concept Part 2 of 4" which contains file "Proof of Concept Part 2 of 4 - February 17.png" which is a screenshot that contains a stopwatch log. I give acknowledgement to one of my book references [Matthes, E. (2023). Phyton crash course 3rd edition: A hands-on, project-based introduction to programming. No Starch Press.] for providing me with a step-by-step approach how I can create this full stack for my project.
February 18, 2025	5.5	Studied for Proof of Concept 3 of 4 which is basically trying to combine Django with Machine Learning to show data visualization (Proof of Concept 1 of 4 and 2 of 4 were learning and implementing Django to be used for my project) found in The Implementation folder containing a folder called "NEW - Supporting Docs Proof of Concept Part 3 of 4" which contains "NEW - Proof of Concept Part 3 of 4 - Feb 18.png" which is a screenshot that contains a stopwatch log - added to Repo. Updated file "Updated - References.docx" under Misc folder. I give acknowledgement to two of my website references [1. Braun, N. (2025). Fantasy Math: Fantasy sports predictions and analytics. Fantasy Math. https://fantasymath.com/; 2. Broni, K. (2020). KenBroTech. YouTube. https://www.youtube.com/@KenBroTech] for teaching me how to integrate Machine Learning data visualization with Django.
February 19, 2025	2	Further studying for Proof of Concept 3 of 4 to learn more about Numpy, Pandas, Matplotlib, and Seaborn found in The Implementation folder containing a folder called "Supporting Docs Proof of Concept Part 3 of 4" which contains "NEW - Proof of Concept Part 3 of 4 - Feb 19.png" which is a screenshot that contains a stopwatch log - added to Repo. I give acknowledgement to two of my website references [1. Braun, N. (2025). Fantasy Math: Fantasy sports predictions and analytics. Fantasy Math. https://fantasymath.com/; 2. Broni, K. (2020). KenBroTech. YouTube. https://www.youtube.com/@KenBroTech] for teaching me how to integrate Machine Learning data visualization with Django.
	4	Started Proof of Concept Part 4 of 4 doing the Machine Learning Process still in the Data Acquisition and Data Cleaning parts using datasets that I will be using for my proof of concept submission for the Midterms found in the Implementation folder containing folder called "NEW - Proof of Concept Part 4 of 4" which contains folder "NEW - NBA Los Angeles Lakers" which contains csv data of injury and stats of the Los Angeles Lakers players as well as a Jupyter notebook called "Lakers.ipynb" - added to Repo. Also found in the Implementation folder is a folder called "NEW - Supporting Docs Proof of Concept Part 4 of 4" which contains "NEW - Proof of Concept Part 4 of 4.png" which is a screenshot that contains a stopwatch log - added to Repo.
February 21, 2025	3	Still currently working on Proof of Concept Part 4 of 4 doing the Machine Learning Process still in the Data Acquisition and Data Cleaning parts using datasets that I will be using for my proof of concept submission for the Midterms.

February 22, 2025	4.5	A continuation of February 21, 2025, still currently working on Proof of Concept Part 4 of 4 doing the Machine Learning Process still in the Data Acquisition and Data Cleaning parts using datasets that I will be using for my proof of concept submission for the Midterms and whatever I have done so far I added in Repo and uploaded to folder "NEW - NBA Los Angeles Lakers" and deleted files there from February 19, 2025 because they are not going to be used anymore. Also found in the Implementation folder under folder called "NEW - Supporting Docs Proof of Concept Part 4 of 4" a file "NEW - Proof of Concept Part 4 of 4 - Feb 22.png" which is a screenshot that contains a stopwatch log - added to Repo.
February 23, 2025	7	Still currently working on Proof of Concept Part 4 of 4 integrating Django with Machine Learning to bring out injury risk prediction with a data visualization.
February 24, 2025	7	Finished Proof of Concept Part 4 of 4 having integrated Django with Machine Learning to bring out injury risk prediction with a data visualization and added to Repo in Implementation folder under FINISHED - PROOF OF CONCEPT in a zipped file called "lal_injury_risk.zip". Added 2 csv files in folder Proof of Concept Part 4 of 4 with datasets used for "lal_injury_risk.zip" - added to Repo.
	3	Finished Video Demo of Product and uploaded on YouTube with link saved in front page of Midterm Report. Finished "JuanCarlosK_MidtermReport.pdf", uploaded it on Blackboard and added it to Repo including finishing ReadMe file in Github.
March 11, 2025	3	Restarted doing the project again by making another proof of concept called Proof of Concept 2 this time trying to make the project a little bit closer to what Ma'am Priya suggested during the Midterm Check-in which is to focus on using machine learning to predict sports injury risk based on NBA players performance statistics this time providing benchmarks to know which 2 or 3 algorithms predict best for this and to show data visualizations of a basketball athlete wanting to see his injury risk by means of dashboards. So I went back to looking for NBA injury datasets to try to merge the dataset with my current NBA performance statistics dataset and train it to create a model for machine learning for my project. I give acknowledgement to Sir Nikhil Bhardwaj from his CSIS4260 Special Topics in Data Analytics class because he made us do an assignment last week which required us to use benchmarks and dashboards to compare the algorithms that were used. I also give acknowledgement to the Kaggle contributors for providing these datasets which I uploaded in the Repo in the Implementation folder under Proof of Concept 2 folder.
March 12, 2025	3.5	Still currently working on Proof of Concept 2 of trying to make the project a little bit closer to what Ma'am Priya suggested during the Midterm Check-in which is to focus on using machine learning to predict sports injury risk based on NBA players performance statistics this time providing benchmarks to know which 2 or 3 algorithms predict best for this and to show data visualizations of a basketball athlete wanting to see his injury risk by means of dashboards. So still cleaning the NBA data found in the Implementation folder under Proof of Concept 2 that I got from the Kaggle contributors to make it ready for training.
March 15, 2025	3	Still currently working on Proof of Concept 2 of trying to make the project a little bit closer to what Ma'am Priya suggested during the Midterm Check-in which is to focus on using machine learning to predict sports injury risk based on NBA players performance statistics this time providing benchmarks to know which 2 or 3 algorithms predict best for this and to show data visualizations of a basketball athlete wanting to see his injury risk by means of dashboards. So still cleaning the NBA data found in the Implementation folder under Proof of Concept 2 that I got from the Kaggle contributors to make it ready for training.

March 16, 2025	3	Still currently working onProof of Concept 2 of trying to make the project a little bit closer to what Ma'am Priya suggested during the Midterm Check-in which is to focus on using machine learning to predict sports injury risk based on NBA players performance statistics this time providing benchmarks to know which 2 or 3 algorithms predict best for this and to show data visualizations of a basketball athlete wanting to see his injury risk by means of dashboards. So whatever NBA data found in the Implementation folder under Proof of Concept 2 that I got from the Kaggle contributors to make it ready for training was uploaded to Repo in the same folder as "Cleaned Dataset So Far.csv.zip". Uploaded also my assignment from the CSIS4260 Special Topics in Data Analytics class called "Katigbak_300366535_Assignment1.zip" in the Repo in the Implementation folder under Proof of Concept 2 folder as a reference.
	1	Finished "JuanCarlosK_ProgressReport2.pdf", uploaded it on Blackboard and added it to Repo
March 18, 2025	6	Continued with Proof of Concept 2 and took the time to strategize/brainstorm how to make the project usable for the non-data analyst basketball athlete who wants to see his risk of injury based on his inputs into the app. Uploaded a rough sketch of how my data visualization will come out in the frontend called "Rough Draft of Data Visualization.pdf" in the Repo in the Implementation folder under Proof of Concept 2 folder.
March 19, 2025	3	Continued with Proof of Concept 2 and continued strategizing/brainstorming this time finalizing how I want the Final Dataset csv file to be like for the machine learning of Proof of Concept 2
	3	Worked on Updated - Cleaned Dataset So Far.csv.zip and uploaded to Repo in the Implementation folder under Proof of Concept 2
March 20, 2025	6	Did some updates on Updated - Cleaned Dataset So Far.csv.zip and uploaded to Repo in the Implementation folder under Proof of Concept 2; this is a continuation of cleaning my dataset to ensure that it can be trained by 2 or 3 algorithms and finally be created as the final app of the project
March 21, 2025	6	Organized the Repo especially the Implementation folder and getting all the files about Proof of Concept 1 and putting them in a folder named "Archive" just so the filing is clean. Proceeded to do the same with some files in Proof of Concept 2 folder which is also in the Implementation folder by putting most of the files in an "Archive" folder of Proof of Concept 2 folder so the only ones not archived are Rough Draft of Data Visualization.pdf and Updated - Final Cleaned Dataset.csv.zip. The Updated - Final Cleaned Dataset.csv.zip is my dataset that I will use moving forward for Proof of Concept 2 which I just accomplised today and uploaded in Repo which is subject to edits depending on the outcome with training the dataset using 2 to 3 algorithms which will decide how the final project will look like.
March 23, 2025	3	sports_management_portal folder containing the files for Proof of Concept 2 with the new data visualizations uploaded in Repo
	1	Finished "JuanCarlosK_ProgressReport3.pdf", uploaded it on Blackboard and added it to Repo
March 26, 2025	1	Finished "JuanCarlosK_Title_and_Abstract.pdf", uploaded it on Blackboard and added it to Repo
March 27, 2025	3	Continued with Proof of Concept 2 and uploaded to Repo a folder called updated_sports_management_portal which is basically an updated folder of the sports_management_portal folder and contains new updated data visualizations, a trained model which used the Random Forest algorithm called "rf_injury_model.joblib" which predicts the injury risk of the sports injury, "updated_data_visualization.py" to be able to make use of the Random Forest-trained model, "training_data.ipynb" jupyter notebook which trained my dataset using the Random Forest algorithm, "retrain_model.py" and "retrain_model.ipynb" files to add to my final Django work to retrain my dataset everytime I add new inputs to make the prediction even better, and the same "player_data_enhanced.csv" dataset. The updated_sports_management_portal folder uses Streamlit so in case you want to see the visualizations, open a Terminal (python -m venv env, then source env/bin/activate) or Command Prompt (python -m venv env, then env\Scripts\activate), install dependencies (pip install streamlit), then run application (cd updated_sports_management_portal, then streamlit run updated_data_visualization.py).

	4	Finalized Proof of Concept 2 with how the whole app will look like for the defense including the logo of the app which is called PlayAnalytics. Uploaded "playanalytics_logo.png" and "Final Rough Draft of PlayAnalytics.pdf" in the Repo in the Implementation folder under Proof of Concept 2 folder.
March 28, 2025	6	Finished Proof of Concept 2 having integrated Django with Machine Learning to bring out injury risk prediction with data visualization as well as have a player profile with journal entries. Working on the errors before gathering user feedback which I will be doing in the last phase which is PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client.
	1	Finished "JuanCarlosK_ProgressReport4.pdf", uploaded it on Blackboard and added it to Repo
March 29, 2025	2	Continuing on working on the errors of Proof of Concept 2 (which is now PlayAnalytics which will serve as the MVP or minimum viable product) before gathering user feedback which I will be doing in the last phase which is PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client. Added PlayAnalytics to Repo in Implementation folder > Proof of Concept 2 folder > play_analytics folder.
March 30, 2025	1	Edited "JuanCarlosK_ProgressReport4.pdf" and re-uploaded it on Blackboard and added it to Repo
	2	Continuing on working on the errors of Proof of Concept 2 (which is now PlayAnalytics which will serve as the MVP or minimum viable product) before gathering user feedback which I will be doing in the last phase which is PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client.
April 2, 2025	3	Completed PlayAnalytics 1.0 for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 1.0)
April 3, 2025	6	Completed PlayAnalytics 1.1 and 1.2 for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 1.1, PlayAnalytics 1.2) - reason for these versions are that I am getting close to the intended sketch that I made before I send it out for user feedback and after that I can make PlayAnalytics 2.0 based on the user feedback and have it ready for the thesis defense
April 4, 2025	5	Completed PlayAnalytics 1.3 for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 1.3)
	4	After hitting an impasse with my PlayAnalytics 1.1 to 1.3 because there were too many errors, I had to start from scratch by uploading PlayAnalytics 2.0 for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 2.0) and had to work on my process in reverse from what I did with PlayAnalytics 1.1 to 1.3
April 5, 2025	1	Completed PlayAnalytics 2.1 for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 2.1 Profile done!) . Basically 1 of 3 features (finished the Profiles page, with the remaining pages being Prediction page and Dashboard page) is finally done ready for user feedback and defense presentation
	6	80% done with PlayAnalytics 2.2/2.2.1 for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 2.2 Profile done and Prediction 80 percent done, PlayAnalytics 2.2.1 Profile done and Prediction 80 percent done) . Basically 2 of 3 features (finished the Profiles page, 80% done with Prediction page and remaining is the Dashboard page) is finally done ready for user feedback and defense presentation
April 6, 2025	5	Finished PlayAnalytics 2.3 for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 2.3 Profile and Prediction done) . Basically 2 of 3 features (finished the Profiles page and Prediction page with the remaining one being the Dashboard page) is finally done ready for user feedback and defense presentation

	3	90% done with PlayAnalytics 2.4 for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 2.4 Profile and Prediction done and Dashboard 90 percent done) . Basically 3 of 3 features (finished the Profiles page and Prediction page, 90% done with Dashboard page) is finally done ready for user feedback and defense presentation
	1	Finished "JuanCarlosK_ProgressReport5.pdf", uploaded it on Blackboard and added it to Repo
April 7, 2025	2	100% done with PlayAnalytics (now called PlayAnalytics 2.5 ready for user feedback and presentation) for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 2.5 ready for user feedback and presentation) . Basically 3 of 3 features (finished the Profiles page, Prediction page, and Dashboard page) is finally done ready for user feedback and defense presentation
	1	Created a PlayAnalytics User Feedback Survey using Microsoft Forms for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client and shared link with 21 persons with deadline to gather responses by April 10, Thursday
	1	Made some minor tweaks with PlayAnalytics (now called PlayAnalytics 2.5.1 ready for user feedback and presentation) for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client which I uploaded to Repo (Implementation > PlayAnalytics 2.5.1 ready for user feedback and presentation) and created a zipped file of PlayAnalytics which is outside the Implementation folder so that it is ready for user feedback and presentation
	1	Updated file "README.md" and uploaded to Repo
April 10, 2025	1	Uploaded to Repo "PlayAnalytics User Feedback Survey(1-17).xlsx" under ReportsAndDocuments folder collecting user feedback from 17 out of 21 respondents for PHASE 4: PlayAnalytics (Sports Management Portal) Presentation to Client
	1	Updated file "README.md" and uploaded to Repo
	2	Fixing, commenting, and cleaning up code of playanalytics.zip in preparation for the defense; uploaded in Repo. Brushing up on the technical aspects of how the PlayAnalytics app works in preparation for the defense.
April 11, 2025	2	Finished fixing, commenting, and cleaning up code of playanalytics.zip in preparation for the defense; uploaded in Repo (also in Implementation > playanalytics.zip).
	2	Doing "JuanCarlosK_FinalReport.pdf"
April 12, 2025	1	Updated file "README.md" and uploaded to Repo
	4	Still doing "JuanCarlosK_FinalReport.pdf"
April 13, 2025	1	Updated file "README.md" and uploaded to Repo
	1	Finished "Appendix A Installation Instructions Guide.pdf" and "Appendix B User Instructions Guide.pdf" and uploaded to Repo (ReportsAndDocuments > Appendix A Installation Instructions Guide.pdf, Appendix B User Instructions Guide.pdf). Uploaded to Repo "features.ipynb" and "training.ipynb" which served as machine learning model training, data preprocessing, algorithm benchmarking, and visual data analysis for the PlayAnalytics app (Implementation > features.ipynb, training.ipynb)
	7	Finished "JuanCarlosK_FinalReport.pdf" and "Presentation Slides.ppt" and uploaded to Repo (ReportsAndDocuments > JuanCarlosK_FinalReport.pdf, Presentation Slides.ppt

Concluding Remarks

First and foremost, the researcher would like to thank the advisor/client, Ma'am Priya Kandhadai, wholeheartedly because without her push, guidance, and encouragement, he would have not been able to achieve what he previously thought was impossible which was to combine software engineering and data analytics into a workable app considering he has been working in the HR industry and non-IT at that for the past 15 years. He also did this app by himself without the help of a team. So for that, the researcher would like to thank the advisor/client once again since he will be able to use this newly gained software engineering/data analytics experience and skill to hopefully find work here in Canada and for any future endeavours that he might have next.

Secondly, the researcher would like to thank his instructor Nikhil Bhardwaj from CSIS4260 Special Topics in Data Analytics because he was able to challenge students to be able to make benchmarks and dashboards to compare the algorithms for predicting things like stock market prices. This proved invaluable for the researcher as he was able to use his learnings here and apply them in his PlayAnalytics app.

The researcher also acknowledges the help of all the references listed in the References page which contributed to his knowledge of how to go about in making the app with the most notable ones as follows:

1. Phyton crash course 3rd edition: A hands-on, project-based introduction to programming by Eric Matthes from No Starch Press and this book taught the researcher all about the Django web framework which has become the foundation of the PlayAnalytics app.

2. Fantasy Math website created by Nathan Braun who also has a book called Learn to Code with Basketball 3rd Edition and these two things gave ideas to the researcher on how to do basketball-related data analytics.

3. KenBroTech YouTube channel by Kenneth Broni for teaching the researcher how to integrate Machine Learning data visualization with Django.

Lastly, the researcher openly admits to using ChatGPT Plus, an artificial intelligence tool, to be able to make PlayAnalytics a reality by means of ideas generation, algorithm benchmarking, prediction, creating the kanban board, coding (with comments), debugging, report creation (including ReadMe file), survey creation, and data analysis. Without it, he would have not been able to make everything work since the books and YouTube channels that he looked at did not have any information on how to combine software engineering with machine learning especially for the topic that he wanted to do which was in sports data science since they were limited to just data visualizations and not training of data so he had to do things from scratch with the help of ChatGPT.

References

I. Research Journals and Studies

1. Owen, R., Owen, J. A., & Evans, S. (2024). Artificial intelligence for sport injury prediction. In Artificial Intelligence for Sport Injury Prediction. Springer.

2. Lu, Y., Pareek, A., Lavoie-Gagne, O. Z., Forlenza, E. M., Patel, B. H., Reinholz, A. K., Forsythe, B., & Camp, C. L. (2022). Machine learning for predicting lower extremity muscle strain in National Basketball Association athletes. The Orthopaedic Journal of Sports Medicine, 10(7), 23259671221111742.

3. Papageorgiou, G., Sarlis, V., & Tjortjis, C. (2024). Evaluating the effectiveness of machine learning models for performance forecasting in basketball: A comparative study. Knowledge and Information Systems, 66, 4333–4375.

4. Chmait, N., & Westerbeek, H. (2021). Artificial intelligence and machine learning in sport research: An introduction for non-data scientists. Frontiers in Sports and Active Living, 3, Article 682287.

5. Dindorf, C., Bartaguiz, E., Gassmann, F., & Fröhlich, M. (2022). Conceptual structure and current trends in artificial intelligence, machine learning, and deep learning research in sports: A bibliometric review. Preprint.

6. Li, S., & Zhang, W. (2022). Evaluation method of basketball teaching and training effect based on wearable devices. Frontiers in Physics, 10, Article 900169.

7. Yang, X. (2024). Construction of measurement index system of basketball players' specific physical fitness training based on AI intelligence and neural network. Molecular & Cellular Biomechanics, 21(1), 250.

8. Georgievski, B., & Vrtagic, S. (2021). Machine learning and the NBA game. Journal of Physical Education and Sport.

9. Simenz, C. J., Dugan, C. A., & Ebben, W. P. (2005). Strength and conditioning practices of National Basketball Association strength and conditioning coaches. The Journal of Strength and Conditioning Research, 19(3), 495–504.

II. Books

1. Braun, N. (2023). Learn to code with basketball.

2. Matthes, E. (2023). Phyton crash course 3rd edition: A hands-on, project-based introduction to programming. No Starch Press.

3. Foran, B. (2026). Complete conditioning for basketball: National Basketball Strength & Conditioning Association. Human Kinetics.

4. Gillett, J. & Burgos, B. (2020). Strength Training for Basketball: National Strength and Conditioning Association. Human Kinetics.

III. Websites

1. National Basketball Association. (n.d.). NBA official website. https://www.nba.com
(**Note:** NBA player performance statistics and profile pictures from two famous players, LeBron James and Luka Dončić, from the current Los Angeles Lakers were obtained here and used for the app)

2. Pro Sports Transactions Archive. (n.d.). Pro Sports Transactions.
https://www.prosportstransactions.com/
(**Note:** NBA player injury statistics were obtained here and used for the app)

3. Braun, N. (2025). Fantasy Math: Fantasy sports predictions and analytics. Fantasy Math. https://fantasymath.com/

4. Broni, K. (2020). KenBroTech. YouTube. https://www.youtube.com/@KenBroTech

5. Gilermo, D. (n.d.). NBA players stats [Dataset]. Kaggle.
https://www.kaggle.com/datasets/drgilermo/nba-players-stats/data

6. Lauton, L. (n.d.). NBA injury stats (1951-2023) [Dataset]. Kaggle.
https://www.kaggle.com/datasets/loganlauton/nba-injury-stats-1951-2023

7. Lounias, T. (n.d.). NBA stat scrape and analysis [Kaggle Notebook]. Kaggle.
https://www.kaggle.com/code/themelissoulounias/nba-stat-scrape-and-analysis

8. Chiodi, A. (n.d.). NBA injury forecasting [Code notebook]. Kaggle.
https://www.kaggle.com/code/anthonychiodi/nba-injury-forecasting

9. Liu, I. C. (n.d.). NBA player stats and injured data from ‘13 to ‘23 [Dataset]. Kaggle.
https://www.kaggle.com/datasets/icliu30/nba-player-stats-and-injured-data-from-13-to-23

IV. AI Tools Section

1. ChatGPT Plus (AI tool) - Assisted in every step of the app's creation: ideas generation, algorithm benchmarking, prediction, creating the kanban board, coding (with comments), debugging, report creation (including ReadMe file), survey creation, and data analysis.

V. Technology Stack

1. Amazon. (2024). Amazon Prime [E-commerce platform]. https://www.amazon.com/prime

2. Bootstrap Team. (2024). Bootstrap (Version 5) [Front-end framework]. https://getbootstrap.com

3. Braun, N. (2024). Fantasy Math [Sports data science platform]. https://www.fantasymath.com

4. Canva. (2024). Canva Pro [Design software]. https://www.canva.com
(**Note:** Photos of basketball players shown in this final report including the previous proposal, progress and midterms reports that were submitted were obtained and used from Canva)

5. Chart.js Contributors. (2024). Chart.js (Version 4) [JavaScript charting library]. https://www.chartjs.org

6. DataCamp. (2024). DataCamp [Online learning platform]. https://www.datacamp.com

7. Django Software Foundation. (2024). Django (Version 5.2) [Web framework]. https://www.djangoproject.com

6. GitHub. (2024). GitHub [Code hosting platform]. https://github.com

7. Hipp, D. R. (2024). SQLite (Version 3) [Database engine]. https://www.sqlite.org

8. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://matplotlib.org

9. Microsoft. (2024). Microsoft Office 365 [Productivity suite]. https://www.microsoft.com/microsoft-365

10. Microsoft. (2024). Microsoft Forms [Survey tool]. https://forms.office.com

11. Microsoft. (2024). Microsoft Outlook [Email client]. https://outlook.office.com

12. Microsoft. (2024). Microsoft Excel [Spreadsheet software]. https://www.microsoft.com/microsoft-365/excel

13. NumPy Developers. (2024). NumPy (Version 2) [Python library]. https://numpy.org

14. OpenAI. (2024). ChatGPT (GPT-4) [Large language model]. https://chat.openai.com/

15. Pandas Development Team. (2024). Pandas (Version 2) [Python library]. https://pandas.pydata.org

16. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org

17. Python Software Foundation. (2024). Python (Version 3.12) [Programming language]. https://www.python.org

18. Tieleman, T. (2024). Joblib (Version 1.4) [Model serialization tool]. https://joblib.readthedocs.io

19. Willison, A. (2024). Pillow (Version 11.1) [Image processing library]. https://python-pillow.org

20. Udemy. (2024). Udemy [Online learning platform]. https://www.udemy.com

21.YouTube. (2024). YouTube [Video sharing platform]. https://www.youtube.com

Jupyter Project. (2015). Jupyter Notebook [Interactive computing platform]. https://jupyter.org

VI. Instructor

1. Nikhil Bhardwaj's CSIS4260 Special Topics in Data Analytics - he made the researcher do an assignment which required him to use benchmarks and dashboards to compare the algorithms that were used.

Appendix A:

Installation Guide

1. Extraction of playanalytics.zip

First, click on the green (<> Code) button and click on Download ZIP to be able to download the whole Repository. Extract and unzip playanalytics.zip. If there is a nested folder (playanalytics/playanalytics/), ensure the inner folder is moved to a convenient location like your Desktop.

2. Set Up a Virtual Environment

Recommended: Use Python 3.11+ (Anaconda or system Python)

For macOS/Linux (Terminal):

python -m venv env
source env/bin/activate

For Windows (Command Prompt):

python -m venv env
env\Scripts\activate

3. Install Dependencies

For macOS/Linux (Terminal)/For Windows (Command Prompt):

pip install django django-bootstrap5 joblib scikit-learn numpy pandas pillow

4. Run the Application

using macOS/Linux's Terminal, run:

cd playanalytics (depending on the location of the folder, suggestion would be to put it on the Desktop, then on the terminal cd ~/Desktop then cd playanalytics)

python manage.py runserver

using Windows' Command Prompt, run:

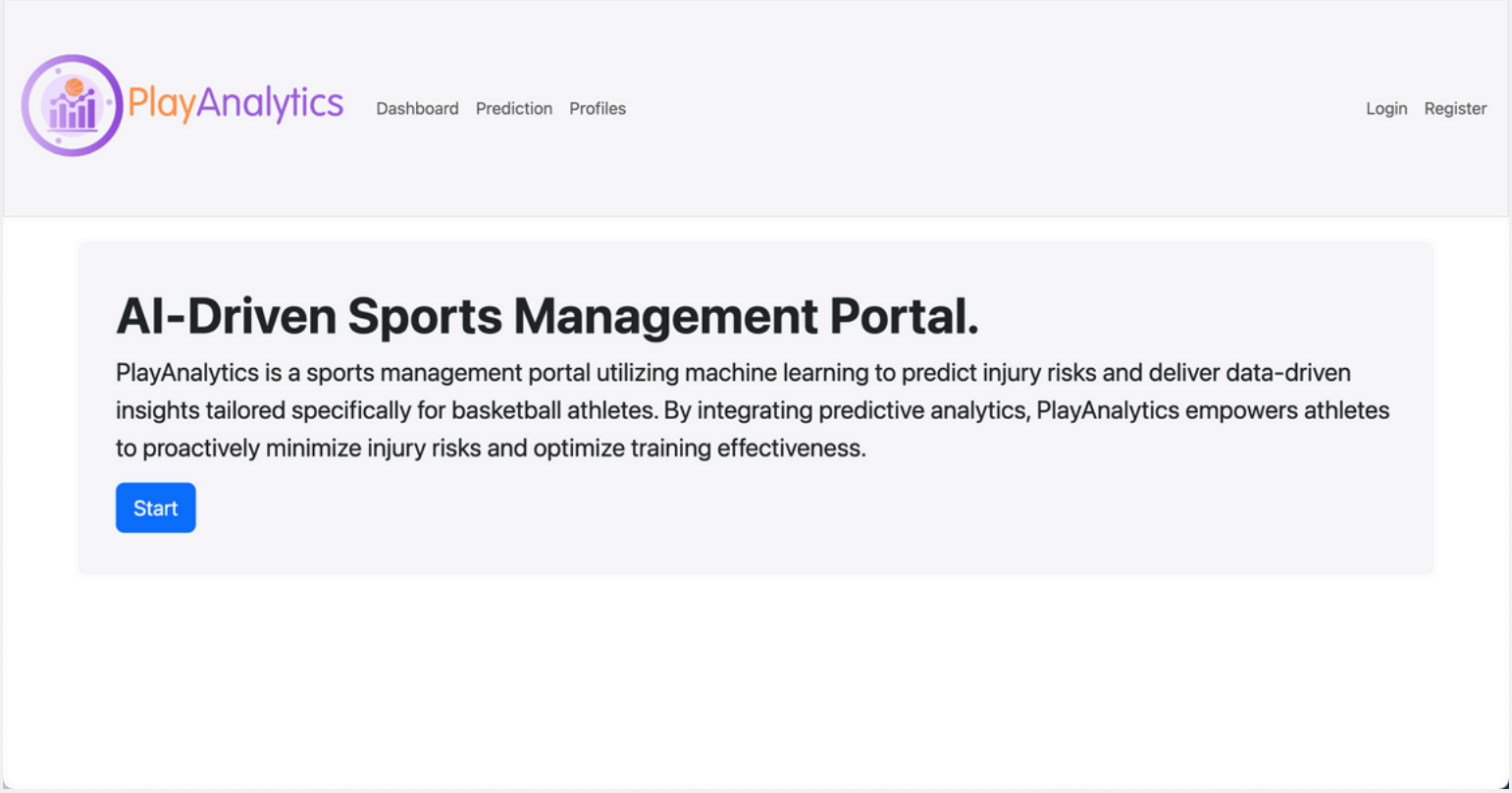
cd %USERPROFILE%\Desktop\playanalytics
(e.g. cd OneDrive\Desktop\playanalytics)

python manage.py runserver

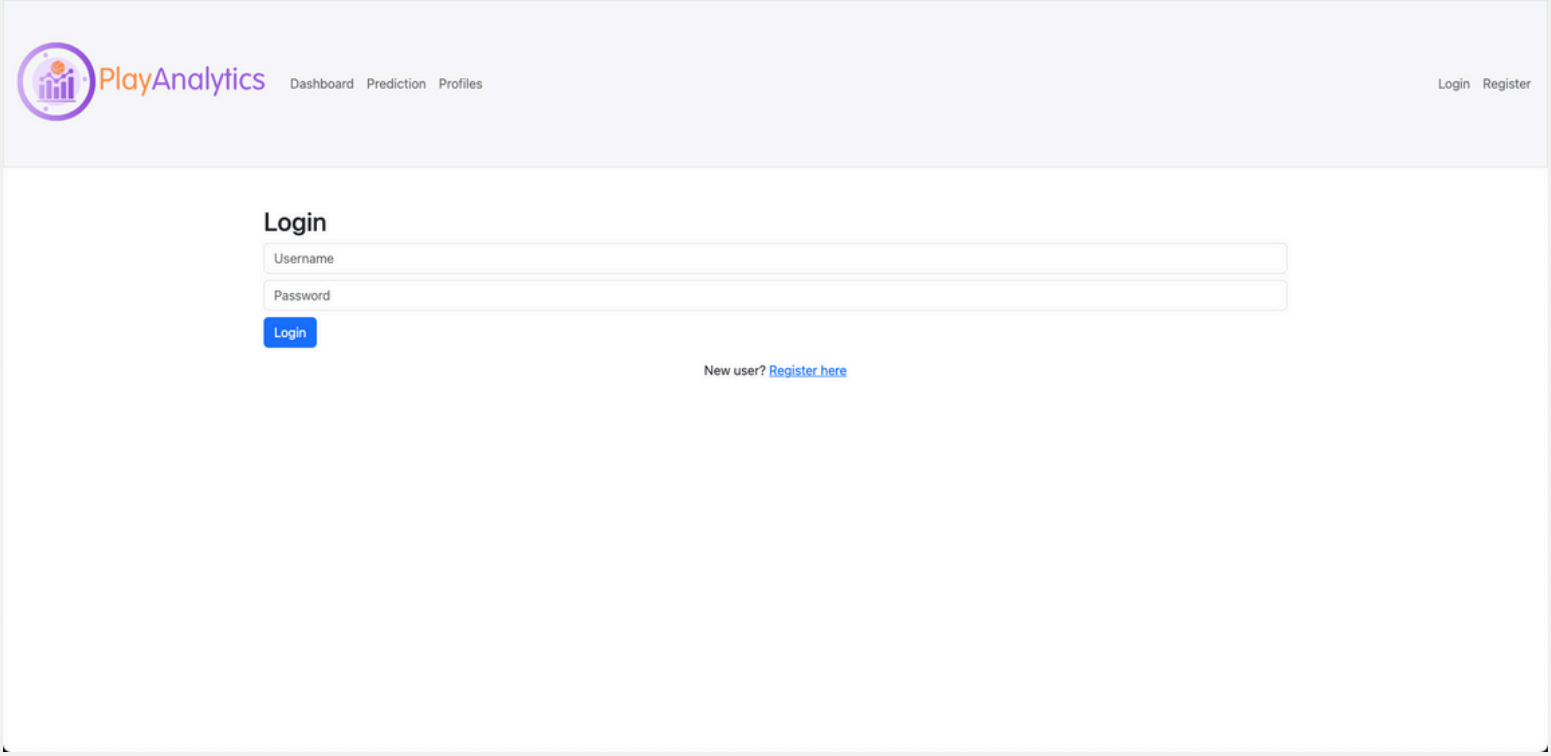
Open your browser and go to http://127.0.0.1:8000/ (or localhost:8000 - which is usually the default)

Appendix B: User Guide

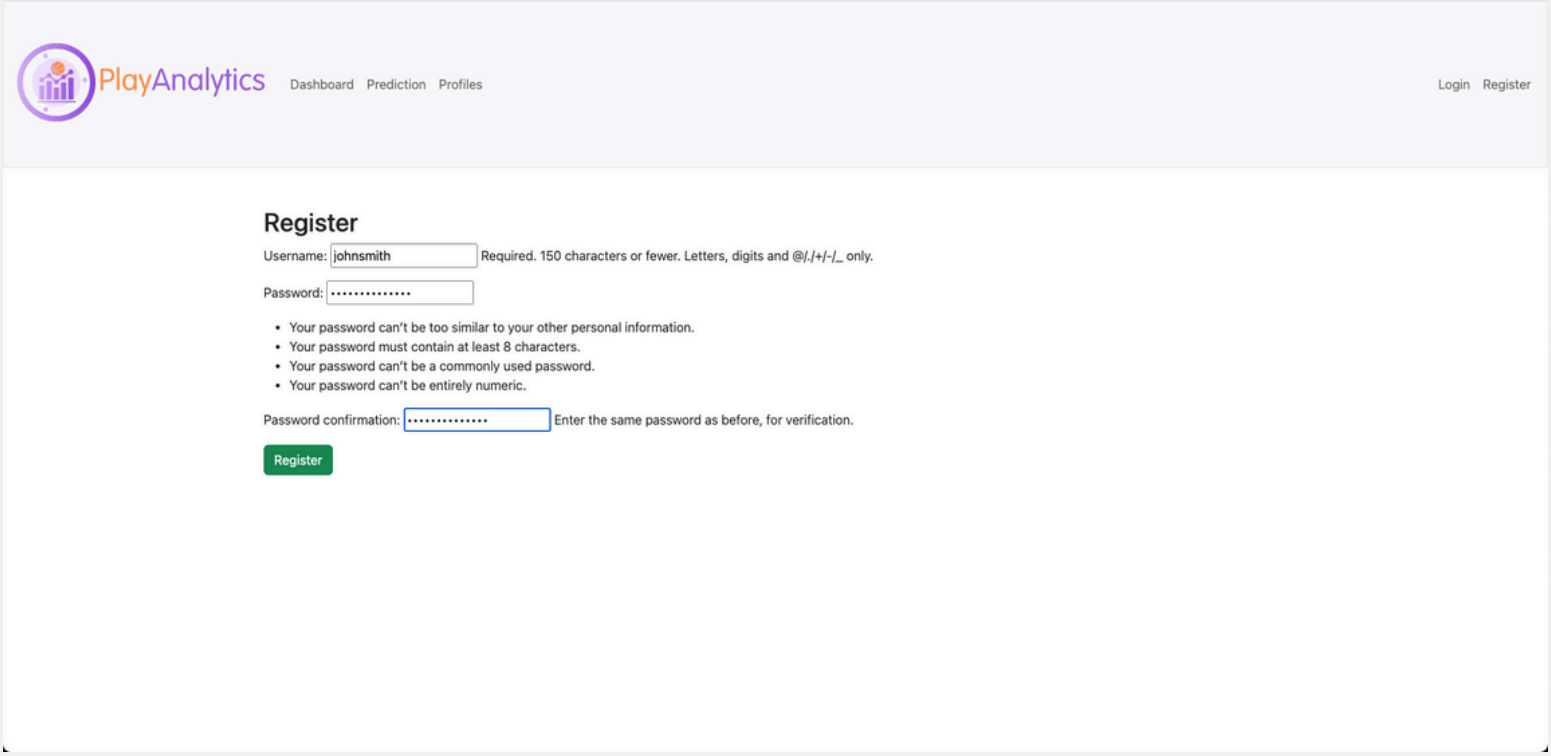
Step 1: Landing Page – Click "Start" to go to the User Registration and Login.



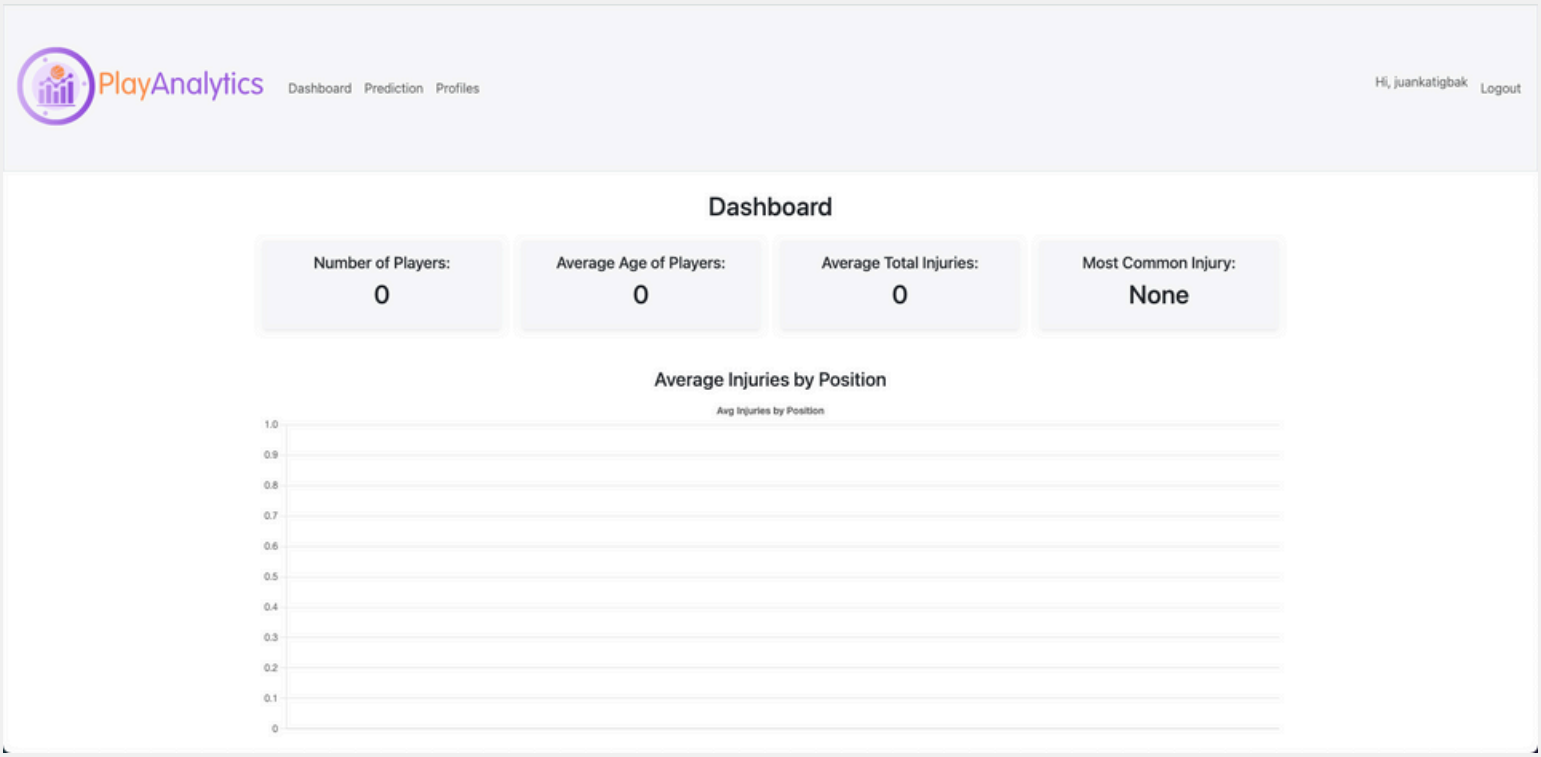
Step 2: User Registration and Login – If first time user, click "Register here" and input Username and Password (two times) and click "Register".



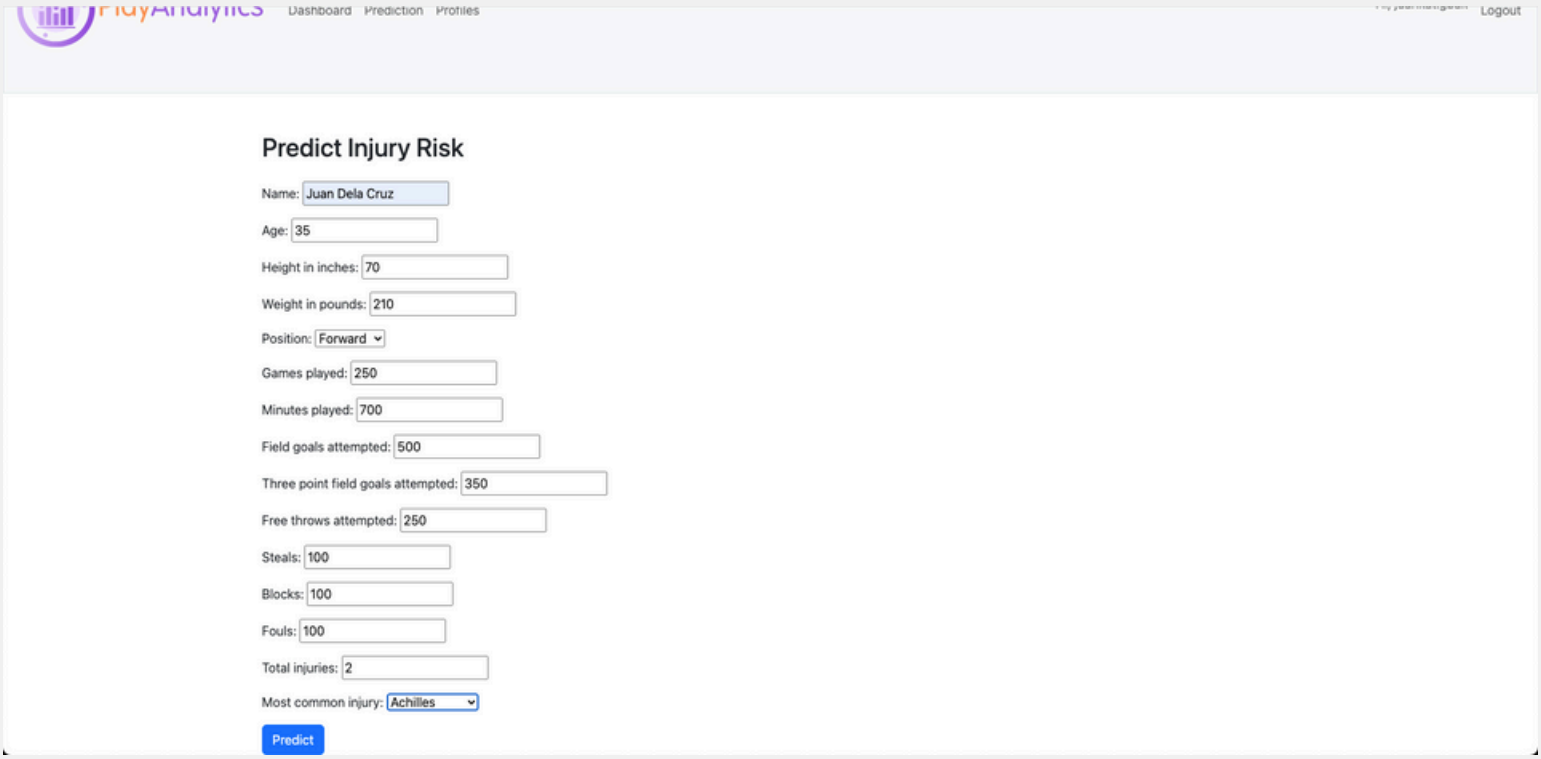
If already registered, simply input Username and Password and click "Login". This brings user to the Dashboard page.

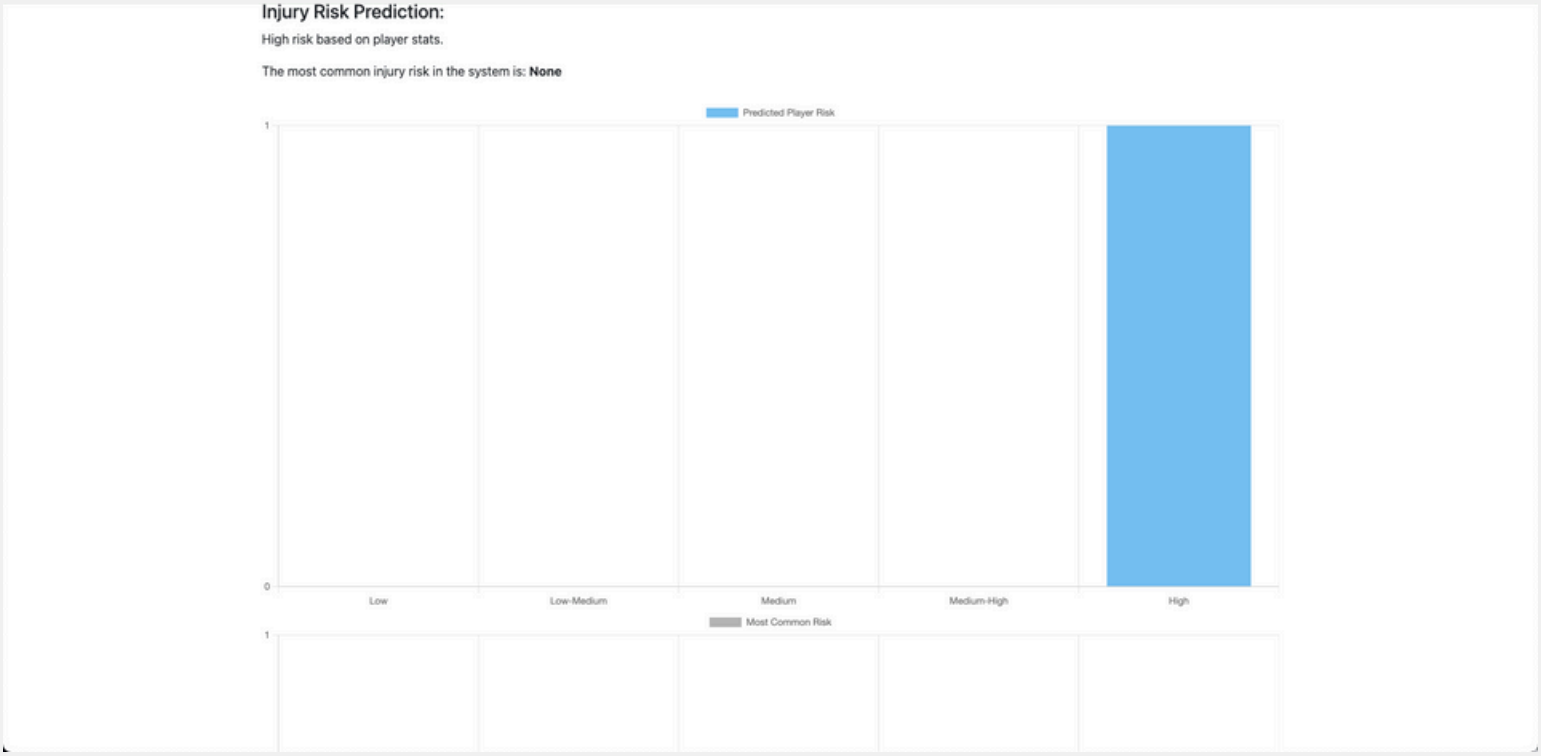


Step 3: Dashboard Page – Initially empty until players are added so click on "Prediction" in the navigation bar to go to the Prediction page.

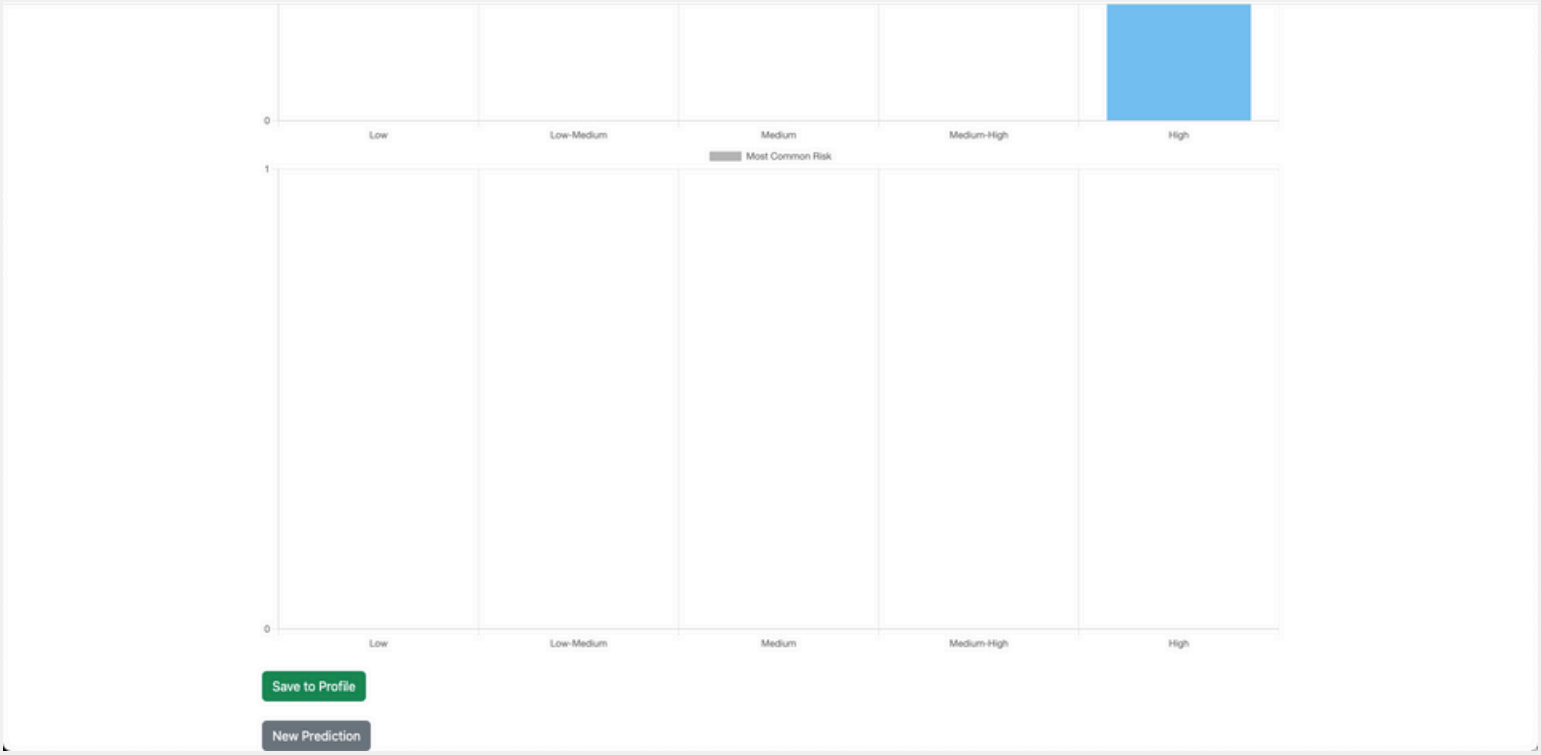


Step 4: Prediction Page – Enter player stats and injury history to get an AI-driven risk prediction by clicking "Predict". View risk bar charts.

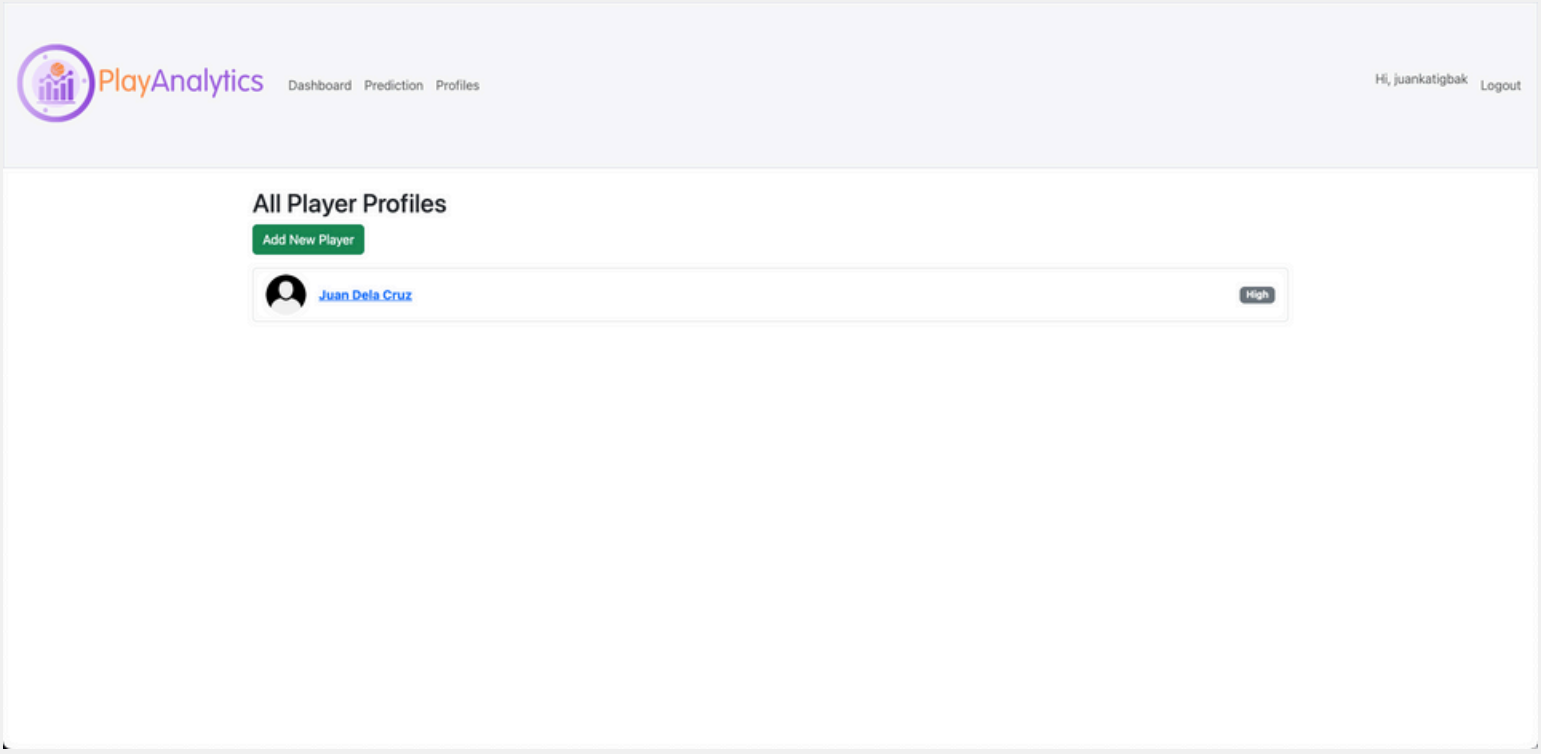




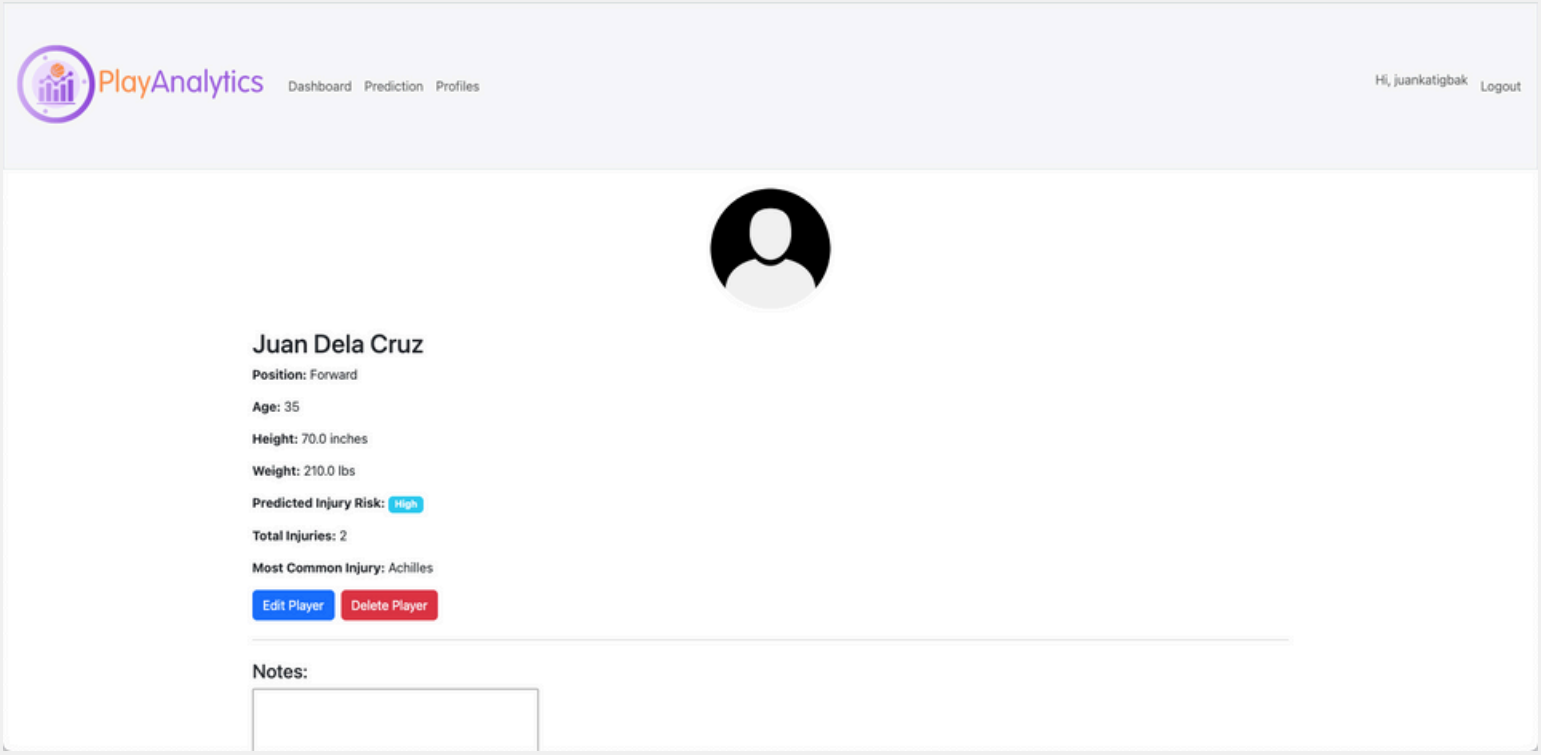
To refresh prediction, click "New Prediction" which will bring you back to the Prediction page with empty fields for inputting again. If satisfied, click on "Save to Profile".



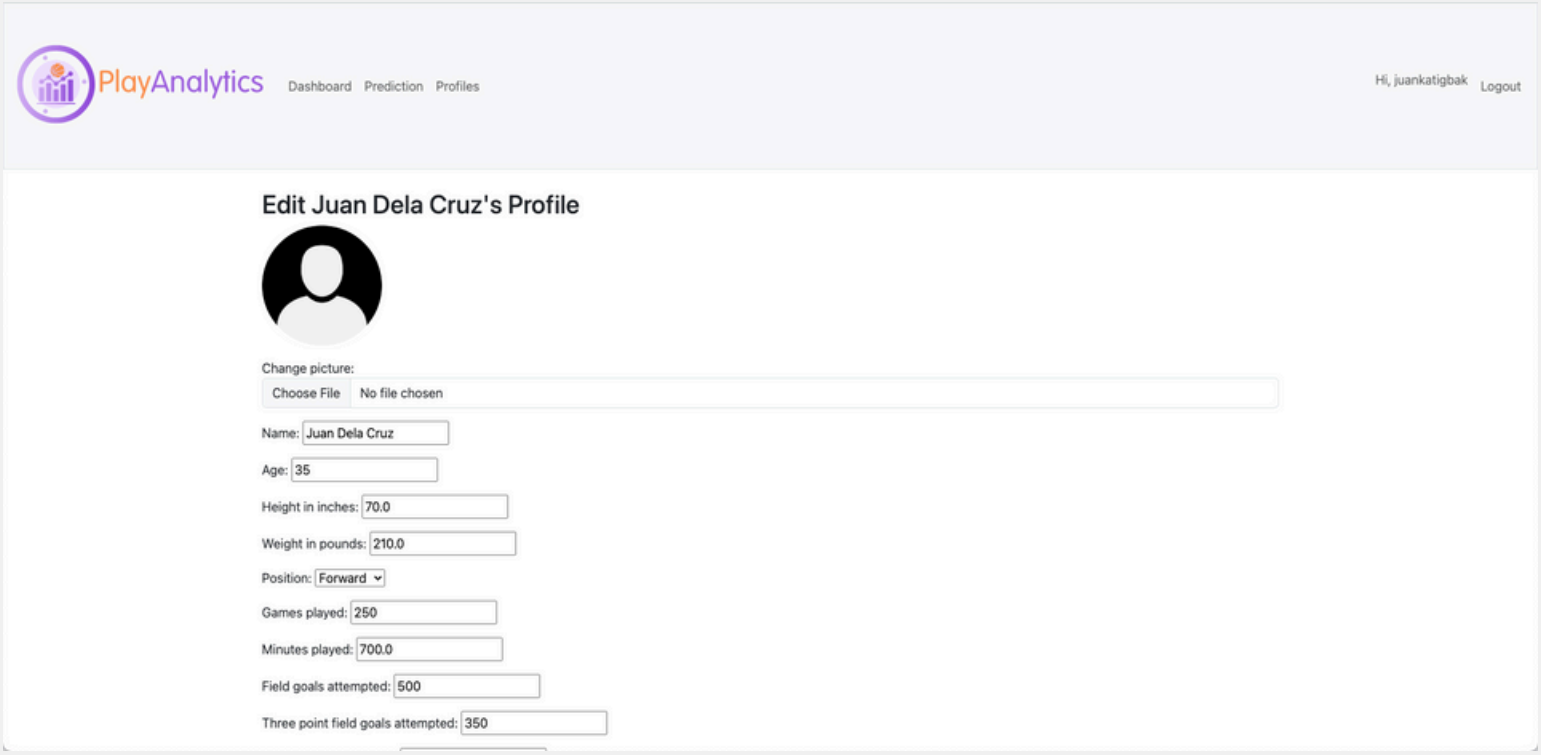
Step 5: Profiles Page (Profile Picture, Performance Statistics, and Injury History portion) – User will see the first player added in the Profiles page if "Save to Profile" had been clicked on from the Prediction page. Click on the name of the player.

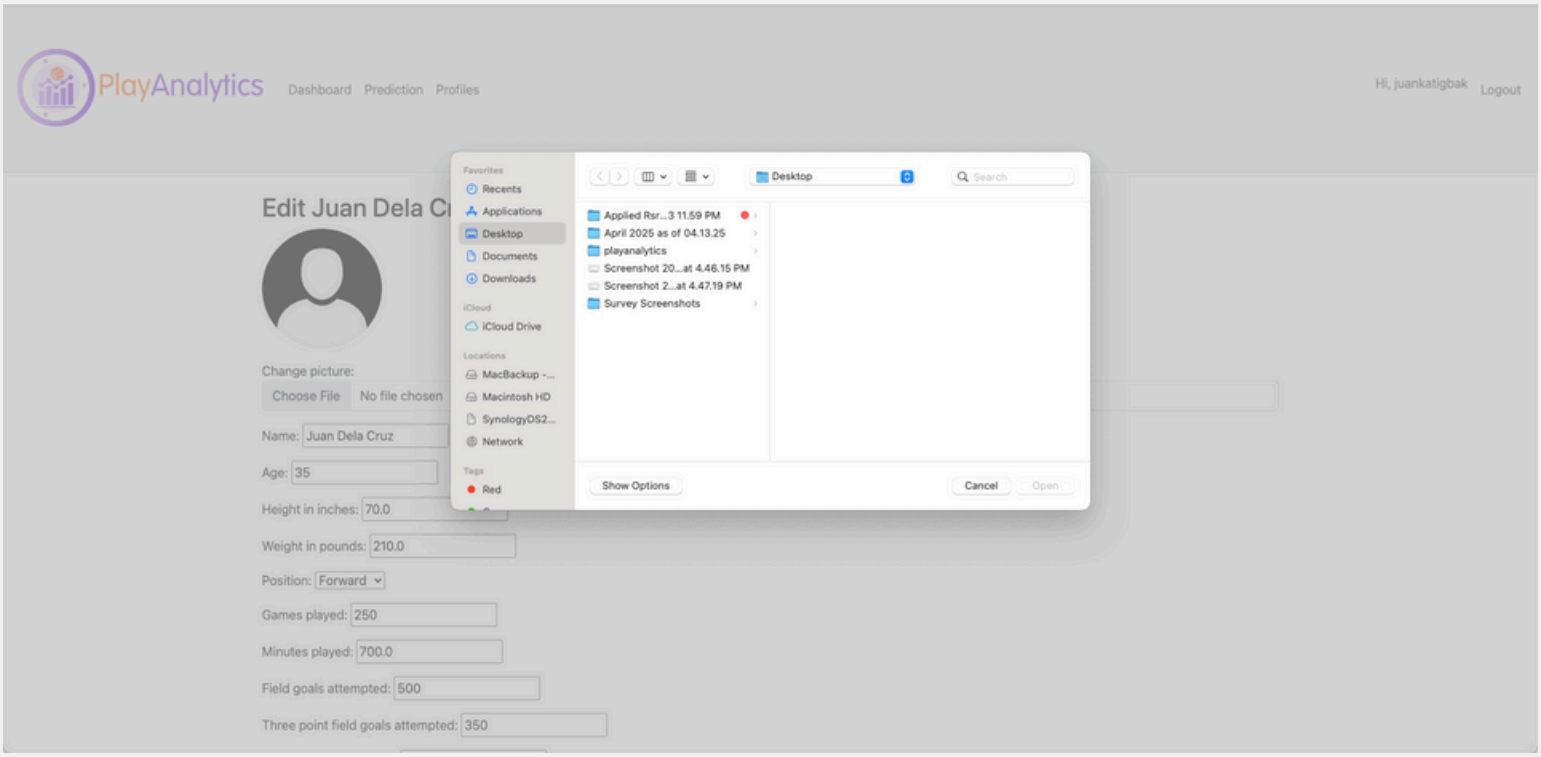


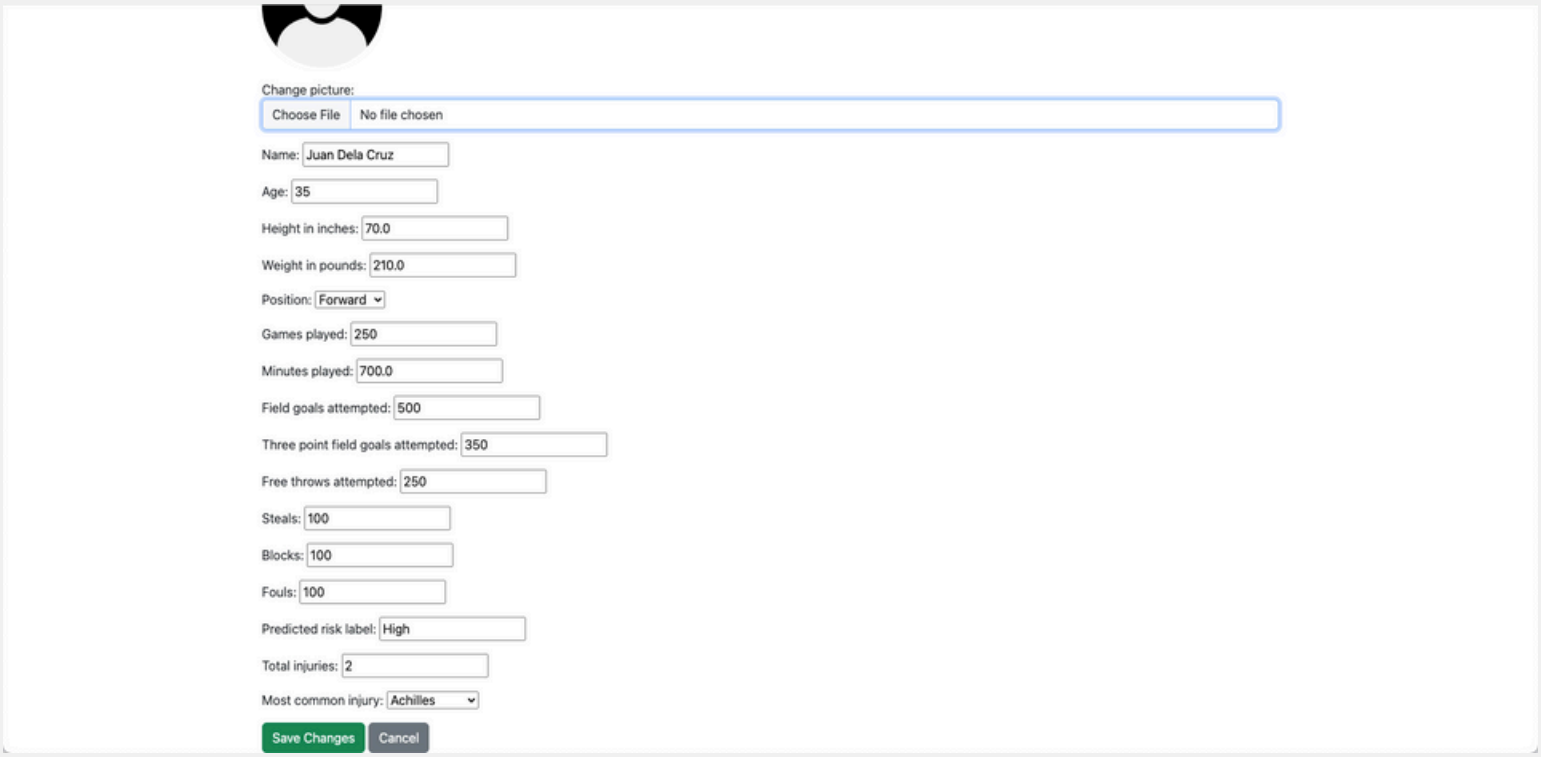
Once the user is in the Profile page of the player, click "Edit Player" so user is transported to the edit part of the player's profile.



Once there, the user can add a profile photo (which the user will see by clicking on "Choose File" and selecting a photo wherever it is located in the user's computer), and/or edit stats (note: editing performance stats does not change the predicted risk label per se but the user can change the label as long as it is inputted as either "Low", "Low-Medium", "Medium", "Medium-High", or "High" which will affect the charts in the Dashboard page that the user will be able to see. If user does not input it in those predicted risk labels, the charts in the Dashboard page will not show). If satisfied with changes, click "Save Changes", otherwise, click "Cancel". If user wants to delete the profile of the player, simply click "Delete Player" and the player will disappear from the Profile page and if that was the first one, the Profile page will be blank.

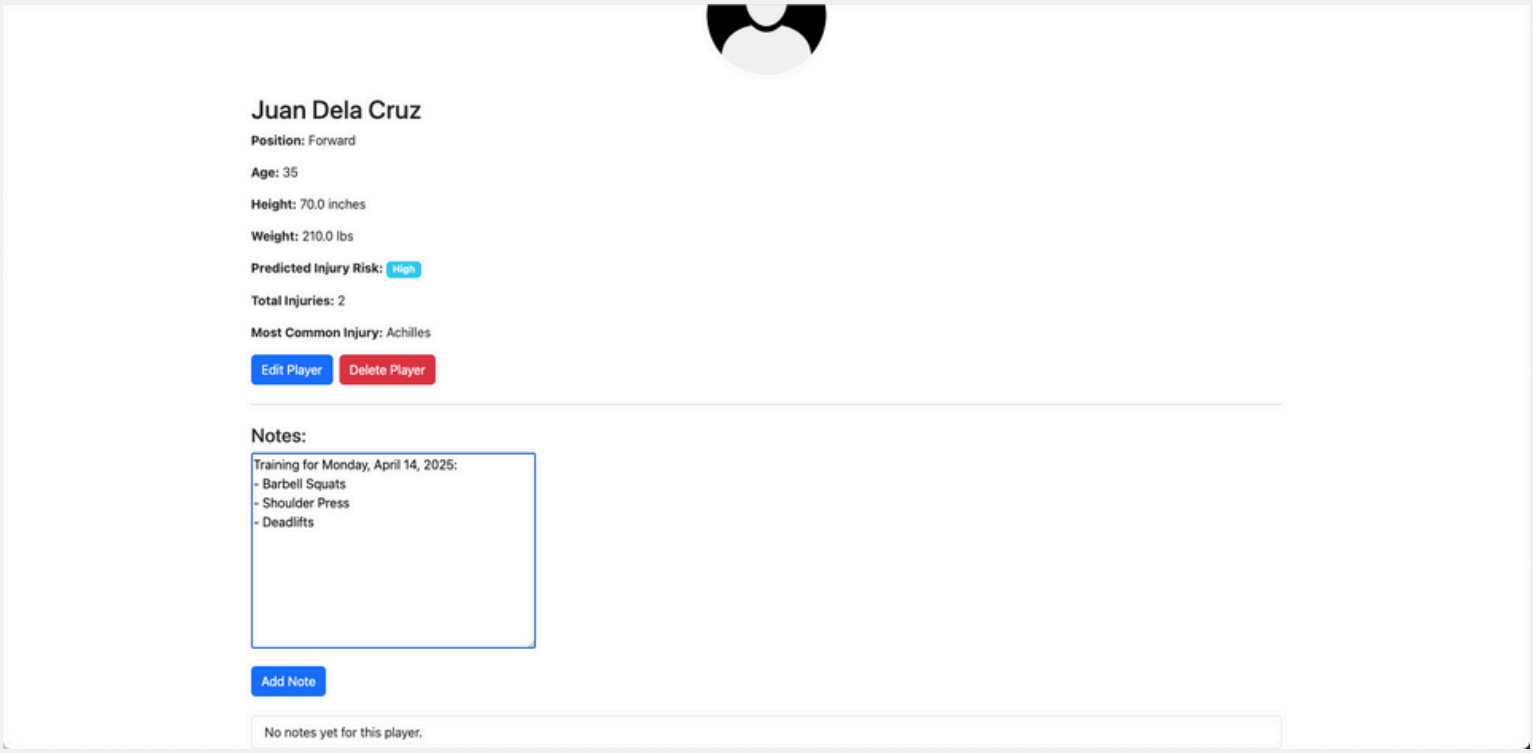


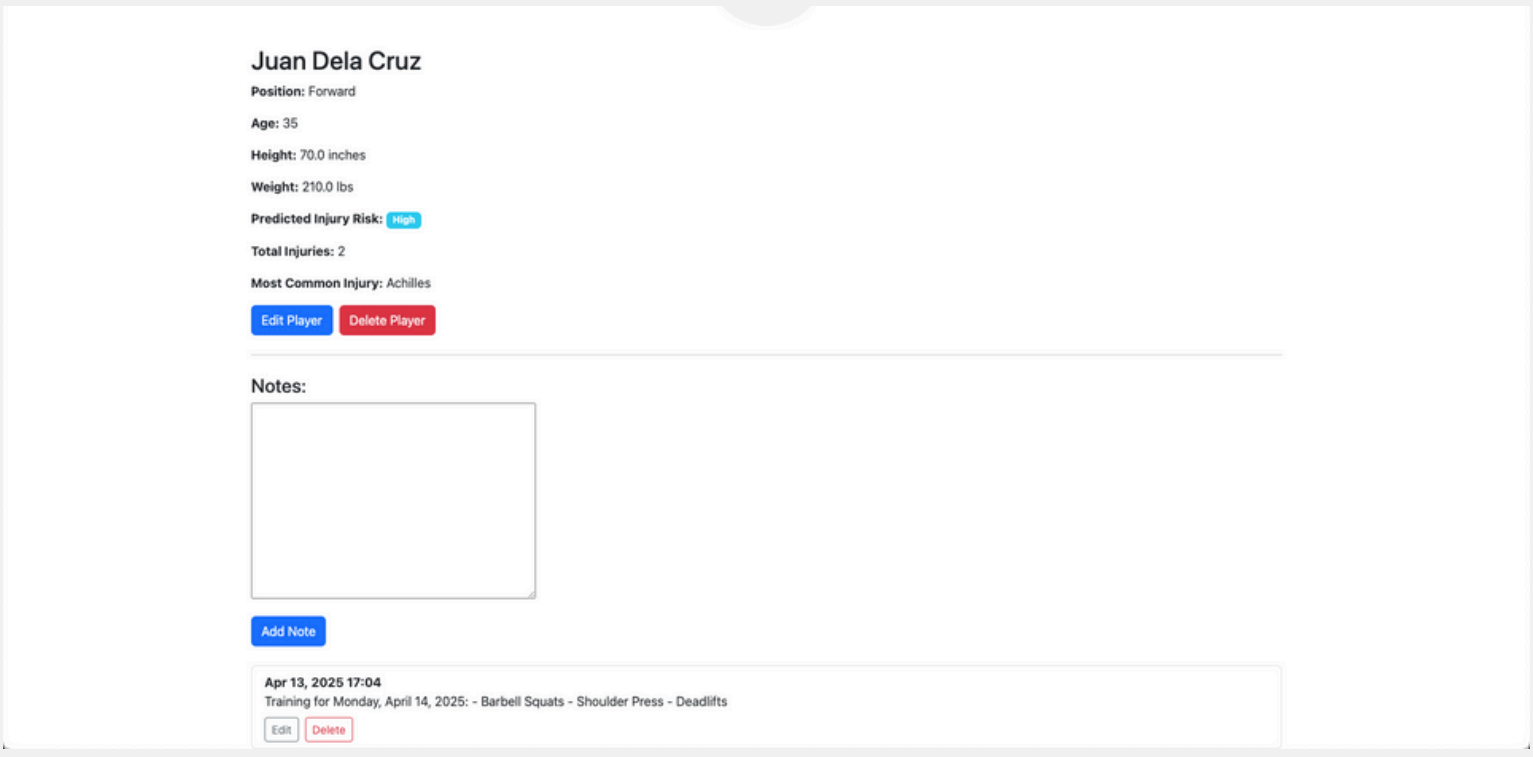


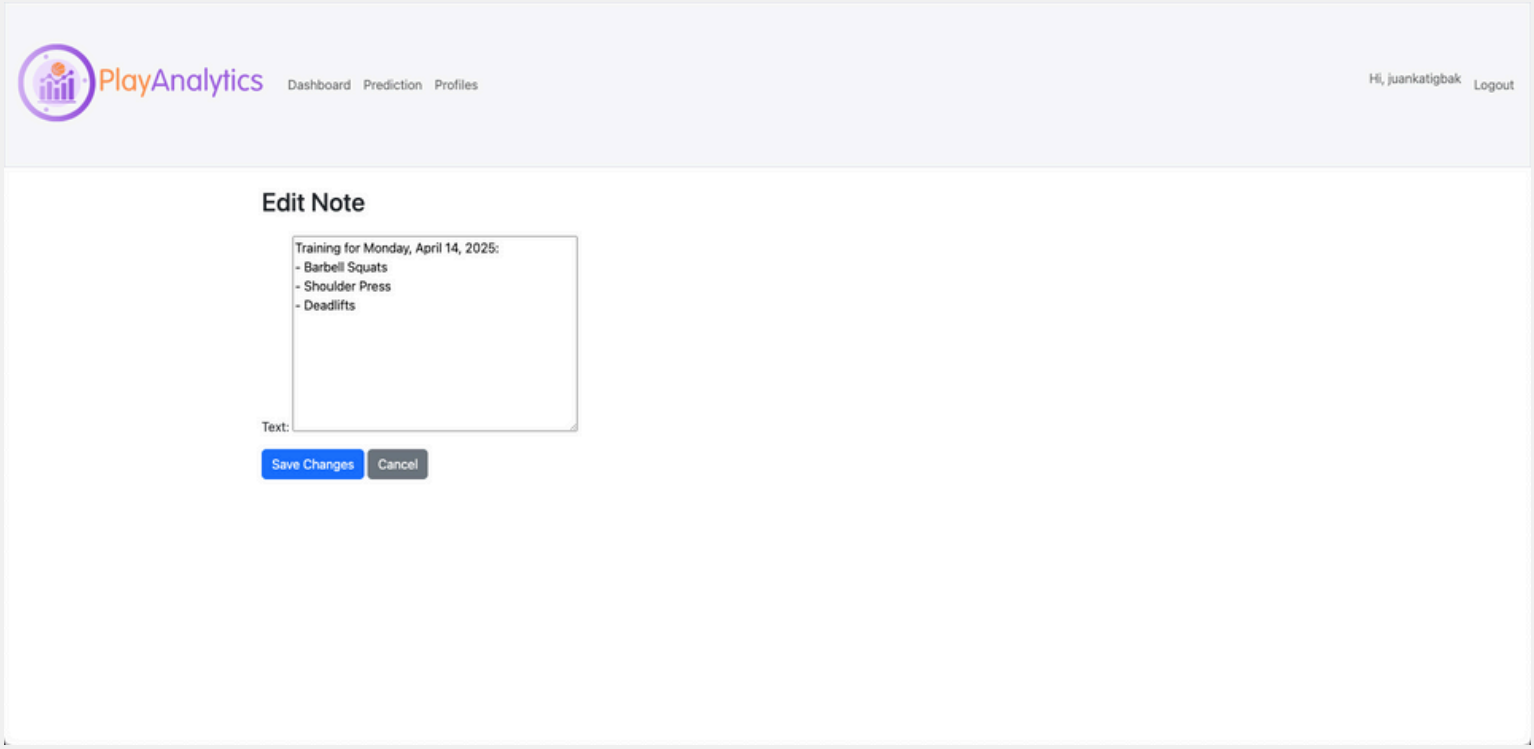


User can also bypass the Prediction page if he/she simply wants to add a player and go from there which is done by going back to the Profiles page and clicking "Add New Player". Input fields for Add New Player are the same as Edit Player.

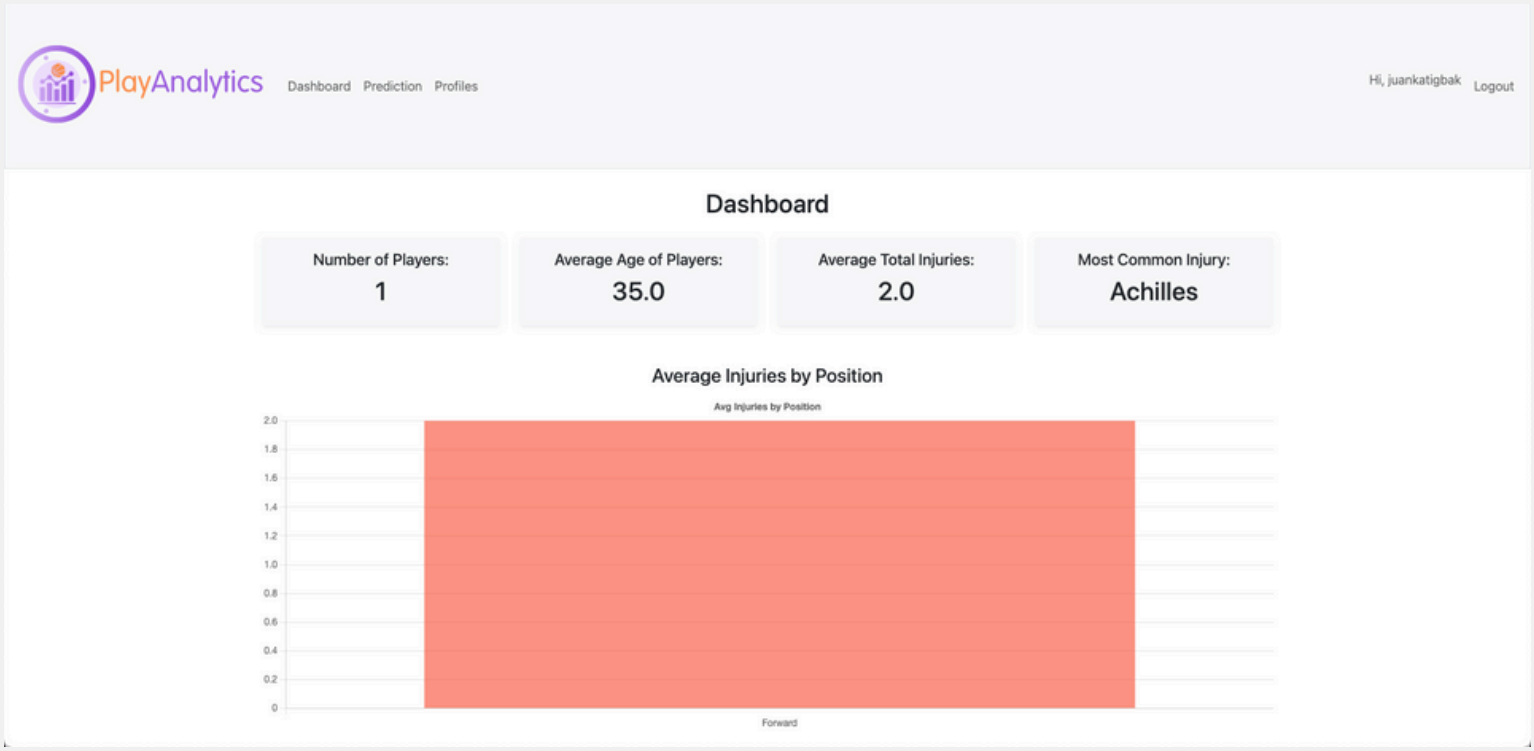
Step 6: Profiles Page (Notes portion) – User can also add notes in the Notes field. Simply write any note (like performance from a previous game, injury that took place, rehabilitation progress since injury, training regimen, food diary, journal log, thoughts at the moment, et cetera) and click on "Add Note". Entry of note will show below with date and time when note was created. To edit note if user wishes to add anything to the entry, click "Edit" and edit anything in the note field. Once satisfied, click "Save Changes" otherwise, click "Cancel". If user wants to delete the entry of note, click "Delete".

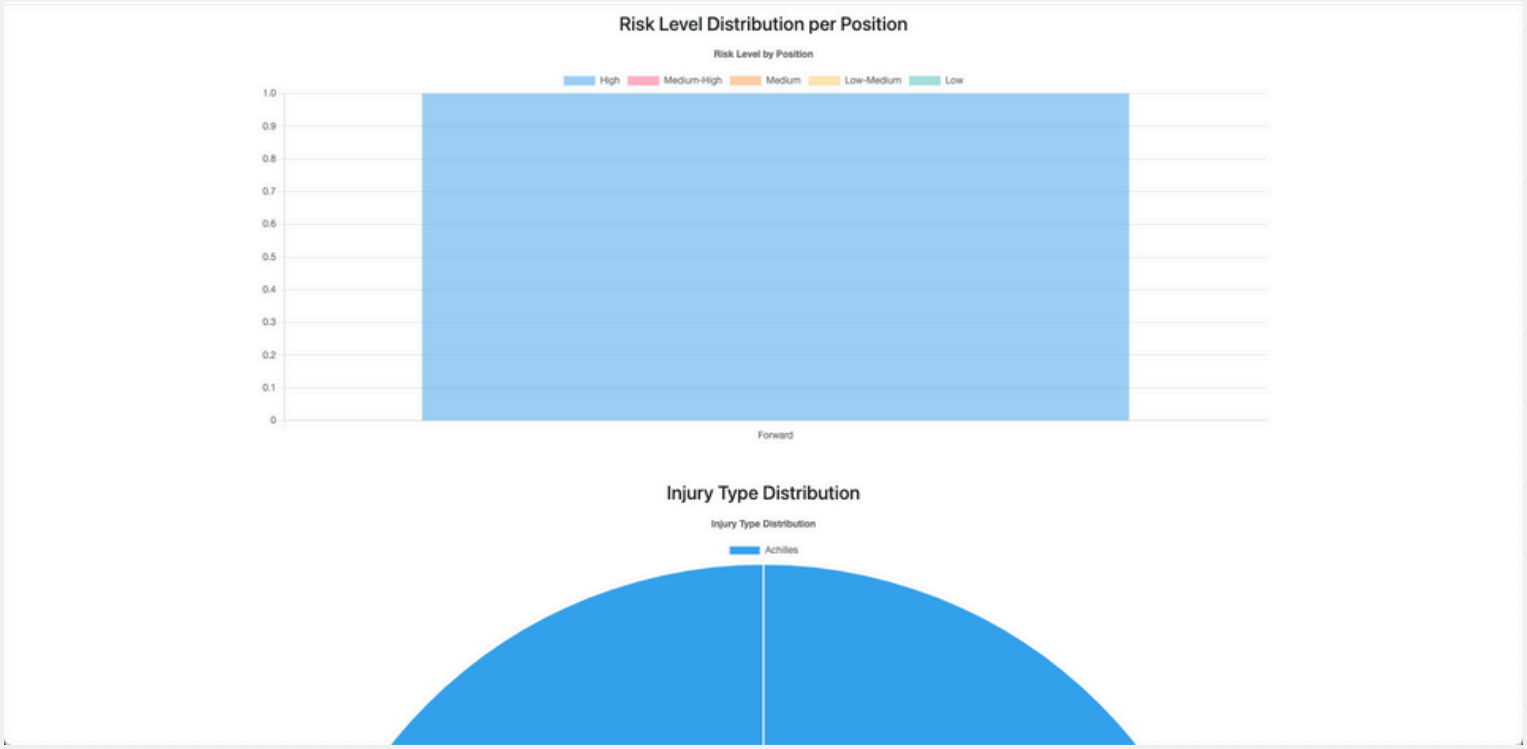


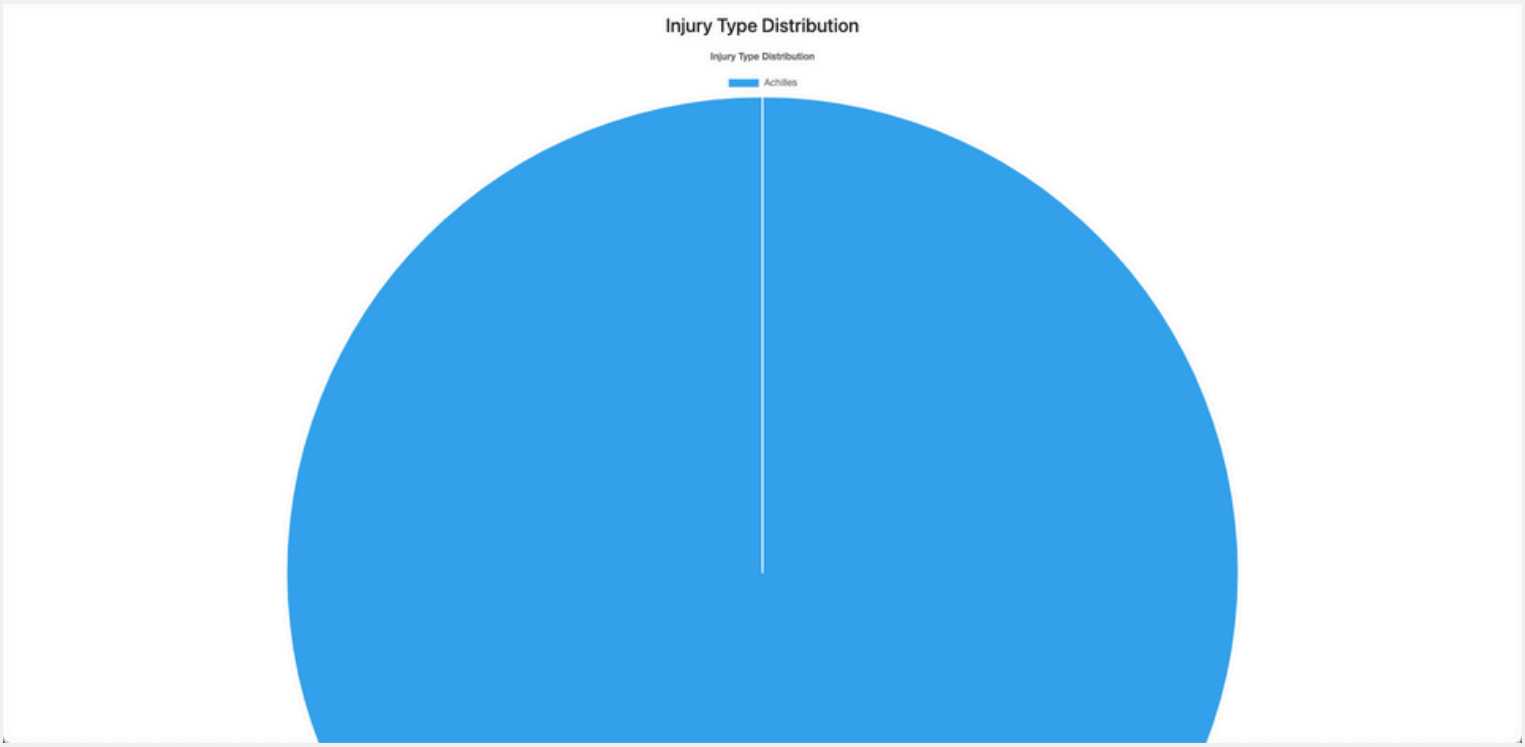




Step 7: Dashboard Page (again) – Visuals and summary indicators update in real-time as more data is added.







Note: Keep adding player data to enrich the dashboard insights and track trends.

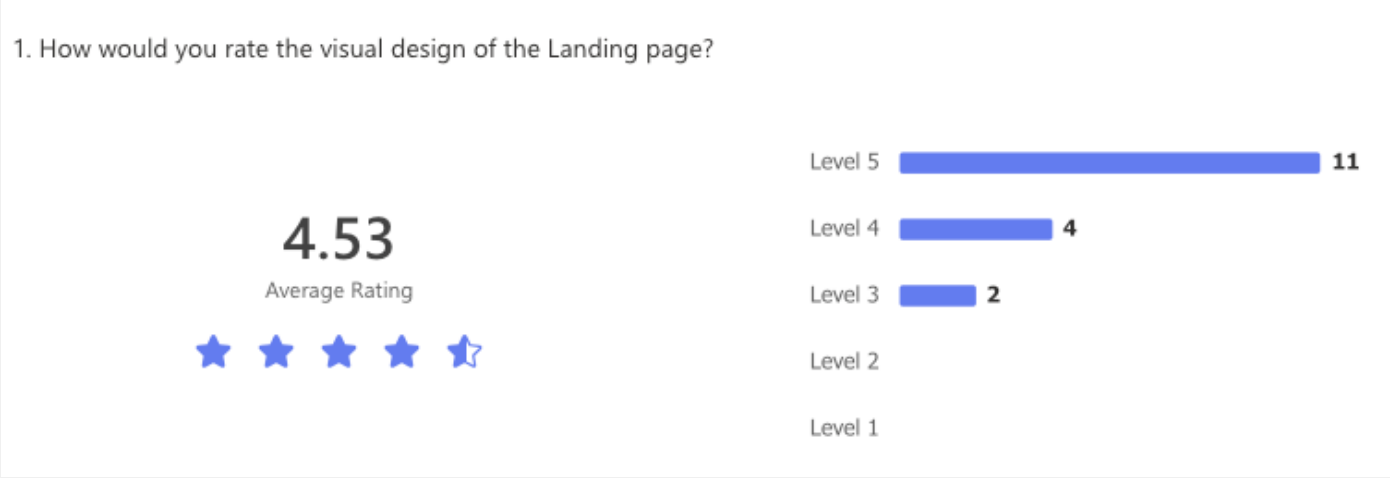
Appendix C: User Feedback Survey

1. Landing Page

Quantitative Summary
Average Rating: 4.53 / 5
Breakdown:

- Level 5: 11 respondents
- Level 4: 4 respondents
- Level 3: 2 respondents
- Levels 1–2: 0 respondents

Qualitative Summary
Participants praised the landing page for being clean, simple, and easy to navigate, with a clear purpose and intuitive layout. Suggestions for improvement included adding sports-related imagery or video loops, stronger visual branding (like a professional color palette), background information about the app, and enhancing buttons and icons for engagement and clarity—especially for first-time users and non-technical audiences.



2. Dashboard Page

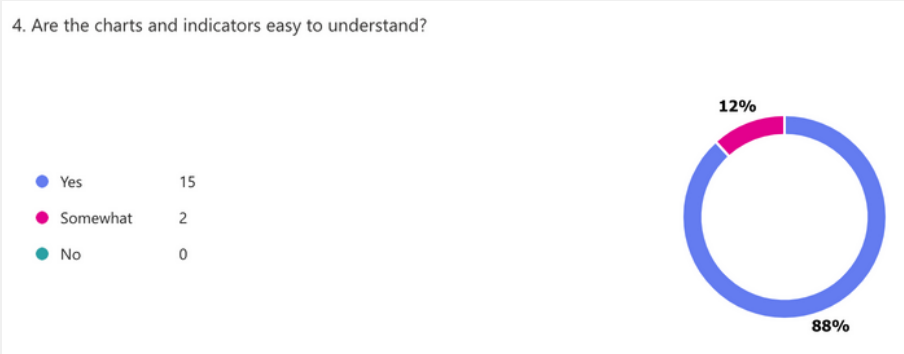
Quantitative Summary

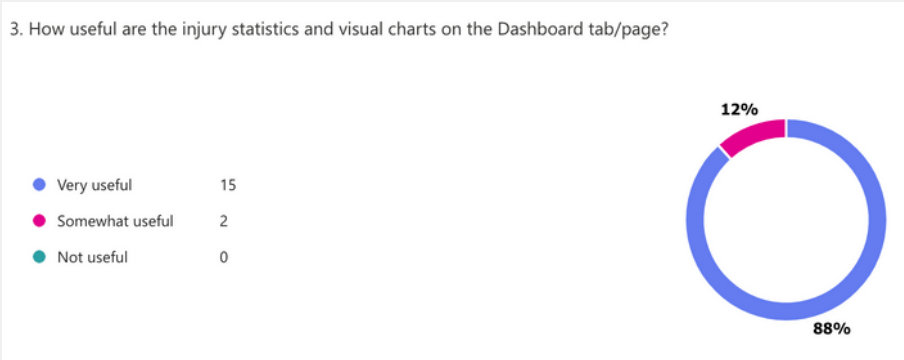
1. Usefulness of injury statistics and visual charts:
   - Very useful: 15 respondents
   - Somewhat useful: 2 respondents
   - Not useful: 0 respondents

2. Ease of understanding charts and indicators:
   - Yes: 15 respondents
   - Somewhat: 2 respondents
   - No: 0 respondents

Qualitative Summary

Users appreciated the dashboard's clean layout, clarity, and effectiveness in communicating injury-related data like total injuries and common risks. Suggestions focused on improving user experience through features like dropdown filters, trend graphs, color-coded risk levels, and better chart scaling, while others emphasized keeping the design readable and not overwhelming for first-time users.





3. Prediction Page

Quantitative Summary

1. Ease of Use:

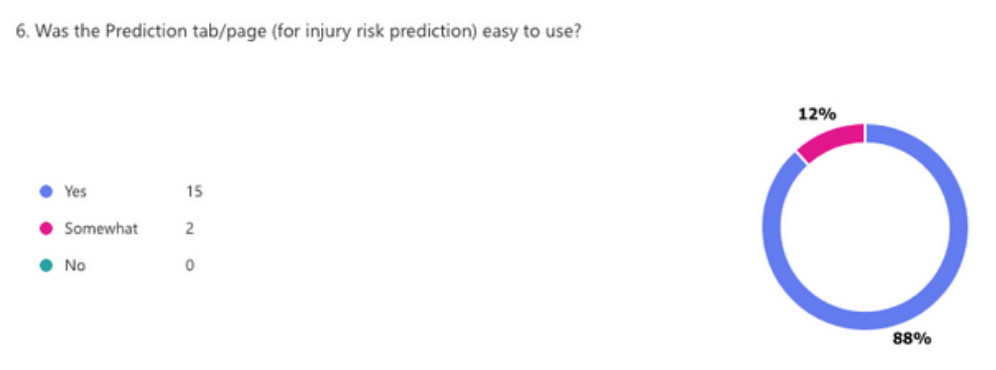
- Yes: 15 respondents
- Somewhat: 2 respondents
- No: 0 respondents

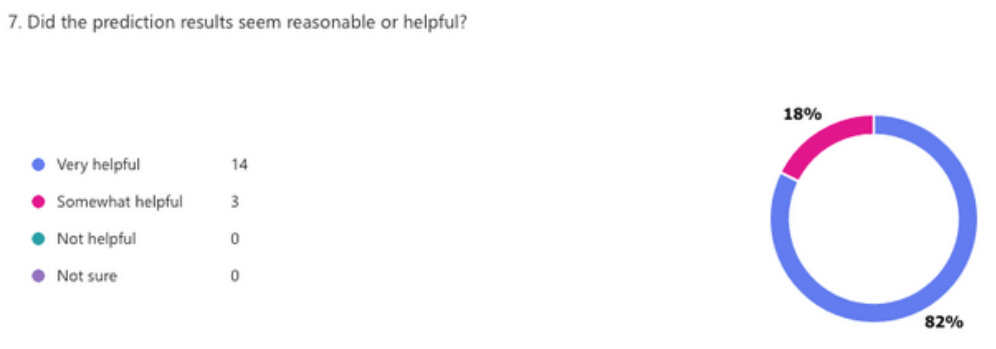
2. Usefulness of Prediction Results:

- Very helpful: 14 respondents
- Somewhat helpful: 3 respondents
- Not helpful / Not sure: 0 respondents

Qualitative Summary

Users offered insightful suggestions to enhance the prediction tool, such as adding comparative stats (e.g., minutes played vs. others), team affiliations, risk trend tracking, and heatmap visualizations. Many emphasized the need for personalized recommendations, clearer risk categories, and AI-generated advice to make the injury predictions more actionable and meaningful.





4. Profiles Page

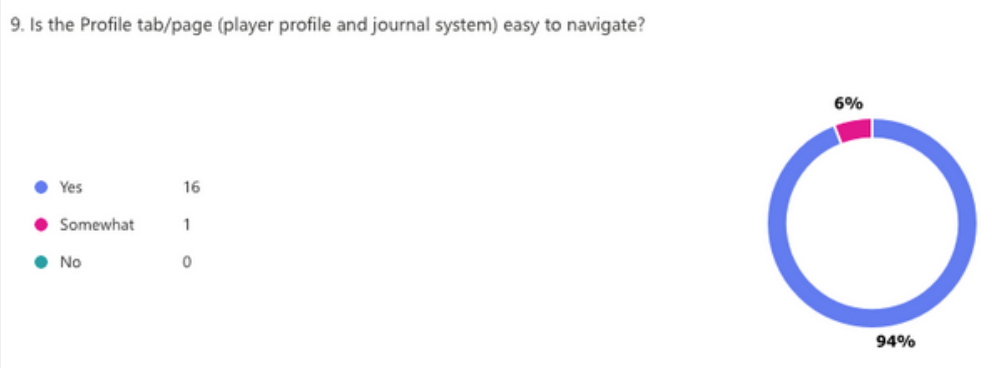
Quantitative Summary

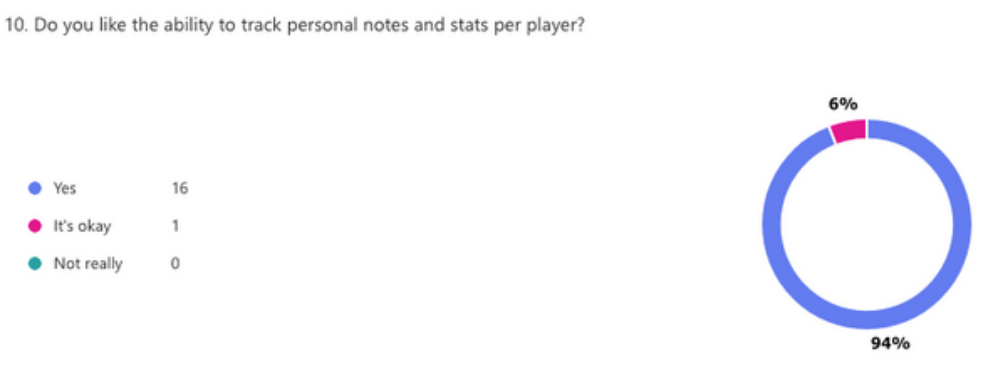
1. Ease of Navigation:
- Yes: 16 respondents
- Somewhat: 1 respondent
- No: 0 respondents

2. Usefulness of Notes and Stats Feature:
- Very helpful: 14 respondents
- Somewhat helpful: 3 respondents
- Not helpful / Not sure: 0 respondents

Qualitative Summary

Respondents found the profile page intuitive and appreciated features like journal tracking and player-specific stats. Suggestions included adding visuals, collapsible sections, dual measurement units, and personalized medical advice.



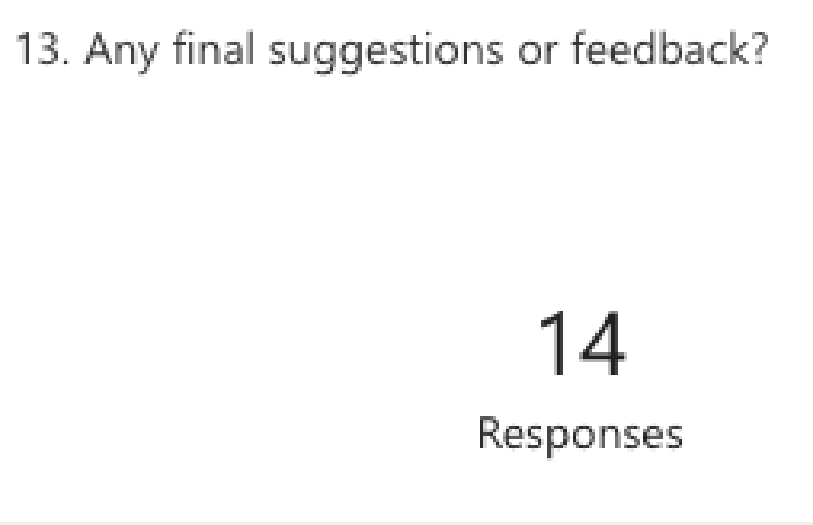


5. PlayAnalytics Overall Experience

Quantitative Summary
Average Rating: 4.65 / 5
Breakdown:

- Level 5: 12 respondents
- Level 4: 4 respondents
- Level 3: 1 respondent
- Levels 1–2: 0 respondents

Qualitative Summary
Users described PlayAnalytics as intuitive, visually clean, and helpful for understanding injury risk through AI-driven predictions. Suggestions included improving onboarding, adding player-level context (e.g., team info, pro/amateur status), refining UI clarity, and integrating personalized training or rest advice to elevate the app's long-term usefulness and professional appeal.



Appendix D: Jupyter Notebook

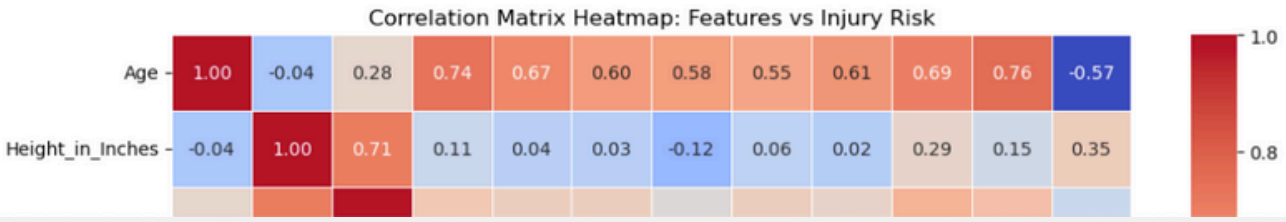
1. features.ipynb

The notebook begins by importing essential libraries and reading in the cleaned player dataset. A custom "performance load" metric is calculated by multiplying minutes played by offensive contributions (field goals, free throws, etc.). The injury risk label is heuristically assigned using a rule-based function that considers both the total number of injuries and the computed performance load. This label is then encoded numerically for model training.

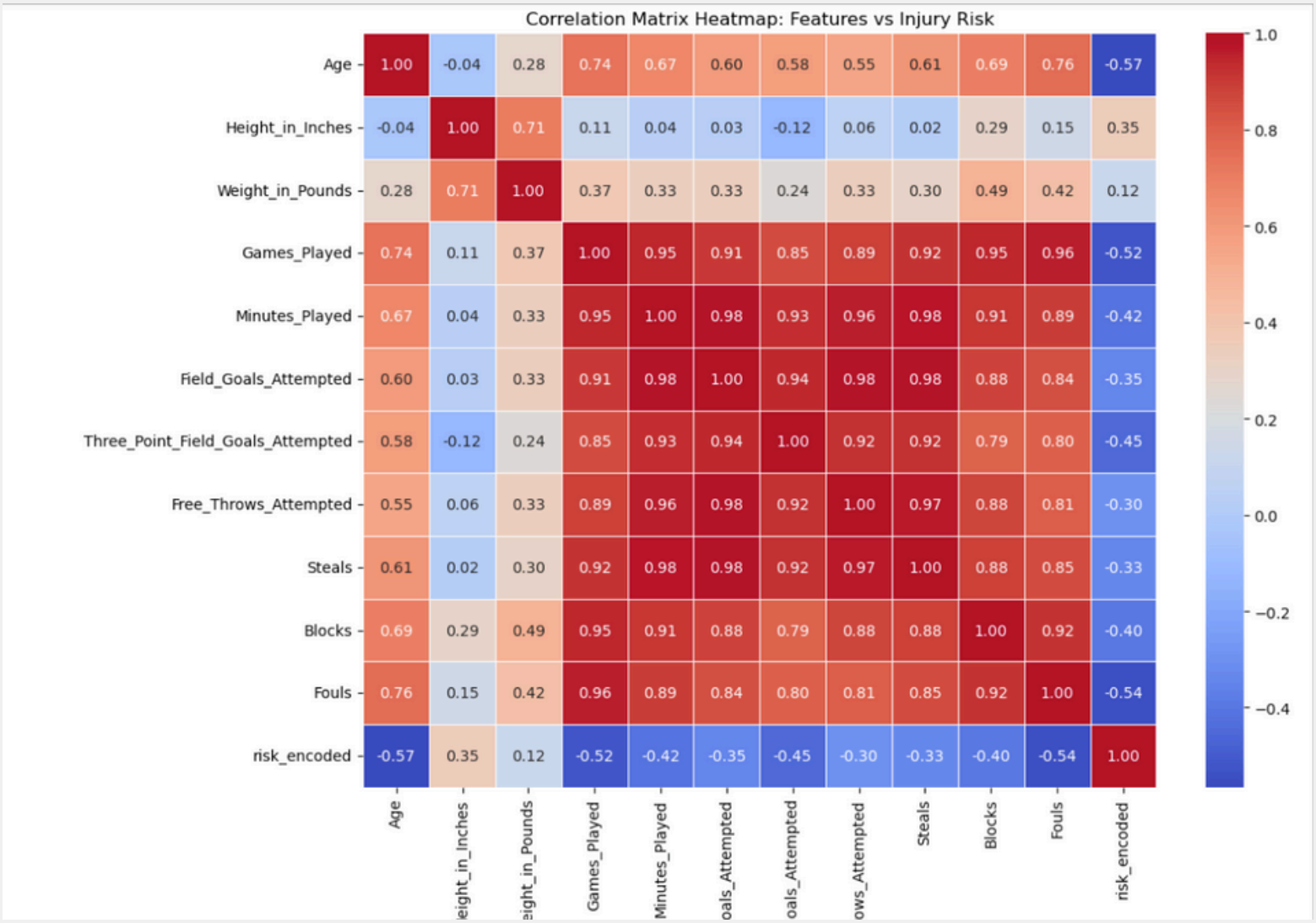
```python
# 1. Imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# 2. Load the dataset
df = pd.read_csv("player_data.csv")

# 3. Compute performance load
df["performance_load"] = (
    df["Games_Played"] * df["Minutes_Played"]
    + df["Field_Goals_Attempted"]
    + df["Three_Point_Field_Goals_Attempted"]
    + df["Free_Throws_Attempted"]
)

# 4. Heuristic label assignment
def assign_risk_label(row):
    injuries = row["Total_Injuries"]
    perf_load = row["performance_load"]
    injury_type = str(row["Most_Common_Injury"]).lower()
    if injuries <= 2 and injury_type in ["eye", "illness", "none"]:
        return "Low"
    elif 3 <= injuries <= 5 and perf_load < 10000:
        return "Low-Medium"
    elif 3 <= injuries <= 5 and perf_load >= 10000:
        return "Medium"
    elif 6 <= injuries <= 10 and perf_load >= 15000:
        return "Medium-High"
    elif injuries > 10 and perf_load >= 20000:
        return "High"

df["predicted_risk_label"] = df.apply(assign_risk_label, axis=1)

# 5. Encode target labels
label_encoder = LabelEncoder()
df["risk_encoded"] = label_encoder.fit_transform(df["predicted_risk_label"])

# 6. Define features
features = [
    "Age", "Height_in_Inches", "Weight_in_Pounds",
```

The Random Forest Classifier was selected for its robustness, ease of interpretation, and ability to handle both categorical and numerical features without scaling. After splitting the dataset (80–20), the model was trained and tested for accuracy and feature relevance.

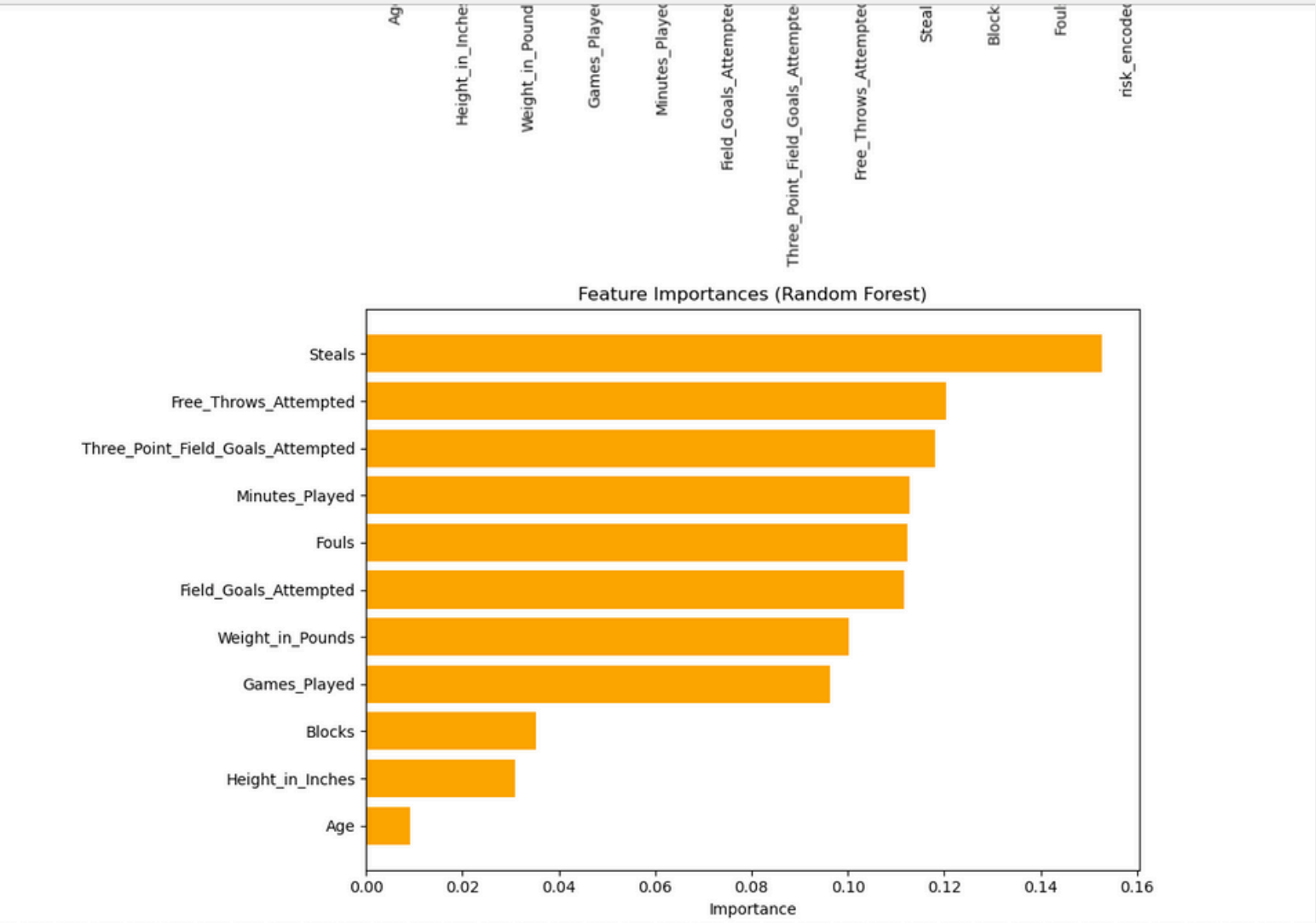
```
    "Games_Played", "Minutes_Played",
    "Field_Goals_Attempted", "Three_Point_Field_Goals_Attempted",
    "Free_Throws_Attempted", "Steals", "Blocks", "Fouls"
]

X = df[features]
y = df["risk_encoded"]

# 7. Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 8. Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# 9. Correlation Matrix Heatmap
df_corr = df[features + ["risk_encoded"]]
corr_matrix = df_corr.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Matrix Heatmap: Features vs Injury Risk")
plt.tight_layout()
plt.show()

# 10. Feature Importances Bar Chart
importances = model.feature_importances_
sorted_idx = np.argsort(importances)

plt.figure(figsize=(10, 6))
plt.barh(np.array(features)[sorted_idx], importances[sorted_idx], color="orange")
plt.title("Feature Importances (Random Forest)")
plt.xlabel("Importance")
plt.tight_layout()
plt.show()
```



To identify relationships between individual features and the target variable (encoded injury risk), a correlation matrix heatmap was generated. Strong positive correlations were observed between injury risk and performance-related metrics such as games played, minutes played, fouls, and steals. Notably, Age and Height_in_Inches showed weaker or negative correlations, justifying their lower importance in the model. This analysis helped refine feature selection and confirmed that training load indicators (e.g., fouls, steals, shot attempts) are more predictive of injury than static characteristics like height.



The final visual output was a ranked bar chart of feature importances, automatically generated from the trained Random Forest model. The model emphasized Steals, Free_Throws_Attempted, and 3-Point Field Goals Attempted as the top predictors of injury risk. These high-importance features reflect active gameplay involvement and overall load, validating the earlier heuristic logic and the correlation heatmap findings. By visualizing these results, the researcher was able to interpret model behavior and ensure alignment between domain intuition and algorithmic insight.



2. training.ipynb

To train a reliable injury risk prediction model for basketball players, the researcher began with feature engineering based on domain knowledge. As shown in the notebook screenshots, a performance_load metric was created using key in-game workload indicators (e.g., Games Played, Minutes Played, Field Goals Attempted, 3-Point FG Attempted, Free Throws Attempted). The model then assigned risk labels (Low, Medium, High, etc.) using a heuristic function that factored in both total injuries and performance load, as visualized in the class distribution output, which revealed significant class imbalance (e.g., 630 High vs. 73 Medium-High).

[67]:
```python
import pandas as pd

# Load your dataset
df = pd.read_csv("player_data.csv")
```

[69]:
```python
df["performance_load"] = (
    df["Games_Played"] * df["Minutes_Played"]
    + df["Field_Goals_Attempted"]
    + df["Three_Point_Field_Goals_Attempted"]
    + df["Free_Throws_Attempted"]
)
```

[71]:
```python
def assign_risk_label(row):
    injuries = row["Total_Injuries"]
    injury_type = str(row["Most_Common_Injury"]).lower()
    perf_load = row["performance_load"]

    if injuries <= 2 and injury_type in ["eye", "illness", "none"]:
        return "Low"
    elif 3 <= injuries <= 5 and perf_load < 10000:
        return "Low-Medium"
    elif 3 <= injuries <= 5 and perf_load >= 10000:
        return "Medium"
    elif 6 <= injuries <= 10 and perf_load >= 15000:
        return "Medium-High"
    elif injuries > 10 and perf_load >= 20000:
        return "High"
    else:
        return "Medium"
```

[73]:
```python
df["predicted_risk_label"] = df.apply(assign_risk_label, axis=1)
```

[75]:
```python
print(df["predicted_risk_label"].value_counts())
```

```
predicted_risk_label
High           630
Medium         297
Medium-High     73
```

To prepare the data for machine learning, labels were encoded numerically and an 80-20 train-test split was applied. Three classification algorithms were then benchmarked: Random Forest, Support Vector Machine (SVM), and XGBoost.

[75]:
```
print(df["predicted_risk_label"].value_counts())
```

```
predicted_risk_label
High           630
Medium         297
Medium-High     73
Name: count, dtype: int64
```

[77]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df["risk_encoded"] = label_encoder.fit_transform(df["predicted_risk_label"])
```

[79]:
```
features = [
    "Age", "Height_in_Inches", "Weight_in_Pounds", "Games_Played", "Minutes_Played",
    "Field_Goals_Attempted", "Three_Point_Field_Goals_Attempted",
    "Free_Throws_Attempted", "Steals", "Blocks", "Fouls"
]

X = df[features]
y = df["risk_encoded"]
```

[81]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

[83]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report

models = {
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "SVM": SVC(probability=True, kernel='rbf', C=1),
    "XGBoost": XGBClassifier(eval_metric='mlogloss', random_state=42)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
```

As shown in the screenshots of the accuracy and classification reports, Random Forest outperformed the others, achieving 96.5% accuracy with strong precision and recall for the "High" and "Medium" risk categories, and slightly better handling of the underrepresented "Medium-High" class. SVM and XGBoost also achieved the same accuracy rate but had zero precision and recall for some classes due to the imbalance, triggering UndefinedMetricWarning.

[83]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report

models = {
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "SVM": SVC(probability=True, kernel='rbf', C=1),
    "XGBoost": XGBClassifier(eval_metric='mlogloss', random_state=42)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    print(f"\nModel: {name}")
    print("Accuracy:", accuracy_score(y_test, preds))
    print("Classification Report:\n", classification_report(
        y_test, preds, target_names=label_encoder.classes_
    ))
```

```
Model: Random Forest
Accuracy: 0.965
Classification Report:
               precision    recall  f1-score   support

         High       0.99      0.98      0.99       141
       Medium       0.94      1.00      0.97        51
  Medium-High       0.57      0.50      0.53         8

     accuracy                           0.96       200
    macro avg       0.84      0.83      0.83       200
 weighted avg       0.96      0.96      0.96       200


Model: SVM
Accuracy: 0.855
Classification Report:
               precision    recall  f1-score   support

         High       0.83      1.00      0.91       141
       Medium       0.97      0.59      0.73        51
  Medium-High       0.00      0.00      0.00         8

     accuracy                           0.85       200
    macro avg       0.60      0.53      0.55       200
 weighted avg       0.83      0.85      0.83       200
```

Based on accuracy, robustness against imbalanced data, and interpretability, Random Forest was chosen as the final model and exported using joblib for deployment in the Django web application.

```
   macro avg       0.60      0.53      0.55       200
weighted avg       0.83      0.85      0.83       200


Model: XGBoost
Accuracy: 0.965
Classification Report:
               precision    recall  f1-score   support

        High       0.99      0.98      0.99       141
      Medium       0.94      1.00      0.97        51
 Medium-High       0.57      0.50      0.53         8

    accuracy                           0.96       200
   macro avg       0.84      0.83      0.83       200
weighted avg       0.96      0.96      0.96       200

/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
[85]: import joblib

# Save the best model (Random Forest) and label encoder
joblib.dump(models["Random Forest"], "rf_injury_model.joblib")
joblib.dump(label_encoder, "risk_label_encoder.joblib")

print("✅ Model and label encoder saved as .joblib files!")
```

```
✅ Model and label encoder saved as .joblib files!
```

```
[ ]:
```