AN INTELLIGENT MOBILE APPLICATION TO DIAGNOSE INJURIES AND RECOMMEND TRAININGREGIMENS USING MACHINELEARNING, NATURAL LANGUAGE PROCESSING, ANDCOMPUTER VISION

Daniel Zhang [1], Jonathan Thamrun [2]

[1] Beckman High School, 3588 Bryan Ave, Irvine, CA 92602
[2] California State Polytechnic University, Pomona, CA, 91768

ABSTRACT

Injuries are common among both competitive and casual runners. Traditionally, patients seek medical experts like physiotherapists to treat their ailments. However, this can be expensive and time-consuming. To address this, we created an AI-driven mobile application that utilizes Natural Language Processing (NLP) and Computer Vision to diagnose users and recommend treatments based on user reported symptoms and uploaded images. This application was built using the Flutter framework and leverages Firebase for data storage. Additionally, it uses the OpenAI API to request OpenAI's gpt 4.1. In experimentation, we found that the model could detect injuries in knee X-Rays with about 64% accuracy and generate responses in around 82 seconds on average. These findings suggest that our method's accuracy is comparable to other methods, and that our response times are superior. Although our method isn't meant to replace medical experts, it can act as a swift, mobile first opinion for injured runners.

KEYWORDS

Machine Learning, Computer Vision, Nature Language Processing

1. INTRODUCTION

Running has become a popular form of exercise in the modern world. It is a simple activity with primal roots that make it second nature to most. In ancient Africa, early humans were endurance hunters who outran their prey because they could simply run longer [1]. Despite its ancient origins, the modern trend of running, particularly marathoning, began in the 1970s. This came after Frank Shorter won the Olympic marathon in 1972 and Jim Fixx published his Complete Book of Running 5 years later [1]. This inspired Americans to begin running as a hobby or even a sport. Additionally, increases in obesity, overweightness, and other health concerns encouraged people to begin running for its numerous health benefits and ability to aid in weight loss [2]. As a result, millions of Americans engage regularly with the activity. According to runningwithgrit.com, "approximately 60 million people participated in running, jogging and trail running" in the United States alone [3]. That equates to around 10%-20% of the US population, a percentage supported by other papers [4]. As a result, plenty of injuries occur, both chronic and acute. According to one paper, they occur in about 40%-50% of runners on an annual basis. In fact, 25% of runners are injured at any given moment [4]. Often, runners don't have immediate access to proper medical care which can diagnose their ailments, treat them, and prevent recurrence. This has numerous negative effects. First, runners may experience intense physical

David C. Wyld et al. (Eds): NLMLT, SEAU, ITCS, CST – 2025
pp. 45-56, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151605

pain for longer than they have to. Additionally, competitive runners may be unable to train for or compete in important races. This is an even bigger problem for athletes from underfunded programs who lack proper coaching and medical care. Athletes have very short primes, with mid-distance runners having only 4 years of peak performance [5]. For these runners, race time is precious and injury is their biggest fear.

In this paper, we compared our application with 3 different methodologies. The first was a Deep Gaussian Covariance Network (DGCN) trained for running injury prediction. This model utilized a more comprehensive range of data points to make its predictions. It had access to bloodwork and data from IMUs. Although this should result in more accurate results, it makes the model harder to use and less accessible. Our application is superior in that it's more mobile, cheaper, and doesn't require any physical equipment. The next methodology used MoCap (Motion Capture) technology to analyze a user's running form and recommend changes to prevent injury. While the MoCap technology used in this method is accurately able to record running form, it requires a medical professional to review the data and provide feedback. Although this gives the results full medical credibility, it results in longer wait times for the user. Our application is capable of responding in a matter of minutes or even seconds, making it quick and easy to use. Finally, another methodology trained a random forest AI model to view which aspects of a triathlete's running gait contributed most to injury, and to predict whether a triathlete had been injured recently. The model utilized data collected from IMUs worn by the triathletes while they ran and predicted injuries with around 68% accuracy. Our solution had a 64% accuracy while predicting injuries based solely on X-Ray images. Most importantly, our application doesn't require any physical equipment to make diagnoses. If the user has no scans, they can fill out some simple questionnaires and still get usable results.

This application attempts to solve that problem by providing an easy to use, portable and swift solution which can diagnose injuries and recommend treatments using a variety of data points. Our mobile application allows the user to input their pain levels in each region of their body, attach images of MRI or X-ray imaging and add additional comments to be utilized by a Machine Learning model to create a list of possible diagnoses. The model uses all available data points to attempt a diagnosis and recommend treatments. The user is given a variety of advice. They will be notified of whether they need a medical professional, if they should still engage in training, and may also be prescribed strengthening exercises or stretches among other treatments. Our solution also provides weekly running plans for the user adapted to their level of injury. If the user follows the app's prescription, they are expected to experience faster recovery and reduced risk of reinjury afterwards. Although medical professionals may be more accurate and reliable, our AI-driven solution is significantly quicker and more accessible, making it a great tool to use for minor injuries or as a first opinion. While our application typically produces a full result in under 1-2 minutes (see experiments for more accurate times) and can be accessed immediately from the user's phone, a user would have to wait days or even weeks to be diagnosed by a doctor. Additionally, our application is significantly cheaper than seeing a medical professional or seeing a traditional physical therapist.

In our first experiment, we tested the accuracy of the AI model used by our application. More specifically, we tested its ability to accurately diagnose abnormalities in knee X-Rays by comparing its conclusions with the conclusions of medical experts. To automate requesting the model, we wrote a python script to emulate the requests made by our application. We also used a different, much shorter prompt to limit tokens and save costs. We ultimately found the model we used, gpt 4.1, had an approximately 64% accuracy. Next, we decided to test the speed of the model by recording its response times. Like in the first experiment, we used a python script for automation purposes, but used the original prompt to emulate our application as best as possible. We did multiple trials on 2 different days, and found that results varied greatly between the two

days. On the worst day (Day 1), response times were on average 151 seconds. On the best day (Day 2), response times were on average 13.5 seconds. This was due to varying amounts of traffic on OpenAI's servers.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Managing AI Costs and Limits in Diagnostic Apps

One major component of the application is the generative AI used to diagnose the user and recommend treatments. The AI is an OpenAI model which is accessed via the OpenAI API (Application Programming Interface). The application specifically utilizes gpt 4.1, a LLM (Large Language Model) that is able to interpret and respond to human language. The model also has computer vision capabilities, allowing it to interpret image input. This results in some critical issues. In order to prevent overloading/abuse of its services, OpenAI must put limits on requests and charge users based on the length and frequency of their requests. For instance, gpt-4.1 can only be requested at a rate of 500 RPM (requests per minute). If too many users utilize the application at once, our organization's account will be temporarily restricted from further requests. This could result in outages and difficulty for users. To help reduce outages, users of the application could be limited to only a few diagnoses a day, maybe even less. Additionally, a high number of users will also mean high costs for the organization operating the OpenAI API account. To reduce costs for our organization, we could shorten our prompt and limit the model's output. OpenAI charges users based on the number of input tokens and the number of output tokens. A token is a group of characters. Generally, tokens are about 4 English characters, but there are a variety of different tokenization methods that group characters in different ways resulting in different lengths. By shortening the prompt and prompting the model to reduce its output, we could increase cost-efficiency.

2.2. Securing User Data in Firebase

Another major component of the application would be the database it uses to store user information. Our application uses Firebase, a Google owned service which allows for the storing of text, images, and all sorts of data on the cloud. We use it to store user account information, user submitted images, and user settings. This results in a risk for user privacy, since the database could be accessed without permission and sensitive data could be leaked. This is a common risk among applications which store data remotely. To reduce this risk, multi-factor authentication and stronger passwords could be put on accounts with access to our database on firebase. Firebase also has tools to secure databases and stricten permissions which we could use to further improve security.

2.3. Improving UI Accessibility and Usability

Finally, the most visible component of the application is the User Interface (UI). The app's UI consists of multiple pages and menus which allow the user to perform a variety of actions and interact with the application's backend. Although the UI isn't very complex or cluttered, it could pose a challenge to some users. Some users may be confused by its design, frustrating them and preventing them from fully using the application and its features. To remedy this, an onscreen tutorial could be implemented, and the UI could be redesigned for simplicity and readability. Some pages could also be removed, and text could be made larger to accommodate those with poor vision.

3. SOLUTION

We used Flutter, Firebase, and the OpenAI API to create this program. Flutter is a mobile app development framework we used to make development and distribution easier. Flutter allows developers to port their application to multiple platforms from a single codebase. It makes development convenient by providing a variety of premade UI elements, and is compatible with a variety of google services. Firebase and the OpenAI API were external services we used to give our application essential functions. There are 3 major components to this program: the authentication section, symptoms inputting, and AI recommendations. First, the authentication section refers to the Log-in and Sign-up pages displayed to the user. These are the first pages they see upon entry into the application. Here, they may either create or login to an existing account. Once the user enters their account information, Firebase is used to authenticate the user. If Firebase confirms that the user's login information is correct, they are sent to a main menu screen **[6]**. From here, they may access the symptoms inputting section by pressing "Diagnose." They are sent to a diagnosis screen where they are given multiple tabs to input various data points like images and pain levels across their body. Once they are finished reporting their symptoms, they may press confirm and be sent to the results page. This is where they encounter the AI recommendations section of the program. They are shown a list of diagnoses by the AI among a variety of recommendations. This is generated by sending a request to the OpenAI API containing the user's information and a premade prompt **[7]**. Finally, the user may check other recommendations by returning to the main menu, and viewing the rehab plan and training plan. These recommendations are generated in the same manner.

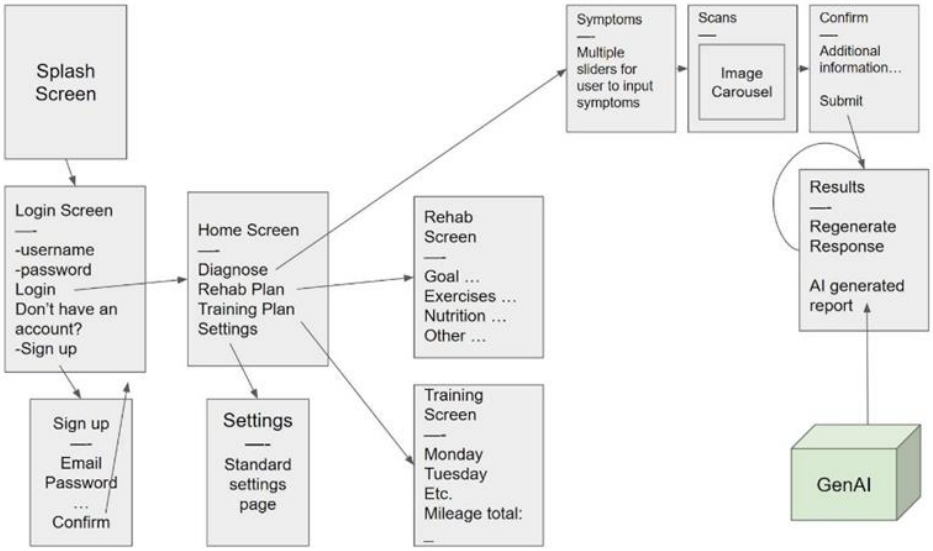

Figure 1. Overview of the solution

The authentication section serves to either grant the user a new account or allow them to log in to an existing one. This allows users to access settings and data saved to their account. Firebase was utilized to implement this system due to its data storage capabilities and account management tools. Authentication is the process of validating if a user is allowed access to an account, keeping accounts secure and protecting user privacy. This section of the program facilitates the use of accounts throughout the program, increasing convenience for users of the application.

Figure 2. Screenshot of the log in page

```
Future<User?> logInWithEmail(String email, String password) async {
  try {
    RegExp exp = RegExp(".+@.+\\..+");
    if (!exp.hasMatch(email)){
      showToast("Invalid Email", isError: true);
      return null;
    }
    if (passwordController.text.length < 6){
      showToast("Password must be at least 6 characters long", isError:  true);
      return null;
    }
    UserCredential userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: email, password: password,
    );
    print("Log in success");

    return userCredential.user; // Returns user if successful
  } catch (e) {
    print(e);
    if(e.toString() == "[firebase_auth/invalid-credential] The supplied auth credential is incorrect, malformed or has expired."){
      showToast("Invalid Credentials", isError: true);
    }
    else{
      showToast("Unknown Error", isError: true);
    }
    return null;
  }
}
```

Figure 3. Screenshot of code 1

The above method is a sample method from the codebase. This method facilitates the process of logging a user into their desired account. The “email” and “password” parameters store the credentials inputted by the user. First, the method checks to see whether the user inputted a valid email and password. It does this by checking if the email fits the standard format of an email, and checks to see if the password is long enough to meet the minimum requirement of 6 characters. If they didn’t, then the method will display an error toast and terminate itself. Otherwise, the method will continue. Next, a call is made to Firebase with the signInWithEmailAndPassword method, passing in the email and password. Firebase will authenticate the user, returning the user’s credentials if they were valid. Otherwise, null is returned. The above method is called when the user presses the login button on the login screen.

The symptoms inputting system allows the user to submit their symptoms across multiple tabs for a diagnosis. In one tab, they are given a list of body parts and muscles where they can input their reported levels of pain on a scale of 1-10 in each section. The user isn’t required to fill out all sections, but it is in their best interest if they are able, since this gives the model more information to work with. In relation to the rest of the program, this section gathers the user’s symptoms to be submitted to a NLP model which generates diagnoses and recommendations.

Figure 4. Screenshot of Diagnosis

```
onPressed: () async{
  await db.collection('Diagnosis').doc(user?.uid).set({
    "Symptoms": {
      "lHamstringPain": lHamstringPain,
      "rHamstringPain": rHamstringPain,
      "lQuadPain": lQuadPain,
      "rQuadPain": rQuadPain,
      "lKneePain": lKneePain,
      "rKneePain": rKneePain,
      "lCalfPain": lCalfPain,
      "rCalfPain": rCalfPain,
      "lAnklePain": lAnklePain,
      "rAnklePain": rAnklePain,
    },
  });
  await db.collection("AdditionalInfo").doc(user?.uid).set({
    "Text":additionalInfoController.text,
  });
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => DiagnoseResultPage(
        title: "Results",
        userContent: '''
        lHamstringPain = ${lHamstringPain}
        rHamstringPain = ${rHamstringPain}
        lQuadPain = ${lQuadPain}
        rQuadPain = ${rQuadPain}
        lKneePain = ${lKneePain}
        rKneePain = ${rKneePain}
        lCalfPain = ${lCalfPain}
        rCalfPain = ${rCalfPain}
        lAnklePain = ${lAnklePain}
        rAnklePain = ${rAnklePain}
        Images: ${stringifyFileURLS}
        Additional Information:
        ${additionalInfoController.text}
        Mileage = ${mileage}
        User goals: ${goals}
        ''',
      )  // DiagnoseResultPage
    )  // MaterialPageRoute
  );
}
```

Figure 5. Screenshot of code 2

The following method is run when the “Submit” button is pressed on the Diagnosis screen. The method begins by uploading the pain symptoms provided by the user to Firebase. The user inputs their level of pain from 1-10 on a list of sliders, and these values are stored as integers in variables such as “lHamstringPain”. The user also has an option to input additional comments into a textbox, and these contents are also uploaded to Firebase. These values are stored in Firebase so that when the user reopens the page, they can see their past inputs if they decided to save their changes. Afterwards, the “Navigator.push()” method switches to the next screen where the user is given the AI’s diagnosis. All previous information uploaded to Firebase is transferred to the next page by being passed into userContent. This is done so that the next page doesn’t need to make extra requests to Firebase to retrieve this data. Additionally, the result of “stringifyFileURLS” is passed to the next screen. “stringifyFileURLS” returns a string containing a list of URLs which link to image files uploaded by the user. ChatGPT is able to access these images and interpret them.

The AI recommendations section serves to give the user diagnoses for their injury in addition to treatment recommendations. This section uses the OpenAI API to send requests to OpenAI’s ChatGPT NLP model [8]. An NLP is capable of understanding and interpreting human language, allowing for communication with human-like chatbots. The AI recommendations section provides the user with the diagnoses and recommendations they are seeking for their injury.

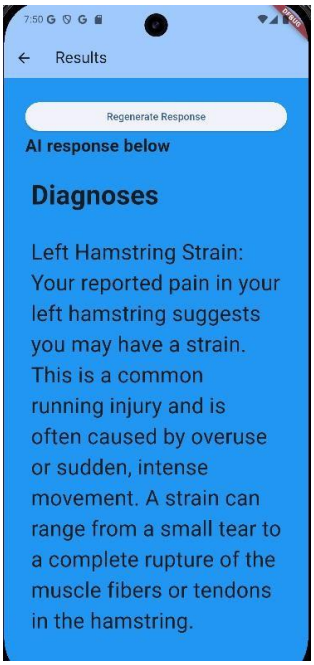

Figure 6. Screenshot of AI response

```
Future<void> ai_analyze() async {
  final url = Uri.parse('https://api.openai.com/v1/chat/completions');
  final headers = {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer ${OpenAIKey}',
  };
  final data = {//test
    'model': 'gpt-4',
    'messages': [
      {
        'role': 'system',
        'content': await _devMsg,
      },
      {
        'role': 'user',
        'content': widget.userContent,
      }
    ],
    'temperature': temperature,
  };
  try {
    final response = await http.post(url, headers: headers, body: jsonEncode(data));
    if (response.statusCode == 200) {
      final jsonData = jsonDecode(response.body);
      if (jsonData == null)
      {
        print("jsonData null");
      }
      else{
        final responseText = jsonData["choices"][0]["message"]["content"].split("[NEW SECTION]");
        setState(() {
          displayText = responseText[0];
        });
        await FirebaseFirestore.instance.collection("Responses").doc(user?.uid).set(
        {
          "Diagnosis": displayText,
          "Rehab Plan": responseText[1],
          "Training Plan": responseText[2],
          "Date": DateTime.now().toString().split(" ")[0],
        });
      }
    } else {
      print('Failed to generate. Status code: ${response.statusCode}');
    }
  } catch (e) {
    print('127 Error: $e');
  }
}
```

Figure 7. Screenshot of code 3

The above ai_analyze method is run when the results screen is first opened or when the user requests to regenerate the response via a button press. “OpenAIKey” stores the API key necessary to make requests. The variable “data” contains the prompt given to the NLP which instructs it on how to respond. It also contains the symptoms inputted by the user in string format. “data” is converted to a json file with the jsonEncode method and sent to the OpenAI API via HTTP [9]. The API runs the NLP and returns its response. The model’s response is split up into 3 strings [10]. The first string is the model’s diagnosis, the next a plan for the rehab page, and finally a training plan for the training page. These 3 are stored on Firebase along with their date of generation for later use. The diagnosis is stored in displayText and displayed on screen to the user.

4. EXPERIMENT

4.1. Experiment 1

One feature of our application is the ability to upload images and have the model diagnose the user. These images are meant to be MRI or X-ray scans, but any image files are accepted. Although this feature isn’t meant to replace traditional analysis by a radiologist or MRI tech, it is meant to provide a quick and semi-reliable way of detecting abnormalities/providing a second opinion. Thus, it is important that the model has at least some level of accuracy.

We will test the NLP and parts of our prompt by giving the model multiple X-Rays and a modified prompt to ask whether the user is injured or not. The model is instructed to return a simple yes or no answer for convenience and to limit tokens used. Thus, the prompt given to the model differs from the prompt used in the real application. Other factors were kept entirely the same to replicate the original API request as closely as possible. We will use a python script to

automate this experiment. We sourced a dataset full of knee X-rays from Kaggle **[11]**. Some X-rays are healthy, others are abnormal to varying degrees of severity (0-4 with 0 being normal and 4 being severe). The script will record the number of normal X-rays diagnosed correctly, and the number of normal X-rays diagnosed incorrectly. It will do the same for the other degrees of severity.

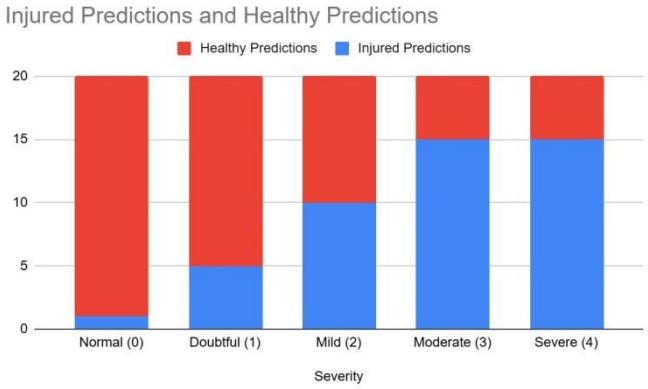

Figure 8. Figure of experiment 1

Overall, the results of this experiment met our expectations. The greater the severity of the injury in the knee X-Ray, the more frequently the model concluded that the user was injured. Among the X-Rays that were classified as Severe (4), the model accurately diagnosed the user 75% of the time. The same goes for those classified as Moderate (3). Among the X-Rays that were classified as Mild (2), the model accurately diagnosed the user in 50% of cases. Among the XRays that were classified as Doubtful (1), the model accurately diagnosed the user 25% of the time. Finally, among the X-Rays that were tagged as Normal (0), the model accurately concluded that the user was healthy 95% of the time. Thus, the model very rarely produced false positives, but had trouble producing true positives. It had decent accuracy (75%) if the the injuries were more severe, but lacked accuracy even if the injuries were just mild (50%). Overall, the model produced a correct diagnosis 64% of the time.

4.2. Experiment 2

While we previously tested the AI's accuracy, we will also test its speed. The application's diagnoses and recommendations are meant to be quick and easy to receive, so it's important to measure its exact speed and ensure it can respond in a reasonable amount of time.

The experiment will consist of 2 sets of 5 trials. In each trial, a request will be made to the model through the OpenAI API by a python script. The script simulates requests made by the actual application. A script is used instead of the real app for the sake of automation and convenience. The exact same developer prompt (which instructs the model on how to respond) is used, and the user prompt (which stores the symptoms and other inputs reported by the user) is based on a fake scenario. Once the script receives a response, it will calculate the time taken (in seconds) and display it to the console. These values will be stored in a spreadsheet, graphed, then analyzed. Due to the fact that OpenAI receives varying amounts of traffic depending on day and time, 2sets of trials will be done, each on a different date.

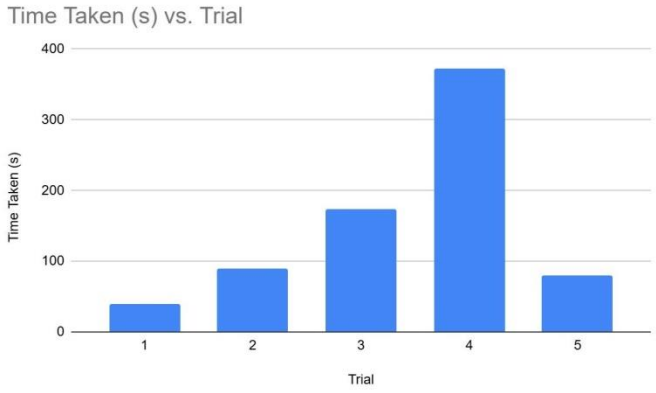

Figure 9. Figure of Day 1 (6/26/2025 at around 8 pm PST)

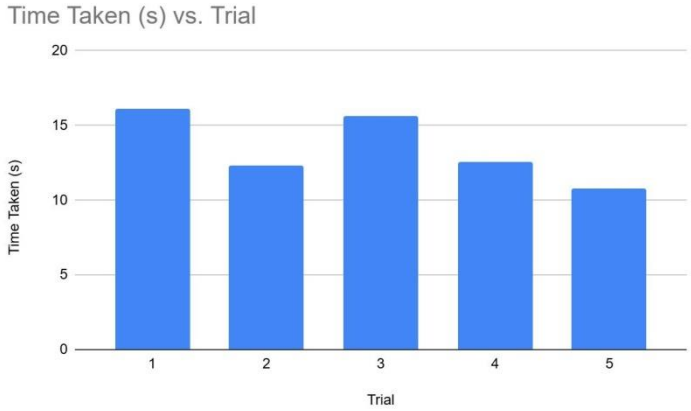

Figure 10. Figure of Day 2 (7/1/2025 at around 5:30 pm PST)

In the first set of data (Day 1), the mean response time was approximately 151 seconds or around 2 and a half minutes. The median value was around 89 seconds (or 1 minute and 19 seconds), the lowest value was approximately 39 seconds, and the highest value was about 372 seconds (over 6 minutes). These response times were somewhat disappointing, since they were multiple minutes on average. This was likely caused by a greater use of OpenAI's services at that date and time. In the next set of data (Day 2), the mean was about 13.5 seconds, the median about 12.6 seconds, the lowest value about 10.8 seconds, and the highest about 16.1 seconds. The response times were extremely quick on this day, and indicated a lack of other requests to OpenAI ' s servers. Clearly, the response time varies greatly depending on when the application is used. Overall, the mean response time across both days was about 82.1 seconds.

5. RELATED WORK

In a 2022 research study, Rahlf, A.L. et al. trained and tested an AI model for running injury prediction [12]. The model is a Deep Gaussian Covariance Network (DGCN) which is a probabilistic model and is therefore capable of predicting model uncertainty. In the research study, the model uses baseline data collected in an initial session in addition to data collected continuously throughout the course of 10 months. Baseline data used by the model include information from questionnaires, a biomechanics assessment in a laboratory setting, and a biochemical analysis from a blood test. Continuous data is collected every training session using

an IMU that was worn by test subjects [13]. Our application is naturally less accurate than the above method, since our input data is less comprehensive and our model isn't specially trained for injury detection. Injuries result from a wide variety of factors, and this alternative is able to take significantly more into account. However, our application is superior in terms of accessibility and ease of use. No blood tests or physical tracking devices are required. Our application exists as a more lightweight and accessible alternative.

A different solution leverages MoCap (Motion Capture) to prevent running injuries [14]. Motion Capture is a powerful technology which is highly accurate in tracking movement. The solution is known as KeepRunning, and is a running analysis test conducted on a treadmill to analyze the runner's form and provide pointers on how they can improve. First, the user runs on a treadmill while wearing MoCap sensors and while being recorded by a MoCap camera. Then, raw data is produced by a program and sent to a medical professional to be analyzed. This is a major difference from our AI-based solution. Although the use of a medical professional gives the results of KeepRunning greater medical authority and accuracy, it results in the solution taking longer than our alternative. While Motion Capture requires special sensors which are placed on the body, our solution doesn't require any physical equipment. Our application is at a major disadvantage however, since it is unable to analyze the user's running form, a major data point that is often the primary cause of injury.

Martínez-Gramage et al. created a machine learning model to prevent running injuries among young triathletes. A random forest model was used in this study, and it allowed researchers to see which characteristics of a triathlete's running gait contributed most to injury. 19 experienced teenage triathletes participated in the study, providing the input data for the model. All athletes underwent a 7 month gait retraining program which helped them adjust their running gaits to prevent injury. They wore IMUs and were recorded via camera to gather information about their gait including cadence and stride length. Prior to the program, some of the athletes were injured, and so the researchers created an ML model to predict whether an athlete was injured before the gait retraining based on the data collected. The model had some accuracy. "About the values resulted from the confusion matrix, TP (true positives) were 5 and FP (false positives) were 2, nevertheless the TN (true negatives) were 8 and FN (false negatives) 4" [15]. In comparison, our application doesn't require IMUs, sensors or recordings of the athlete to make predictions. The application is also more generalized, and not tailored for a small, specific dataset. Unlike the alternative above, we used a Natural Language Processing model which was able to read and analyze an additional comments section by the user. Our model was also given a questionnaire filled out by the user unlike in the above study. Overall, our solution is easier to use and more lightweight, but has more limited accuracy.

6. CONCLUSIONS

Our application has a variety of limitations. Firstly, the AI model doesn't have access to very many data points or information. It is unable to access or interpret the user's running form, blood tests, and other important factors that affect injury. Although this reduces the accuracy and comprehensiveness of the AI's feedback, we do not intend to fix this. The application is meant to be a lightweight "first-aid kit" of sorts. Additionally, the current AI model (gpt 4.1) has limited accuracy while interpreting images as seen in experiment 1. This could be fixed if we trained specialized model(s) on MRIs and X-Rays of running injuries, but we were unable to do this due to time constraints and lack of data [10]. Finally, the section where the user rates their pain from 0-10 is limited to only a few body parts, and the user is unable to specify whether they feel pain in their soft tissue or bone. This could be easily addressed in future versions of the application.
Overall, our application provides a cheap although less reliable alternative to traditional physical therapists, radiologists and other medical professionals. It is capable of providing diagnoses,

training plans and a wide range of recommendations. This application could be a valuable tool for casual or competitive athletes who are unable to access decent healthcare.

REFERENCES

[1] Running Throughthe Ages, 2d ed. (2015). Google Books. https://books.google.com/books?id=OYfeCQAAQBAJ&lpg=PP1&ots=mRsIYOb9JA&dq=ancient%20running%20outrunning%20prey%20humans&lr&pg=PA12#v=onepage&q&f=false.

[2] Vincent, H. K., & Vincent, K. R. (2013). Considerations for initiating and progressing running programs in obese individuals. PM&R, 5(6), 513-519.

[3] (n.d.). *Striking Statistics About Running and Interesting Running Facts About Running Later in Life* [Review of *Striking Statistics About Running and Interesting Running Facts About Running Later in Life*]. Runningwithgrit.com. Retrieved June 17, 2025, from https://runningwithgrit.com/statistics-about-running/#elementor-toc__heading-anchor-16.

[4] Fields, Karl B.1; Sykes, Jeannie C.2; Walker, Katherine M.3; Jackson, Jonathan C.4. Prevention of Running Injuries. Current Sports Medicine Reports 9(3):p 176-182, May 2010. | DOI: 10.1249/JSR.0b013e3181de7ec5.

[5] Mack, G. (2017, July 10). The Prime OfAn Athlete's Career Is Shorter Than You Think - FloTrack. FloTrack; FloSports. https://www.flotrack.org/articles/5067703-the-prime-of-an-athletes-career-is-shorter-than-you-think.

[6] S. Robert Lathan. (2023). A history of jogging and running—the boom of the 1970s. Baylor University Medical Center Proceedings, 36(6), 775–777. https://doi.org/10.1080/08998280.2023.2256058.

[7] Wiegand, K., Mercer, J. A., Navalta, J. W., Pharr, J., Tandy, R., & Freedman Silvernail, J. (2019). Running status and history: A self-report study. Physical Therapy in Sport, 39, 8–15. https://doi.org/10.1016/j.ptsp.2019.06.003.

[8] Claudino, J. G., Capanema, D. de O., de Souza, T. V., Serrão, J. C., Machado Pereira, A. C., &Nassis, G. P. (2019). Current Approaches to the Use of Artificial Intelligence for Injury Risk Assessment and Performance Prediction in Team Sports: a Systematic Review. Sports Medicine - Open, 5(1). https://doi.org/10.1186/s40798-019-0202-3.

[9] Musat, C. L., Mereuta, C., Nechita, A., Tutunaru, D., Voipan, A. E., Voipan, D., Mereuta, E., Gurau, T. V., Gurău, G., & Nechita, L. C. (2024). Diagnostic Applications of AI in Sports: A Comprehensive Review of Injury Risk Prediction Methods. Diagnostics, 14(22), 2516. https://doi.org/10.3390/diagnostics14222516.

[10] Quatman, C. E., Quatman, C. C., & Hewett, T. E. (2009). Prediction and prevention of musculoskeletal injury: a paradigm shift in methodology. British Journal of Sports Medicine, 43(14), 1100–1107. https://doi.org/10.1136/bjsm.2009.065482.

[11] Gornale, S., &Patravali, P. (2020). Digital Knee X-ray Images. Data.mendeley.com, 1. https://doi.org/10.17632/t9ndx37v5h.1.

[12] Rodríguez, J., Marín, J., Royo, A. C., Padrón, L., Pérez-Soto, M., & Marín, J. J. (2023). KeepRunning: A MoCap-Based Rapid Test to Prevent Musculoskeletal Running Injuries. Sensors, 23(23), 9336. https://doi.org/10.3390/s23239336.

[13] Rahlf, A.L., Hoenig, T., Stürznickel, J. et al. A machine learning approach to identify risk factors for running-related injuries: study protocol for a prospective longitudinal cohort trial. BMC Sports Sci Med Rehabil 14, 75 (2022). https://doi.org/10.1186/s13102-022-00426-0.

[14] Chomik, R., & Jacinto, M. (2021). PEAK PERFORMANCE AGE IN SPORT. https://cepar.edu.au/sites/default/files/peak-performance-age-sport.pdf.

[15] Martínez-Gramage, J., Albiach, J. P., Moltó, I. N., Amer-Cuenca, J. J., Huesa Moreno, V., & Segura-Ortí, E. (2020). A Random Forest Machine Learning Framework to Reduce Running Injuries in Young Triathletes. Sensors, 20(21), 6388. https://doi.org/10.3390/s20216388.

[16] Emery, C. A., & Pasanen, K. (2019). Current trends in sport injury prevention. Best Practice & Research Clinical Rheumatology, 33(1), 3–15. https://doi.org/10.1016/j.berh.2019.02.009.