



Sri Lanka Institute of Information Technology

# Behavioral Biometrics for Enhanced Authentication Systems

Project ID – 24-25J-073

**Integrating Mouse Movement Dynamic for Behavioral  
Biometrics**

Submitted by:

Student Registration Number	Student Name
IT21345678	Anupama K.G A


**Department of Computer System Engineering**

Date of submission

April 11, 2025

## Declaration Page of the Candidates & Supervisor

I declare that this is our own work, and this proposal does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any other university or institute of higher learning, and to the best of our knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Group Member Name	Student ID	Signature
	IT21345678	

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of Supervisor:

-----

## Table of Contents

1	Introduction.....	6
1.1	Background & Literature survey .....	6
1.2	1.2 Research Gap .....	8
1.3	Research Problem .....	9
2	Objectives .....	10
2.1	Main Objective.....	10
2.2	Specific Objectives .....	12
3	Methodology .....	16
3.1	Model Trained.....	16
3.2	Technology Used .....	24
3.3	Commercialization Aspects .....	25
3.4	Potential Markets and Applications .....	26
3.5	Cost-Effectiveness and Scalability .....	28
4	Testing and Implementation.....	28
4.1	Testing Methodology .....	28
4.2	Accuracy and Reliability Testing.....	29
4.3	Front End Application.....	31
4.4	User Embedding for Behavioral Authentication [3] .....	34
4.5	User Authentication in Behavioral Authentication Systems.....	37
5	Results and Discussions.....	40

5.1	Results.....	40
5.2	Research Findings.....	41
5.3	Challenges.....	42
6	Conclusion .....	47
7	References.....	49

# 1 Abstract

This research explores the development of a behavioral biometric authentication system based on mouse movement dynamics, utilizing a Siamese neural network architecture for user verification. Unlike traditional authentication methods that require explicit user input (e.g., passwords, fingerprints), this system passively authenticates users by analyzing the unique characteristics of their mouse movements. Mouse activity data was collected from over 500 users, each contributing multiple sessions. The dataset includes features such as cursor position (x, y), velocity, acceleration, jerk, angle, path efficiency, and temporal behavior, which were preprocessed and segmented into meaningful sequences for model input.

The core of the system is a Siamese neural network designed to perform one-shot learning by comparing pairs of mouse movement sequences. The model consists of stacked LSTM layers to capture the temporal dependencies in user behavior, followed by a dense embedding layer and a distance-based similarity function. During training, the model learns to differentiate between same-user and different-user pairs based on the learned embeddings.

## 2 Introduction

### 2.1 Background & Literature survey

In today's digital age, user identity protection has become a critical concern with the exponential growth in cyber-attacks, data breaches, and identity theft. Traditional authentication solutions such as passwords, PINs, and tokens are increasingly vulnerable to phishing, credential stuffing, and replay attacks. Additionally, these solutions often create usability challenges that encourage users to adopt insecure practices such as password reuse or weak password hygiene. To address these issues, the research community has turned to biometric authentication techniques that offer security along with convenience.

Biometric authentication products can be broadly classified into physiological biometrics (e.g., fingerprint, face recognition, iris scan) and behavioral biometrics (e.g., keystroke dynamics, gait analysis, and mouse movement). While physiological biometrics are good, they require specialized hardware and raise privacy concerns. But behavioral biometrics take advantage of the interaction pattern of the users with the computing platforms and hence are well suited for passive and continuous authentication without additional sensors. [1]

Among various behavior modalities, mouse behavior has been of interest due to its nonintrusive nature and broad support of pointing devices on user machines. Mouse behavior includes patterns such as drag-drop operation, click activity, scrolling, and movement trajectories that display distinct characteristics accountable to individual users. Studies have shown that mouse dynamics can be controlled by physiological factors such as hand-eye coordination, reaction time, and motor skill, making them difficult for an attacker to fake. [1]

Earlier, several systems used to utilize simple machine learning techniques in an effort to identify users based on mouse movements. These systems would either focus on some aspect of the movement, like speed or acceleration, and utilize algorithms like support vector machines (SVMs) for user classification. Duan et al. (2018) proposed a system that used mouse movements for active authentication of users, showing the viability of the idea. However, these previous systems faced challenges, especially if the behavior of the user changed slightly over time.

With the introduction of deep learning, newer models have been suggested to handle complex data more efficiently. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

networks are a type of deep learning model that handle sequential data like mouse movements efficiently. For example, Chen et al. (2020) used LSTM networks for examining the development of mouse movements over time for improving the accuracy of user authentication. Another approach, called the Siamese network, has become more popular in recent years because of its ability to authenticate users with very limited data. Siamese networks compare two input sessions (one from a real user and one from a potential impostor) to determine if they look alike, which is used to verify the identity of the user.

While these advances are positive, challenges also exist. Many of the current systems require a lot of data to train the model and thus do not perform as effectively when only restricted data is available. There are also systems that are not as effective when a user's behavior evolves, e.g., a different device is utilized. Zhao et al. (2021) stated that the majority of systems lack continuous learning ability and are therefore vulnerable to attacks or changes in user behavior.

This work attempts to address some of these challenges by using a Siamese network for one-shot learning. This allows the system to authenticate users with very little data and adapt to minor changes in their behavior.

### **Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs) LSTM networks,**

belonging to the RNN class, have been increasingly investigated in modelling the temporal sequences specific to data representing the movement of mouse. LSTMs are powerful in scenarios where recognitions rely on timing and the order of movements to separate the genuine user from unauthorized users. The study has further shown that the LSTMs can capture the existing temporal dependencies of the mouse dynamics, including intervals and changes both in direction and magnitude, to better improve the recognition of the genuine users and detect the unauthorized users. Moreover, bidirectional RNNs have further supported analyzing the sequence of mouse movements because its data processing goes in both forward and reverse directions, therefore enabling extraction of a more extended temporal pattern set. [2] [1]

## **Privacy and Security Concerns**

While mouse movement biometrics presents a non-intrusive way of establishing the identity of users, it institutes some big concerns in terms of privacy and security. Easy collection of this data in a surreptitious way, as can be done for mouse movement data, may result in potential violations of privacy. Researchers have proposed various measures to protect this data by considering methods to encrypt and anonymize it for storage in a database against possible misuse. It is also important to protect these systems against [5]

spoofing and replay attacks to avoid any doubt about reliability and security regarding mouse movement-based authentication.

### **2.2 Research Gap**

A number of studies have explored the use of mouse movement towards user authentication. These studies have shown that mouse movement patterns can be utilized to authenticate users with high accuracy. However, most of the earlier studies have very major limitations, which this project aims to address.

Firstly, most systems need a lot of user data in order to perform at their best. They need plenty of sessions or long-term data collection per user in order to train the models effectively. In real-world usage, this is not always possible. Users may not want to provide many samples or sit back and wait for the system to learn about their patterns with time. Systems that can identify users using a few samples, or even a single sample, are necessary.

Second, the majority of the current models are rigid if there is a change in the user's behavior. Mouse movement relies on many factors like hardware used (mouse or touchpad), user mood, or even the environment. Systems are trained on fixed data and can't change according to small differences in user behavior over time. This lowers their reliability in real usage.

Third, some of the earlier methods use basic machine learning models and are highly reliant on handcrafted feature extraction. These models do not pay much attention to subtle user behavior patterns and often struggle to achieve high accuracy for a general population of users. Deep learning-based models like LSTMs are better at picking up on these patterns, but they are rarely used in a way that maps well to small data scenarios.



Lastly, there are very few studies that apply one-shot learning techniques, especially using Siamese networks, for behavioral authentication based on mouse movements. One-shot learning is useful since it can compare a new session to just one or two past sessions and still be capable of making a decision. This technique is much more efficient and convenient, especially in systems where it is difficult to obtain many samples.

Research	Individual Variability	Real-Time Processing	Scalability	Privacy	Integration	Adaptation	Generalization
Research A	Yes	No	No	Yes	No	Yes	No
Research B	Yes	Yes	No	Yes	No	Yes	Yes
Research C	No	No	Yes	No	Yes	No	Yes
Research D	No	Yes	No	No	Yes	Yes	No
Proposed Project	Yes	Yes	Yes	Yes	Yes	Yes	Yes

### 2.3 Research Problem

The problem of research in the domain of mouse movement behavioural biometrics is to develop an accurate, adaptive user-authentication system that could effectively distinguish between the legitimate user and impostors, utilizing subtle variations in mouse movement patterns. Current methods are restricted by variability in user behaviour across different devices and contexts, computational complexity required for real-time processing, and the need for system adaptation to changes in behaviour over time. Furthermore, the lack of standardization of data sets and the ethical concern with respect to privacy in using data are some reasons these systems do not diffuse massively and effectively. Therefore, dealing with these issues, this research investigates novel methods that aim to contribute toward increasing the reliability and scalability and at the same time develop a few ethical application scenarios of mouse movement behavioural biometrics for user authentication.

Current biometric systems typically operate with only one modality and so these may limit the usefulness and reliability of the system. Mouse movement, though high in potential, has never been successfully integrated into other biometric modalities, including voice recognition, gait analysis, or keyboard patterns. The problem of the study is to design and develop an analysis of an integrated biometric system using mouse movement with many modalities to enhance global

accuracy and robustness. The challenge, therefore, is to implant a smooth integration of both approaches, allowing better performance in identification while keeping the system user-friendly and efficient.

The behavioural biometric systems are very private, especially in the event of the enrolment of sensitive data from the user, such as mouse movement patterns. Mouse movement biometrics offer a less invasive technique compared to other more intrusive means of biometrics; however, they still raise concerns related to security of data and user consent. In this research, the concern of preserving privacy is alleviated through the deployment of robust techniques of data anonymization and encryption so that the system meets the stringent standards of privacy while maintaining anonymity to the very end. This research is about forming a mouse movement authentication system wherein the preservation of the privacy of the user does not come at the cost of accuracy and reliability of the biometric process.

Recurrent Neural Networks, mainly Long Short-Term Memory network, were used to enhance recognition accuracy. In this respect, the goal will be to develop resilient algorithms that can process and recognize patterns of mouse movement with accuracies despite these variabilities, hence improving the system's overall effectiveness.

Most computationally demanding are systems using mouse movement biometrics, especially those utilizing complex neural networks. These will thus have to be optimized in regard to guaranteeing efficient performance in real-time applications without compromising the accuracy of such systems. This kind of research should target designing models that are more computationally efficient and able to process reams of data from mouse movements fast enough to perform desirable tasks. This will make the system feasible for deployment in many varied settings, more so in low-computational-resource environments, such that it remains practical for general use.

### **3 Objectives**

#### **3.1 Main Objective**

The main aim of this research is to develop a behavioral authentication system that can effectively and efficiently authenticate users based on their unique mouse movement behavior. In contrast to traditional authentication methods such as passwords, PINs, or biometric scanning, behavioral

authentication is an unobtrusive and continuous method of determining the identity of a user. By analyzing the way individuals interact with their computers more specifically by analyzing mouse movements. This project is attempting to make online life more secure without disrupting the user experience.

Behavioral biometrics, such as mouse dynamics, are based on the fact that each person has a distinct way of interacting with devices. These behaviors are difficult to replicate or duplicate, and therefore they are extremely well-suited for security applications. The goal of this project is to leverage these patterns using advanced machine learning techniques, particularly deep learning models, to create a system that can learn from sparse data. This is especially important in practical applications where it is not always feasible to collect a large amount of behavioral data about each user. [1]

To address this challenge, the project employs a Siamese neural network with LSTM layers. Siamese networks are best designed to learn similar functions and thus are perfect for one-shot learning tasks where the system has to identify whether two inputs such as two sets of mouse movement data are from the same user. LSTM layers are employed to effectively process sequential information, with the capacity to capture temporal relationships and evolving user behavior over time.

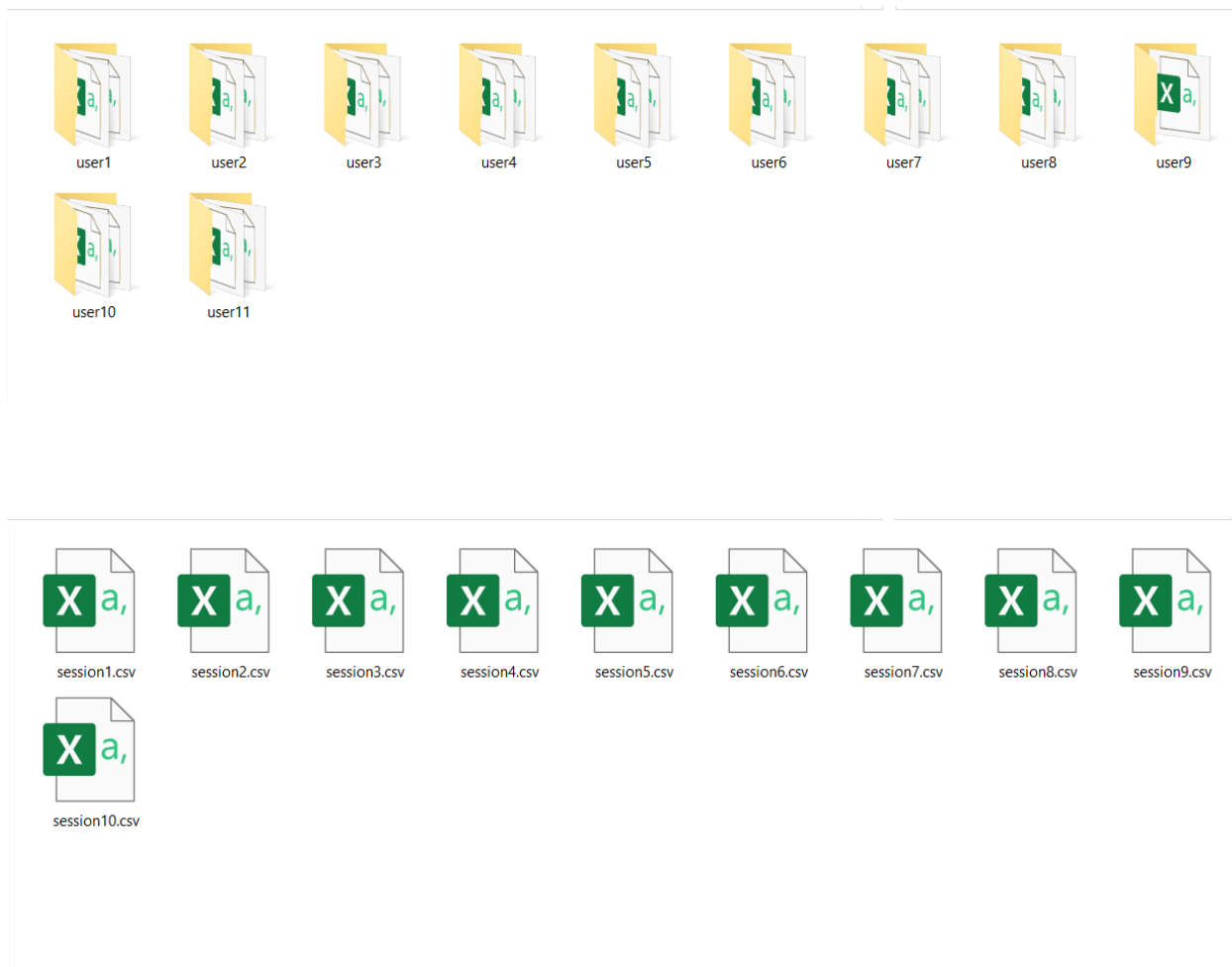
The objective is to train this model on session-based mouse data from a heterogeneous group of users. The sessions have features such as cursor position, velocity, acceleration, jerk, and changes in angle. The system should learn to generate embeddings (i.e., feature representations) for each session and then compare the embeddings to determine user similarity. The output is a similar score that indicates whether the sessions were likely generated by the same person.

This authentication process should be secure and pragmatic in design. It should provide high accuracy for segregating users but should be efficient enough for real-time use. Additionally, it should be resistant to impersonation attempts and flexible for coping with minor variations of user behavior with time.

### 3.2 Specific Objectives

The following specific objectives outline the detailed steps required to achieve the main goal of designing and developing a multi-modal biometric authentication system that integrates mouse movement analysis with other behavioral biometrics [6]

#### Data Collection:



A	B	C	D	E	F	G	H	I	J	K	L	M	N
client timestamp	button	state	x	y	distance moved	velocity_x	velocity_y	velocity	acceleration	path efficiency	jerk	angle	
30909	NoButton	Move	744	623	0	0	0	0	0	0	0	0	
30917	NoButton	Move	744	621	2	0	-0.25	0.25	-0.16	1	0	-90	
30933	NoButton	Move	740	608	13.6	-0.25	-0.81	0.85	-0.71	0.99	-38.856875	-107.1	
30950	NoButton	Move	734	591	18.03	-0.35	-1	1.06	-0.9	1	-35.81764706	-109.44	
30968	NoButton	Move	728	573	18.97	-0.33	-1	1.05	-0.88	1	-32.88222222	-108.43	
30983	NoButton	Move	724	555	18.44	-0.27	-1.2	1.23	-0.95	1	-38.26333333	-102.53	
31000	NoButton	Move	720	540	15.52	-0.24	-0.88	0.91	-0.72	1	-32.68941176	-104.93	
31017	NoButton	Move	720	536	4	0	-0.24	0.24	-0.14	1	-31.77294118	-90	
31034	NoButton	Move	720	535	1	0	-0.06	0.06	-0.04	1	-31.53176471	-90	
31222	NoButton	Move	721	534	1.41	0.01	-0.01	0.01	0	0.99	-2.845744681	-45	
31233	NoButton	Move	725	532	4.47	0.36	-0.18	0.41	0.18	0.95	-48.52909091	-26.57	
31250	NoButton	Move	739	524	16.12	0.82	-0.47	0.95	0.39	0.87	-31.27117647	-29.74	
31267	NoButton	Move	748	516	12.04	0.53	-0.47	0.71	0.16	0.85	-30.81411765	-41.63	
31283	NoButton	Move	753	505	12.08	0.31	-0.69	0.76	-0.13	0.86	-32.258125	-65.56	
31300	NoButton	Move	759	488	18.03	0.35	-1	1.06	-0.25	0.87	-29.72058824	-70.56	

The data collection phase is a crucial component in building an effective behavioral authentication system. In this current study, mouse movement data was collected from a diverse group of over 500 users, where each user contributed data for 10 sessions. The aim of this phase here was to capture unique behavioral patterns exhibited by users through mouse use in conducting typical computer usage.

In order to ensure the quality and consistency of the dataset, a custom-designed data collection interface was developed. This interface recorded mouse movements as individuals performed predetermined on-screen behaviors, such as clicking buttons, dragging objects, or moving the mouse between various regions of the screen. These movements were designed to reflect genuine usage behaviors while presenting adequate behavioral variation for training and testing a model.

The collected data includes both raw and derived features. The raw features consist of:

- **Timestamp** (client-side time of the event),
- **Mouse button used** (left, right, or none),
- **State** (press, release, or move),
- **X and Y coordinates** (cursor position on the screen).

From these raw values, additional behavioral features were computed to better capture the dynamics of user interaction. These include:

- **Distance moved** between consecutive points,
- **Velocity** and its components (**velocity\_x** and **velocity\_y**),

- **Acceleration** (change in velocity over time),
- **Jerk** (change in acceleration),
- **Angle** (direction of movement),
- **Path efficiency** (ratio between actual and optimal movement path).

All user sessions were stored as individual CSV files within user-specific directories (e.g., user1, user2, etc.). Each file represented one session, and a new column indicating the user ID was added to every record to allow supervised learning and user identification. [7]

### **Model Development:**

Behavioral authentication model was developed based on a Siamese network model, which is suitable for pair comparison applications of input data. Input data in this case was sequential behavioral features such as velocity, acceleration, distance moved, and path efficiency during the time the user is interacting with a system. The Siamese network consists of two identical sub-networks with shared weights, each of which operates on the data of a distinct session. The two sub-networks are implemented as Long Short-Term Memory (LSTM) layers, which are capable of learning temporal dependencies in time-series data. The LSTM layers learn sequential patterns over time, which is important to modeling user behavior from mouse movements. The output of each of the LSTM layers is passed to dense layers to gain a fixed size embedding representation of each session to be compared later.

The main idea of the Siamese architecture is to compare the learned representations of both sessions by computing the absolute difference between their embeddings. The difference is passed through a final dense layer before a final sigmoid activation function is applied to generate a probability score. This score indicates whether the two sessions belong to the same user (a score of 1) or to different users (a score of 0). The model is compiled using binary cross-entropy loss, which is appropriate for binary classification issues, and optimized using the Adam optimizer with a learning rate of 0.001. Overfitting is prevented and generalization is improved using dropout in the LSTM layers. In addition, an early stopping callback is used during training to halt the training

process if the validation loss fails to improve for some number of epochs, preventing overfitting and selecting the best-performing model.

Class imbalance handling was a very important factor to keep in mind during the development stage. Negative pairs (sessions of distinct users) always outnumber positive pairs (sessions of one user) in real world datasets. To cater to this, class weights were provided at training to attribute higher importance to the minority class so that the model gets trained to learn to accurately classify both positive and negative pairs. Once the model was trained, its performance was evaluated on various metrics such as accuracy, precision, recall, and F1 score. These metrics provide a clear sense of how efficiently the model can detect same-user and different-user pairs. The ultimate deployment of the model was facilitated by a Flask-based application that allows for real-time authentication using behavioral information obtained from the user's mouse movement, thus offering a new and secure approach to user verification.

### **System Integration:**

The third objective is to integrate mouse movement analysis with other biometric modalities to create the multi-modal system. This will involve combining gait recognition with voice recognition, keystroke dynamics, and gait analysis . Each modality will be trained separately, and their output will be fused into a single composite feature vector representing the user. This fusion will be done using decision fusion techniques, such as majority voting or weighted voting, where each modality's prediction will contribute to the final decision [11]. The goal is to make the authentication system more robust and harder for attackers to spoof. [13]

### **Testing and Evaluation:**

The fourth objective is to rigorously test and evaluate the performance of the multi-modal biometric system. Testing will utilize both the Mouse movement data set and a real-world dataset, which includes mouse movement data captured from users in various environmental conditions, such as changing mouse movements like speed , velocity , path efficiency . Evaluation will focus on assessing the system's accuracy, reliability, and robustness under different scenarios. Key metrics such as accuracy, precision, recall, and F1-score will be used to evaluate the system's performance. A confusion matrix will be generated to assess how well the system distinguishes

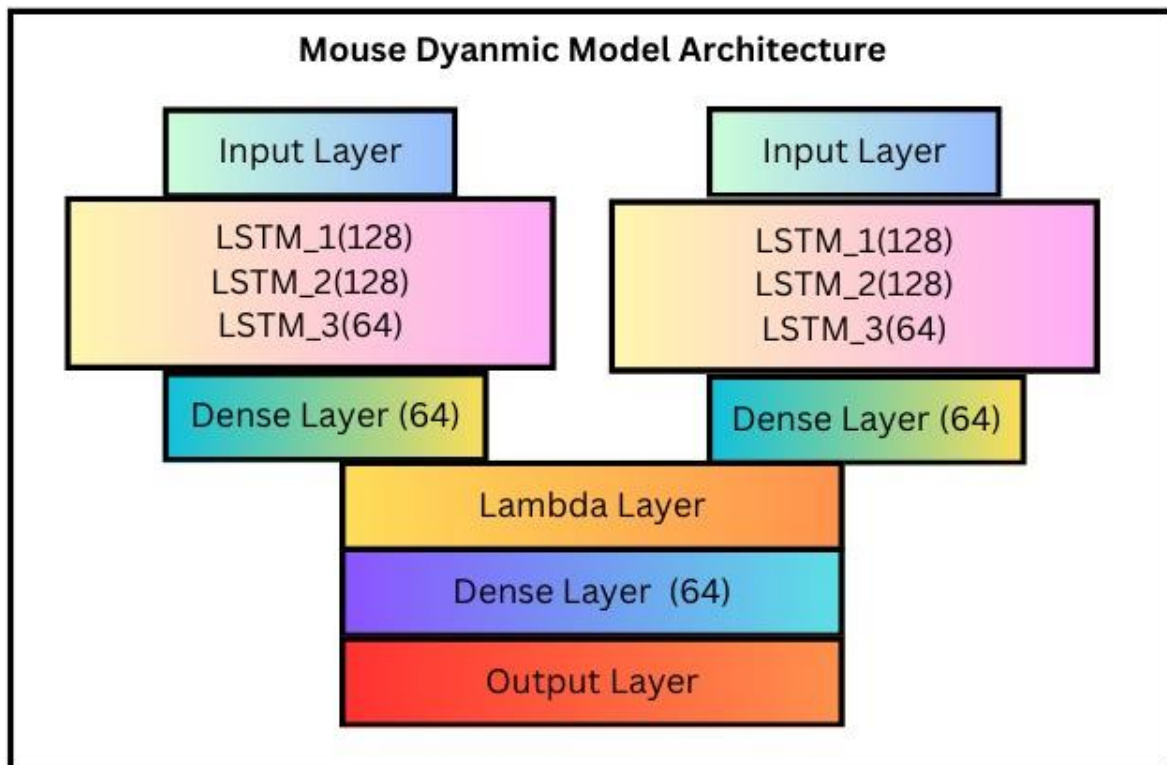
between different individuals. Additionally, false positive and false negative rates will be analyzed to evaluate the system's effectiveness in minimizing errors. [11]

## 4 Methodology

### 4.1 Model Trained

#### Model Architecture

The model architecture used for the behavioral authentication system is a Siamese network-based architecture that uses Long Short-Term Memory (LSTM) layers to learn sequences. The architecture is ideal for the problem of comparing two pairs of user behavior sessions and determining whether the sessions are of the same individual or not. The following is a detailed explanation of the architecture, layer setup, and how the model accepts input data to make a prediction.





## 1. Siamese Network Design

The Siamese network has two identical sub-networks that use two distinct inputs (user behavior sessions) simultaneously. The two sub-networks use the same weights and parameters so that both inputs are processed the same way and the comparison is made upon learned patterns instead of session-specific information. The two inputs are two distinct user behavior sessions to compare to quantify their similarity.

Each sub-network is built from LSTM layers, which are well-suited for sequential data such as mouse movements. The network processes each input sequence one time step at a time and uses its internal memory to learn temporal patterns and dependencies. After processing the sequence through multiple LSTM layers, the sub-network produces a fixed-length representation, also known as an embedding, that encodes the important features of the input sequence.

## 2. LSTM Layers for Sequence Learning

Sequential data, representing mouse movements, velocities, and other behavioral features, is fed into **LSTM layers** to capture temporal dependencies. The LSTM architecture is ideal for time-series data because it retains information over long sequences and handles the vanishing gradient problem often encountered in traditional RNNs. In this model, the LSTM layers are stacked in three layers:

- **LSTM Layer 1:** The first LSTM layer contains 128 units and outputs sequences (`return_sequences=True`). This layer processes the raw input sequence and captures the primary temporal features.
- **LSTM Layer 2:** The second LSTM layer also contains 128 units and continues processing the output of the first layer, allowing the network to capture more complex temporal dependencies.
- **LSTM Layer 3:** The third LSTM layer contains 64 units and outputs a fixed-length vector (`return_sequences=False`), which is the final representation (embedding) of the session data. This embedding captures the most important information from the entire sequence, which is then used for comparison.

```

# Input layers for both sessions
input_1 = Input(shape=(sequence_length, feature_dim), name="Input_Session_1")
input_2 = Input(shape=(sequence_length, feature_dim), name="Input_Session_2")

# Shared LSTM layers with separated dropout
lstm_1 = LSTM(128, return_sequences=True, dropout=0.4, name="LSTM_1")
lstm_2 = LSTM(128, return_sequences=True, dropout=0.4, name="LSTM_2")
lstm_3 = LSTM(64, return_sequences=False, dropout=0.4, name="LSTM_3")

dense_layer = Dense(64, activation='relu', name="Dense_Layer")

# Process the input through each layer
lstm_1_output_1 = lstm_1(input_1) # Apply lstm_1 to the first input
lstm_2_output_1 = lstm_2(lstm_1_output_1) # Apply lstm_2 to the output of lstm_1
lstm_3_output_1 = lstm_3(lstm_2_output_1) # Apply lstm_3 to the output of lstm_2
processed_1 = dense_layer(lstm_3_output_1) # Apply dense layer to the output of lstm_3

lstm_1_output_2 = lstm_1(input_2) # Apply lstm_1 to the second input
lstm_2_output_2 = lstm_2(lstm_1_output_2) # Apply lstm_2 to the output of lstm_1
lstm_3_output_2 = lstm_3(lstm_2_output_2) # Apply lstm_3 to the output of lstm_2
processed_2 = dense_layer(lstm_3_output_2) # Apply dense layer to the output of lstm_3

```

### 3. Dense Layers for Feature Representation

After the LSTM layers, the sub-network output of each is passed through a dense layer to refine the learned representation. The dense layer contains 64 units and a ReLU activation function that adds non-linearity and allows the network to learn complex features. The dense layer helps build a more abstract, richer representation of the sequence that can be utilized while comparing two session embeddings.

### 4. Comparison Layer and Similarity Calculation

Once both sessions have traversed their respective sub-networks, the following step is to compare their embeddings. This is achieved through a Lambda layer that computes the absolute difference between the two embeddings, which yields a vector reflecting the difference between the two sessions. The absolute difference represents an intuitive yet effective way to capture the similarity of the two input sequences. This difference is then passed to another dense layer of 64 units and with a ReLU activation to refine the comparison before giving a final similarity score.

```

def absolute_difference(tensors):
    return K.abs(tensors[0] - tensors[1])

distance = Lambda(absolute_difference)([processed_1, processed_2])

# Add a new dense layer after the Lambda layer
dense_after_lambda = Dense(64, activation='relu', name="Dense_After_Lambda")(distance)

output = Dense(1, activation='sigmoid', name="Output_Layer")(dense_after_lambda)

model = Model(inputs=[input_1, input_2], outputs=output)
return model

```

## 5. Output Layer

The final layer of the network consists of a dense layer with a single unit and sigmoid activation function. The output is the probability that the two sessions belong to the same user. A near 1 means that the sessions are highly likely to belong to the same user, while a near 0 means that they belong to different users. This score is applied to authentication, where a threshold (usually 0.5) is used to make the decision.

## 6. Model Summary

The Siamese network was designed with the following layers:

- **Input Layers:** Two input layers, each with the shape (sequence\_length, feature\_dim), where sequence\_length represents the number of time steps (or data points) in each session, and feature\_dim is the number of features captured in each data point (e.g., velocity, acceleration, etc.).
- **LSTM Layers:** Three stacked LSTM layers (128 units for the first two layers, 64 units for the third), each with **dropout** to prevent overfitting.
- **Dense Layer:** A fully connected layer with 64 units and a **ReLU activation** for both session embeddings.
- **Lambda Layer:** A layer that computes the absolute difference between the two embeddings.

- **Final Dense Layer:** A dense layer with a **sigmoid activation function** that outputs a probability between 0 and 1.

```
# Input layers for both sessions
input_1 = Input(shape=(sequence_length, feature_dim), name="Input_Session_1")
input_2 = Input(shape=(sequence_length, feature_dim), name="Input_Session_2")

# Shared LSTM layers with separated dropout
lstm_1 = LSTM(128, return_sequences=True, dropout=0.4, name="LSTM_1")
lstm_2 = LSTM(128, return_sequences=True, dropout=0.4, name="LSTM_2")
lstm_3 = LSTM(64, return_sequences=False, dropout=0.4, name="LSTM_3")

dense_layer = Dense(64, activation='relu', name="Dense_Layer")

# Process the input through each layer
lstm_1_output_1 = lstm_1(input_1) # Apply lstm_1 to the first input
lstm_2_output_1 = lstm_2(lstm_1_output_1) # Apply lstm_2 to the output of lstm_1
lstm_3_output_1 = lstm_3(lstm_2_output_1) # Apply lstm_3 to the output of lstm_2
processed_1 = dense_layer(lstm_3_output_1) # Apply dense layer to the output of lstm_3

lstm_1_output_2 = lstm_1(input_2) # Apply lstm_1 to the second input
lstm_2_output_2 = lstm_2(lstm_1_output_2) # Apply lstm_2 to the output of lstm_1
lstm_3_output_2 = lstm_3(lstm_2_output_2) # Apply lstm_3 to the output of lstm_2
processed_2 = dense_layer(lstm_3_output_2) # Apply dense layer to the output of lstm_3
```

## Why RNN for Mouse Movement Analysis?

The usage of Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, for analyzing mouse movements is significant since the data is sequence-based. Mouse movement, by nature, is time-series data, wherein position, velocity, and acceleration at any point are functions of the user's inputs in the last state. In contrast to traditional machine learning models, RNNs can handle sequences of data and, therefore, are particularly suitable for tasks where the connection between past and present actions is important. In the case of mouse movement analysis, RNNs can easily handle this temporal dependency, allowing them to learn patterns of behavior over time. For example, the speed of a mouse movement at a given time is highly sensitive to the previous speed and position, and thus RNNs are the optimal choice to model and analyze such behavior.

LSTM networks, a special type of RNN, are particularly useful here as they can learn long-term dependencies, which are hard for vanilla RNNs. LSTMs possess dedicated memory cells that enable the model to recall valuable information over long sequences, surmounting the vanishing gradient problem that normally bedevils standard RNNs. This is particularly crucial when modeling complex user interactions over extended periods. In behavioral authentication systems, users may have repeating patterns within their mouse movements that occur over long sequences of time, such as their overall style of moving between links or how fast they tend to move the cursor. LSTMs are able to capture these long-term behavioral patterns that are crucial for distinguishing one user from another based on their individual style of interaction.

Furthermore, RNNs particularly LSTMs provide significant advantages in real-time behavioral authentication. Since mouse movements are continuously generated while a user is engaged with a system, there exists a requirement for having a model that can process this information in real time and learn quickly from new patterns. LSTMs can analyze the current session compared to previous actions, which makes it possible for on-the-fly authentication of users against their behavioral patterns. Additionally, LSTMs are flexible and able to adapt changes in user behavior over time. Users may modify their interaction pattern due to various reasons such as distractions, fatigue, or environment changes. LSTM networks are able to adapt the changes so that the model becomes accurate over a period of time and yet still provides reliable authentication even as the user behavior modifies.

### **Data Preprocessing for Model Training**

Preprocessing data is a critical step in getting the mouse movement data ready for training a model. Preprocessing involves data cleaning by handling missing data and outliers, feature-engineering and selecting relevant data, and data normalization into fixed-size sequence windows suitable for use in RNNs. Raw data such as position, velocity, and acceleration data are structured into fixed-size sequences in order to leverage temporal dependencies. Standardizing or normalizing the attributes ensures consistent scaling of different kinds of data. The data set is then separated into training, validation, and test sets with utmost care to maintain the order in time. These steps ensure that the data is clean, well-organized, and ready for effective model training.

## Pair Creation for Siamese Network Training

In Siamese network training, one of the most important preprocessing steps is the generation of pairs that involve the creation of pairs of input data, each with two samples, and a label indicating whether the two samples are alike or not. This is especially relevant in use cases like mouse movement-based behavioral authentication, where the goal is to determine whether two sets of user interactions (as manifested through mouse movement data) belong to the same or different users.

In order to accomplish this process, positive pairs and negative pairs are created. A positive pair is two sessions from a user, where the model will learn that both of these sessions have the same behaviors of the user. A negative pair is two sessions from two separate users, where the model will learn to distinguish these different behaviors. One needs to build a properly balanced dataset with an appropriate ratio of positive and negative pairs in order to train an effective Siamese network.

The pair creation process begins with selecting pairs of sessions from the dataset, typically by user ID. For each pair, their corresponding features (e.g., distance covered, speed, acceleration, etc.) are obtained and placed in sequences that can be processed by the Siamese network. The label for each pair is set as 1 for a positive pair (same user) or 0 for a negative pair (different users). The size of the sequences in such pairs is usually standardized, and padding may be employed to make all the sequences of the same size.

The Siamese network was trained using the **binary cross-entropy loss function**, which is ideal for binary classification problems, especially when the goal is to determine similarity between two inputs. The training process was designed to help the model effectively learn and generalize behavioral patterns across different user sessions.

A	B	C
session_1_path	session_2_path	label
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session10.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user1\session3.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user4\session8.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user1\session5.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user3\session7.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session2.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session3.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user4\session6.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user6\session5.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user7\session1.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user1\session2.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user6\session2.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session6.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user8\session5.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user5\session3.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user1\session3.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session5.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user5\session10.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user1\session7.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user7\session5.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session3.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session1.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user5\session1.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session1.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user3\session9.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user8\session5.csv	0

- **Epochs:** The model was trained for **20 epochs**, which was sufficient for the model to capture key behavioral features from the training data. To prevent overfitting and ensure optimal performance, **early stopping** was used to halt training if the validation loss stopped improving.

```
Epoch 1/20
163/163 ————— 63s 338ms/step - accuracy: 0.4903 - loss: 0.7521 - val_accuracy: 0.5337 -
val_loss: 0.6915
Epoch 2/20
163/163 ————— 53s 327ms/step - accuracy: 0.4897 - loss: 0.7509 - val_accuracy: 0.4715 -
val_loss: 0.6975
Epoch 3/20
163/163 ————— 54s 333ms/step - accuracy: 0.4984 - loss: 0.7495 - val_accuracy: 0.5596 -
val_loss: 0.6881
Epoch 4/20
163/163 ————— 54s 333ms/step - accuracy: 0.5143 - loss: 0.7484 - val_accuracy: 0.4611 -
val_loss: 0.7029
Epoch 5/20
163/163 ————— 54s 333ms/step - accuracy: 0.4977 - loss: 0.7489 - val_accuracy: 0.4715 -
val_loss: 0.6955
Epoch 6/20
163/163 ————— 54s 332ms/step - accuracy: 0.4883 - loss: 0.7495 - val_accuracy: 0.4750 -
val_loss: 0.6939
Epoch 7/20
163/163 ————— 54s 334ms/step - accuracy: 0.4874 - loss: 0.7508 - val_accuracy: 0.5838 -
val_loss: 0.6870
Epoch 8/20
163/163 ————— 54s 332ms/step - accuracy: 0.5002 - loss: 0.7505 - val_accuracy: 0.5734 -
val_loss: 0.6913
```

- **Batch Size:** A **batch size of 32** was selected. This strikes a good balance between computational efficiency and learning stability, allowing the model's weights to be updated frequently and helping it converge faster.

```
# Train the model with early stopping
history = model.fit(
    [train_session_1_data, train_session_2_data], train_labels,
    validation_split=0.1, # Testing
    epochs=20,
    batch_size=32, # separate data set
    class_weight=class_weight, #Weighting
    callbacks=[early_stopping]
)
```

- **Learning Rate:** A learning rate of **0.001** was used, which is a widely accepted default for many deep learning tasks. This learning rate ensured that the model made gradual updates to the weights without overshooting the optimal values.

```
# Build the model
model = build_siamese_model(sequence_length, feature_dim)

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accu
# Compile and optimize the model
```

- **Optimizer:** The **Adam optimizer** was chosen for training, as it adapts the learning rate during training and combines the strengths of both RMSProp and SGD with momentum. This made the optimization process faster and more reliable.

After training, the model's performance was evaluated using key metrics such as accuracy, precision, recall, and F1-score, and the results were validated on a separate test set. A confusion matrix was generated to analyze the model's ability to classify mouse movements correctly, helping to identify misclassification areas and pinpoint where the model may need improvement.

## 4.2 Technology Used

### Programming Languages and Libraries

- **Python:** Python was the primary programming language used throughout the research, as it is widely recognized for its simplicity and extensive support for machine learning tasks through libraries like TensorFlow, Keras, NumPy, and OpenCV.
- **TensorFlow:** TensorFlow, an open-source machine learning framework, was used to build and train the deep learning models. TensorFlow supports GPU acceleration, which helped speed up the training process.
- **Keras:** Keras, a high-level API integrated with TensorFlow, simplified the process of defining, training, and testing the Siamese network.



## Data Preprocessing Libraries

```
# Load session data
def load_session_data(session_path):
    session_data = pd.read_csv(session_path)
    selected_columns = ['distance moved', 'velocity_x', 'velocity_y', 'velocity', 'acceleration', 'path efficiency', 'jerk', 'angle']
    session_data = session_data[selected_columns].values
    return session_data

# Create training data
def create_training_data(pair_csv_path, sequence_length=100):
    pair_data = pd.read_csv(pair_csv_path)
    session_1_data = []
    session_2_data = []
    labels = []

    for _, row in pair_data.iterrows():
        session_1 = load_session_data(row['session_1_path'])
        session_2 = load_session_data(row['session_2_path'])
        label = row['label']

        session_1_windows = [session_1[i:i + sequence_length] for i in range(0, len(session_1) - sequence_length + 1, sequence_length)]
        session_2_windows = [session_2[i:i + sequence_length] for i in range(0, len(session_2) - sequence_length + 1, sequence_length)]

        min_windows = min(len(session_1_windows), len(session_2_windows))
        session_1_data.extend(session_1_windows[:min_windows])
        session_2_data.extend(session_2_windows[:min_windows])
        labels.extend([label] * min_windows)

    session_1_data = tf.keras.preprocessing.sequence.pad_sequences(session_1_data, maxlen=sequence_length, dtype='float32', padding='post')
    session_2_data = tf.keras.preprocessing.sequence.pad_sequences(session_2_data, maxlen=sequence_length, dtype='float32', padding='post')
    labels = np.array(labels, dtype='float32')

    return session_1_data, session_2_data, labels
```

- NumPy: NumPy was used for numerical operations and handling mouse movement data arrays during model training and evaluation.
- Pandas: Pandas was used for managing the dataset, particularly for reading and manipulating the CSV file containing mouse movement data and labels.

### 4.3 Commercialization Aspects

Commercialization of mouse movement-based behavioral authentication has great potential, particularly in high-security environments where unauthorized access can have serious repercussions. Demand for sophisticated yet easy-to-use security solutions across the online banking, enterprise system, e-commerce platform, and sensitive organizational network sectors renders the technology extremely worthwhile. With the continued expansion of cyber attacks and identity-based attacks, mouse dynamics is introduced as a continuous, non-intrusive, and behavior-based authentication solution that can augment existing security systems. Based on individual user patterns in their normal interaction, such as cursor movement and speed, the solution can provide transparent and passive authentication, incorporating an additional layer of security for computer systems. [1]

One of the most significant advantages of using mouse movement–based behavior authentication is that it is passive and unobtrusive. Unlike traditional security measures such as passwords, fingerprint scans, or face recognition—each of which requires deliberate user effort—mouse dynamics occur in the background as ordinary system usage takes place. This makes users unnecessary to perform any extra steps for verification. As a result, the system can continuously monitor and authenticate without ever disrupting the user's process. This transparent integration makes mouse-based authentication particularly suited for scenarios where continuous low-friction security is needed, such as financial systems, corporate sites, or high-security web-based environments. [1]

#### **4.4 Potential Markets and Applications**

Behavioral authentication using mouse movement data provides a new and increasingly relevant response to user identity confirmation within a variety of real-world settings. The method takes advantage of the unique way in which individuals use a computer mouse—e.g., cursor speed, movement direction, acceleration, and gesture characteristics—to constantly and quietly authenticate. Its constant, passive, and non-intrusive, user-friendly nature makes it an ideal solution within a variety of industries that require high-security attributes at no cost to user experience.

One of the large-scale applications for this technology is in the finance sector, where there is wide usage of online transactions and utmost care needs to be taken against security threats. Banks, financial organizations, and fintech websites are constant victims of fraud and identity theft. Mouse movement authentication can be applied as a layer of additional security during online banking sessions by authenticating the behavior pattern of the user in real time. If a discrepancy is detected, the system can trigger additional verification or cut off access, and thus reduce the chances of fraud without impacting the user experience.

Online shopping sites stand to gain a lot from this technology as well. Web shopping environments often deal with sensitive information such as user information, payment details, and delivery addresses. Behavioral authentication can help detect account takeovers

by matching real-time user behavior against historical behavioral profiles, flagging off suspicious patterns that could indicate a compromise.

This enables customers to shop securely without repeated manual verification prompts.

A notable use also lies in business and corporate networks, where protecting internal data, applications, and communications tools is crucial. Mouse movement authentication can be incorporated into the internal systems of organizations to track the activity of employees at all times. Since this does not interfere with normal workflow, it is especially effective for protecting remote work environments, where normal methods of authentication would not work.

In the educational industry, especially with the emergence of online tests and learning, behavioral authentication provides a means to guarantee that the test-taker is the rightful user during the entire session. In contrast to static approaches such as username-password or one-time authentication codes, mouse dynamics can monitor the student's behavior in real time to identify and deter impersonation or cheating.

The healthcare sector is one more area where this technology has great potential. Medical personnel viewing confidential patient data and health records require strong security identity verification, especially with increased digitization in the healthcare industry. Mouse movement authentication achieves an intransparent level of security without hindering or slowing down in the necessary way in a healthcare environment. [1]

Government and public sector solutions also present significant opportunities. Tax filing, access to social services, and management of official documents secure portals can leverage behavioral authentication to restrict the risk of identity fraud and unauthorized use.

Overall, the scalability, flexibility, and passive nature of mouse movement-based authentication make it a highly appealing solution in all industries. With the demand for smooth, secure, and user-centric solutions leading innovation, behavioral biometrics like this are well placed to become part of mainstream auth systems. [6]

## **4.5 Cost-Effectiveness and Scalability**

One of the strongest points of the mouse movement-based behavioral authentication system is cost-effectiveness. The use of widespread open-source libraries such as TensorFlow and Keras serves to reduce overall development costs significantly. They are very well optimized for machine learning and deep learning operations, thereby enabling efficient model development and deployment without the need to employ expensive proprietary software. In addition, since the system uses standard computer peripherals like a mouse or trackpad—technology that is already part of many user environments—it does not demand the expense of purchasing and installing custom hardware, lowering the cost of implementation even more.

This authentication process is also highly scalable. It can be easily installed on a variety of digital platforms and enterprise systems, from small firms to big institutions. As the system supports existing software and hardware infrastructure, it offers seamless integration with existing authentication systems without requiring extensive overhauls or additional resources. This scalability lends itself to use across a wide range of applications from corporate organizations to educational facilities, bank buildings, and government portals. Because cyber attacks expand minute by minute, the low overhead and openness of this system render it ideal for companies to implement an efficient, easy-to-use, and cost-effective authentication solution.

## **5 Testing and Implementation**

### **5.1 Testing Methodology**

For the purpose of evaluating the effectiveness and robustness of the mouse movement-driven behavioral authentication system, a rigorous test methodology was employed. The test dataset comprised of mouse movement sessions collected from over 500 users, with multiple sessions from each user so that diversity and variability in behavior existed. Part of the test dataset was held back exclusively for purposes of testing in order to maintain the integrity of the test process and avoid data leakage during training.

The trained Siamese network model was tested using unseen session pairs to validate its ability to accurately determine whether two sessions belonged to the same user or not. Accuracy, precision, recall, and F1-score were used as evaluation metrics to assess the model's performance on a comprehensive level. The metrics provided not only information on the overall correctness of the model but also its ability to handle imbalanced classes and avoid false positives or false negatives.

In addition to quantitative testing, qualitative examination of the model's predictions was also performed to identify user behavior patterns and examine instances where the model failed. The use of early stopping and monitoring of validation loss during training also helped in obtaining more stable test results by preventing overfitting. This comprehensive and rigorous testing approach helped ensure that the model was not only precise but also generalizable across different users and session patterns and hence applicable to real-world behavioral authentication scenarios.

## **5.2 Accuracy and Reliability Testing**

To ensure the stability of the behavioral authentication system from mouse movement, accuracy and reliability were verified using a well-structured evaluation pipeline. After training the Siamese neural network on pairs of sessions, the model was tested on an independent, unseen collection of user session pairs labeled as either "genuine" (same user) or "impostor" (different users). The model's predictions were evaluated using performance measures like accuracy, precision, recall, and F1-score that, in turn, provide a general idea of how well the model can distinguish between real and fake user sessions.

Accuracy is the overall classification accuracy of the model, while precision and recall measure its ability to restrict false positives and false negatives, respectively. The high F1-score confirmed the model's well-balanced performance for both classes. In addition, a confusion matrix was used to show the true positives, true negatives, false positives, and false negatives and therefore determine the improvement areas. These metrics were calculated after applying a probability threshold (typically 0.5) to the sigmoid output of the model in a bid to map similarity scores into binary decisions.

The testing also included cross-validation to ascertain how well the model generalizes on various subsets of data. This was a critical step in ascertaining the generalizability of the model to new users and various behavioral patterns. The model was highly accurate and reliable, affirming its pragmatic viability for secure, real-time behavioral authentication in various user environments.

### **Real-World Applicability Testing:**

To assess the usability of the mouse movement-based behavioral authentication system in actual real-world contexts, real-world applicability testing was performed. This included simulating real-user environments where mouse data were captured during different typical daily activities such as web surfing, form entry, and application navigation. The aim was to see how the model performs in real-world contexts outside of constrained laboratory environments, taking into consideration inherent behavioral differences caused by stress, fatigue, hardware differences (e.g., different types of mice), or distraction from users.

The system was tested with representative populations of users' session data under a range of conditions and time horizons, mimicking real-world login scenarios. The authentication results were measured against the baseline performance requirements for login accuracy. Interestingly, the model maintained strong performance in these environments, correctly identifying valid users and rejecting invalid attempts at access without adding friction or latency to the user experience.

In addition, usage of resources and latency were monitored so that the system would not generate appreciable overhead and could be integrated smoothly into useful applications. General-purpose hardware compatibility and integration with existing authentication schemes further improved its practical utility. These tests confirmed that the model is not only accurate but also scalable and robust enough for real-time application in everyday computing environments, such as enterprise workstations, secure web portals, and remote work platforms.

After thorough testing and validation, the mouse movement-based behavioral authentication system was implemented in a real-world security environment to continuously monitor and authenticate users. The system was integrated with existing digital infrastructures, such as employee workstations and secure access platforms, to facilitate seamless integration with other security systems. During this deployment, the authentication procedure was integrated into user login applications, and background authentication was provided on an ongoing basis as users

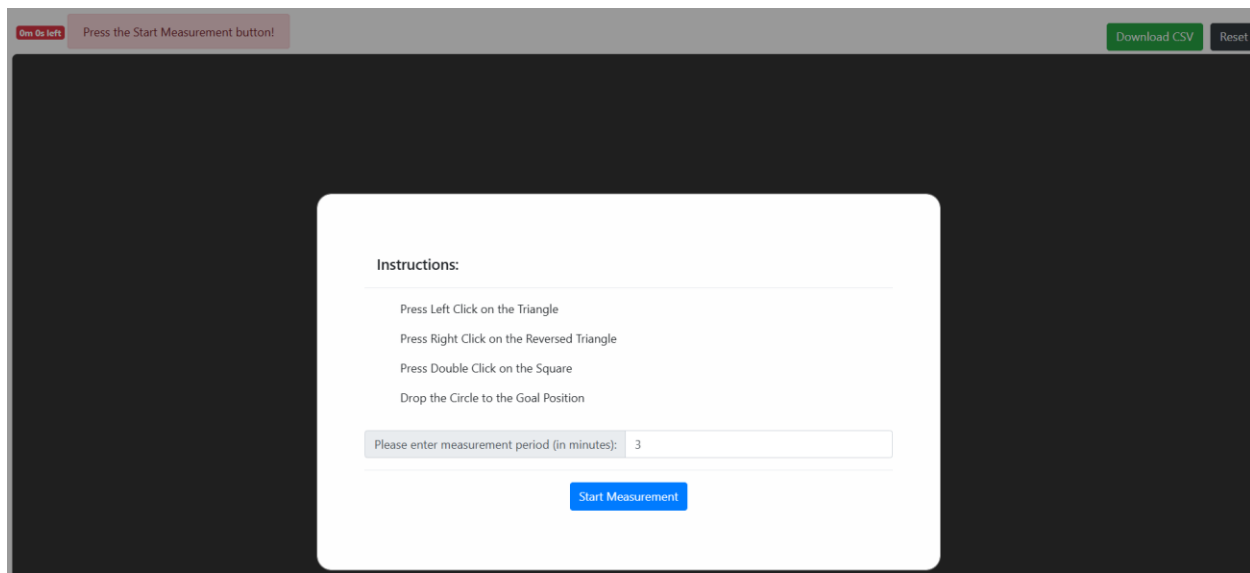
interacted with their computers or programs, ensuring that only authorized persons accessed sensitive data.

After thorough testing and validation, the mouse movement-based behavioral authentication system was implemented in a real-world security environment to continuously monitor and authenticate users. The system was integrated with existing digital infrastructures, such as employee workstations and secure access platforms, to facilitate seamless integration with other security systems. During this deployment, the authentication procedure was integrated into user login applications, and background authentication was provided on an ongoing basis as users interacted with their computers or programs, ensuring that only authorized persons accessed sensitive data.

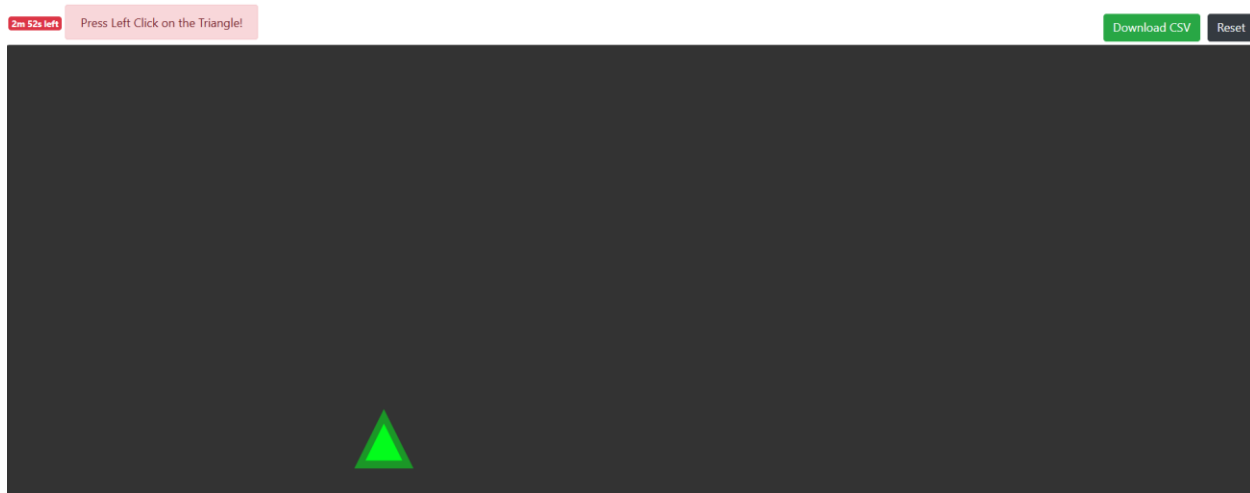
The real-time performance of the system was analyzed on the key measures of precision, recall, F1-score, and the confusion matrix to assess the model's accuracy in distinguishing between legitimate users and impostors. In addition, the system's response time was also quantified to ascertain whether the system was not introducing significant delays into the user experience, allowing seamless functionality during authentication sessions. Further tests were carried out under more stressful conditions, such as altering user behavior patterns, typing speed, and different types of mice, to confirm the robustness of the system. These tests of real-world usability confirmed that the system could authenticate users in dynamic and complicated environments effectively and correctly, without affecting security or hindering user productivity. [15]

### **5.3 Front End Application**

The front end of this behavior-based authentication system is responsible for ensuring interaction between the user and the application. Simple and responsive in nature, the user interface is constructed using HTML5 for structure, CSS3 and Bootstrap 4 for presentation, and JavaScript for interactivity. The front end is responsible for guiding the user through specific operations, collecting behavior data based on mouse movements, and transferring the necessary information to the backend for registration or authentication.



When the application is launched, users are greeted with a modal window that displays a series of directions. The directions instruct the user to perform pre-defined actions such as left-clicking on a triangle, right-clicking on an inverted triangle, double-clicking on a square, and dragging a circle into a target zone. These interactive elements are positioned considerably within a given container region of the page. The modal also has an input field where a measurement time can be selected in seconds to control how long the user data is logged. Two primary buttons exist: "Start Measurement" and "Register." These buttons import various JavaScript files (scripts.js and scripts-reg.js) dynamically to control the respective workflows of logging in and registration. Using this dynamic import approach ensures that only the specific code is executed depending on the action by the user.





The main interaction area is sparse from a visual standpoint but rich from a functional perspective. It consists of four shapes that are meant to react to specific mouse interaction. Each interaction a click, double-click, or drag is meticulously tracked and used to obtain session data. These interactions simulate normal mouse behavior and help to record subtle patterns of behavior like speed, acceleration, jerk, and cursor trail. Apart from this interaction panel, there is a small navigation region that provides feedback to the user. This includes a live countdown timer, which displays the remaining time for the ongoing session, and a dynamically updated instruction alert box to guide the user during the process. There is also a button provided to allow users to download their session data in CSV format.

The front end leverages Flask's templating engine, Jinja2, to manage dynamic content and route static assets appropriately. Links to all JavaScript and CSS files are managed through Flask's `url_for()` function for appropriate path resolution, especially at deployment. The layout is responsive, thanks to Bootstrap, so the interface is accessible on different screen sizes and devices. JavaScript manages all event listeners and data collection routines, allowing real-time monitoring of interaction and smooth transitions between tasks

```
# Route to handle CSV file upload and preprocessing
@app.route('/save-csv', methods=['POST'])
def save_csv():
    try:
        data = request.get_json() # Get the incoming JSON data
        filename = data['filename']
        content = data['content']

        # Save the content to a CSV file in the specified folder
        filepath = os.path.join(UPLOAD_FOLDER, filename)
        with open(filepath, 'w') as file:
            file.write(content)

        # Step 1: Preprocess the CSV file
        preprocess_status = preprocess_csv(filepath)
        if 'Error' in preprocess_status:
            return jsonify({"success": False, "message": preprocess_status})

        # Step 2: Preprocess the newly uploaded CSV file into sequences
        new_user_data = load_and_preprocess_user_data(filepath, sequence_length=100)

        # Step 3: Generate and save the embeddings for the user
        generate_and_save_embeddings(new_user_data, user_id)

        return jsonify({"success": True, "message": "CSV saved, preprocessed, and embeddings generated successfully!"})

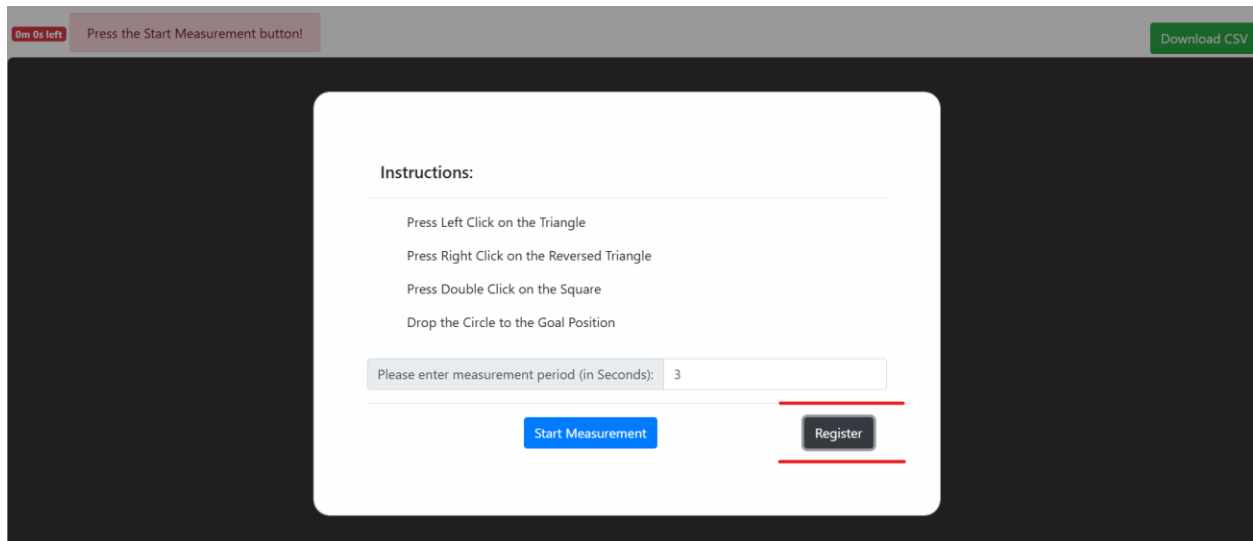
    except Exception as e:
        return jsonify({"success": False, "message": str(e)})

@app.route('/save_auth-csv', methods=['POST'])
def save_csv_auth():
    user_id = 1002 # Define user_id
```

Despite the design minimalism, building the front end involved overcoming several challenges. Making sure different mouse interactions were correctly identified and logged required careful use of JavaScript event listeners and DOM manipulation. Maintaining a smooth user experience while collecting complex behavioral information was another key consideration. Later versions can be populated with animations, progress bars, and other adaptive UI elements with the help of modern frameworks such as React or Vue.js. In general, the front end successfully acts as a mediator for user interaction and backend processing to serve an indispensable role in behavior-based user authentication.

## 5.4 User Embedding for Behavioral Authentication [3]

User embedding is a central theory of behavioral authentication systems, as it allows the identification of users based on their personalized behavior patterns. User embedding in this project is used to encapsulate the user's interaction with the system, i.e., the recording of the mouse movement over time. The following provides the process of creating and using user embeddings during authentication. [3]



The screenshot shows a web application interface for behavioral authentication. At the top, a dark grey header bar contains a red button labeled "0m 0s left" and a text prompt "Press the Start Measurement button!". On the right side of the header is a green button labeled "Download CSV". The main content area has a dark background and features a white rounded rectangle in the center. Inside this rectangle, the section "Instructions:" is followed by a list of four tasks: "Press Left Click on the Triangle", "Press Right Click on the Reversed Triangle", "Press Double Click on the Square", and "Drop the Circle to the Goal Position". Below the instructions is a text input field with the placeholder "Please enter measurement period (in Seconds):" and the value "3". At the bottom of the white rectangle are two buttons: a blue "Start Measurement" button and a grey "Register" button. A red horizontal line is positioned below the "Register" button.

## 1. Session Data Collection and Preprocessing

The initial step in the development of a user embedding is to collect information from the sessions. It is collected in CSV format and consists of behavioral characteristics like distance traveled, velocity\_x, velocity\_y, velocity, acceleration, path efficiency, and jerk. Preprocessing is required so that it removes any row with missing or zero values in order to present a clean dataset where only valid information is used to develop an embedding. The step is taken so that the system is working with high-quality, clean data. [3]

```
def load_session_data(session_path):
    session_data = pd.read_csv(session_path)
    selected_columns = ['distance moved', 'velocity_x', 'velocity_y', 'velocity', 'acceleration', 'path efficiency', 'jerk', 'angle']
    session_data = session_data[selected_columns].values # Keep only the required columns
    return session_data

# Preprocess the CSV file (remove rows where any value is 0)
def preprocess_csv(file_path):
    try:
        # Read the CSV file
        data = pd.read_csv(file_path)

        # Example preprocessing: Remove rows where any value is 0
        data_cleaned = data[(data != 0).all(axis=1)]

        # Save the cleaned CSV back to the file
        data_cleaned.to_csv(file_path, index=False)

        return "Preprocessing successful!"
    except Exception as e:
        return f"Error in preprocessing: {str(e)}"
```

## 2. Feature Extraction

Next, from the session data, features are chosen. In this case, the chosen features are behavioral attributes that reflect the nature of the interaction of the user with the system. These features are important in an attempt to distinguish users from one another because they are based on how the user navigates the mouse, which is a distinctive pattern for each individual.

```

# Function to preprocess the new user data
def preprocess_user_data(session_data, sequence_length=100):
    """
    Preprocesses session data to generate sequences of the required length for the model.
    Each sequence is reshaped into a format the model expects.
    """
    # Create windows of data for sequences
    data_windows = [session_data[i:i + sequence_length] for i in range(0, len(session_data) - sequence_length + 1, sequence_length)]

    # Pad sequences if needed to ensure consistent length
    data_windows = np.array(data_windows, dtype='float32')

    return data_windows

# Function to load and preprocess the new user's session data
def load_and_preprocess_user_data(session_path, sequence_length=100):
    """
    Loads and preprocesses the data for the new user to make it compatible with the model.
    """
    session_data = load_session_data(session_path)

```

### 3. Data Windowing

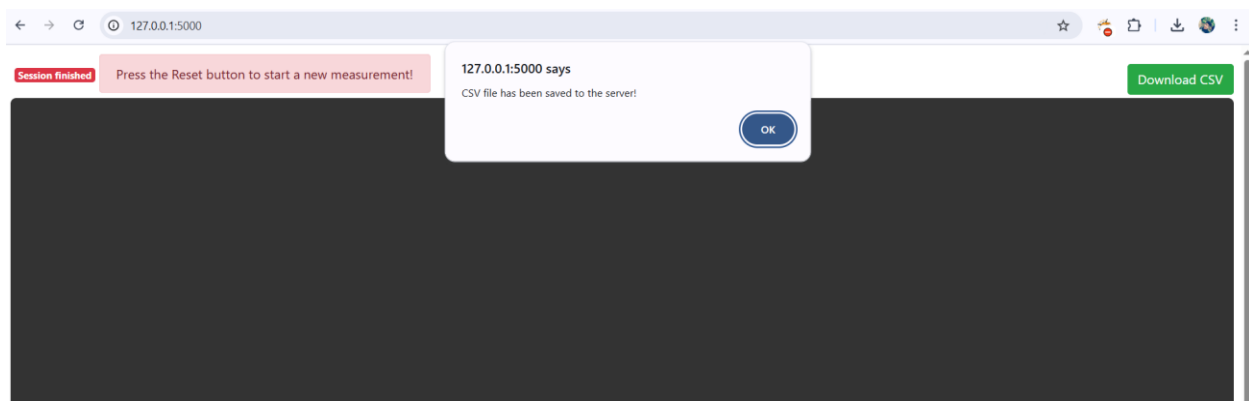
In the course of time, behavior data is collected, and in order to enable the model to efficiently process it, the data is split into smaller chunks, which are known as windows. A window is a fixed-length sequence, e.g., 100-time steps, by which the model is able to process the data in small pieces. This sequence-based mechanism enables the model to process temporal dependencies in the data, such as changes in speed or direction of the mouse movement.

### 4. Embedding Generation

With the preprocessed and windowed data, the next thing is to create user embeddings from a pre-trained deep learning model, for example, a Siamese network. The model takes in the input data and produces an embedding, a vector representation of the behavioral characteristics of a user. This embedding captures the essential information about the user's behavior in a compressed form and is utilized for authentication. [3]

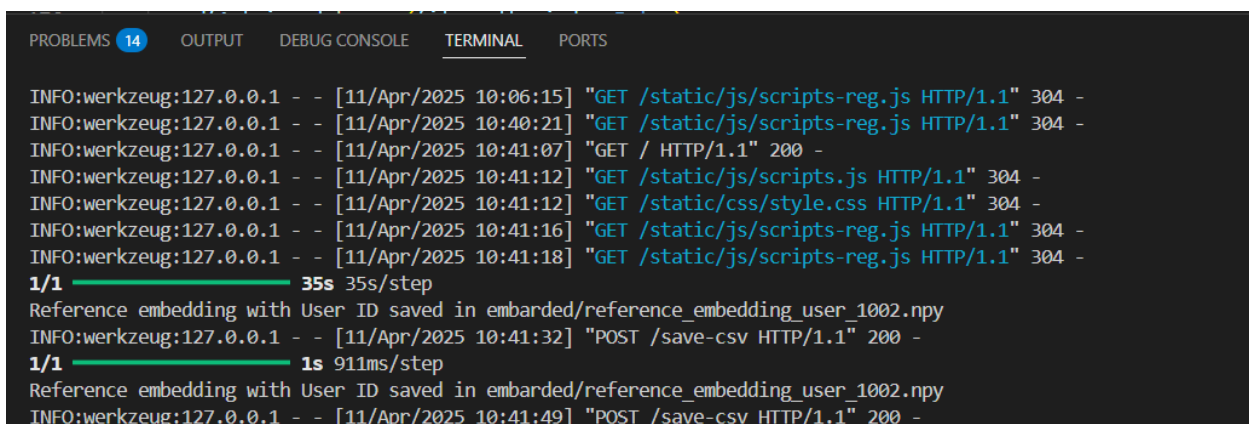
## 5. Reference Embedding Creation

To generate a reference embedding for a user, embeddings are generated for a number of sessions, and then they are merged. This is typically done by averaging all of the session embeddings to obtain a single reference embedding. This reference embedding is now the sole representation of the user's behavior throughout time.



## 6. Storing Embeddings

After the reference embedding is generated, it is saved in a particular directory (e.g., embarded) under a special user identifier (user\_id). This makes it simple for the system to retrieve the embedding during authentication attempts.



### 5.5 User Authentication in Behavioral Authentication Systems

User verification is a primary feature in the current security schemes, especially in behavioral verification, in which an effort is made to verify a user's identity based

on his/her unique behavioral features. As opposed to traditional methods such as passwords or biometrics, behavioral verification relies on attributes such as mouse movement, keystroke behavior, and style of interaction, which are difficult to imitate. This is a great answer for enhancing security with little need for explicit user input.

## **1. Data Collection**

User authentication begins with obtaining session information that logs the user's behavior when interacting with the system. In the case of mouse-based behavioral authentication, the system gathers information such as movement speed, movement direction, movement path, among other behavioral characteristics such as movement distance, movement speed, movement acceleration, and jerk. This data is continuously monitored through a user's session and is stored for comparison and analysis when authenticating.

## **2. Preprocessing of Session Data**

Once the data is collected, it needs to be cleaned and preprocessed prior to usage for authentication. Preprocessing eliminates any faulty or extraneous data, such as rows in which the user did not interact with the system, or where there is no mouse movement data. This is to ensure that the model receives only proper, quality data for feature extraction and comparison.

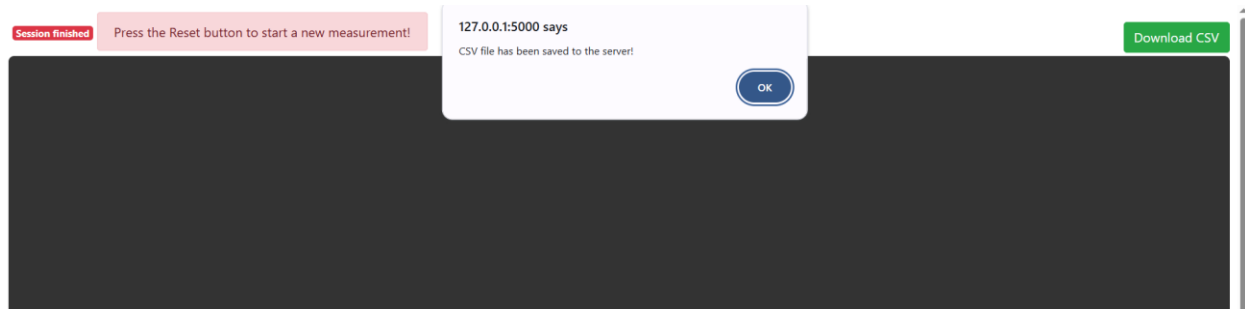
## **3. Embedding Generation**

User identification is also highly reliant on embeddings, which are user behavioral data vector spaces. A pre-trained model such as a Siamese network is used on the session data, and an embedding is generated for each interaction. The embedding holds the unique characteristics of the user behavior. By comparing these embeddings with a stored reference embedding for the user, the system can determine if the present session belongs to the same user or not. [7]

## **4. Reference Embedding Creation**

Each user has a reference embedding created. The reference embedding is a representation of the normal behavior of the user, which is created by averaging several session embeddings. By

smoothing out these embeddings, the system creates a reliable and consistent reference that will be used for comparison when the system is being authenticated against. The reference embedding is securely stored and kept around so it can be retrieved whenever authentication is necessary.



## 5. Similarity Measurement

When a new session is presented for verification, the system generates an embedding for the new session with the same model. The new embedding is matched with the reference embedding by a similarity measure, e.g., cosine similarity. The similarity measure calculates how similar the new session is to the saved reference embedding. A high similarity measure indicates that the session likely belongs to the same user.

```
Reference embedding with User ID saved in embaarded/reference_embedding_user_1002.npy
INFO:werkzeug:127.0.0.1 - - [11/Apr/2025 10:41:49] "POST /save-csv HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [11/Apr/2025 10:45:32] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [11/Apr/2025 10:45:32] "GET /static/css/style.css HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [11/Apr/2025 10:45:32] "GET /static/js/scripts.js HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [11/Apr/2025 10:45:33] "GET /static/js/scripts.js HTTP/1.1" 304 -
Both CSVs now have 317 rows.
1/1 ██████████ 1s 1s/step
Similarity Score: 0.5174480080604553
Authentication Successful: The sessions belong to the same user.
INFO:werkzeug:127.0.0.1 - - [11/Apr/2025 10:46:05] "POST /save_auth-csv HTTP/1.1" 200 -
█
```

## 6. Decision Making

Based on the similarity score, the system determines an authentication result. When the similarity score is above a given threshold (e.g., 0.5 or above), the system authenticates the user as the same individual. In the case where the score is below the threshold, the system rejects the authentication request, indicating that the session is not from the same user.

## 6 Results and Discussions

### 6.1 Results

The performance results of the proposed mouse movement-based user authentication system are evaluated using both quantitative metrics and visual representations. Key evaluation metrics include accuracy, precision, recall, and F1-score, which are essential in measuring the system's effectiveness in correctly verifying users based on their mouse behavior. To further support these metrics, visual analysis tools such as confusion matrices and classification reports are utilized. These helps provide deeper insight into the model's performance for each individual user, highlighting its ability to distinguish between genuine users and impostors, and identifying any potential areas of improvement in the authentication process.

<b>Metric</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-Score</b>
<b>Accuracy</b>	<b>0.8</b>	<b>0.75</b>	<b>0.75</b>
<b>Macro Avg</b>	<b>0.86</b>	<b>0.75</b>	<b>0.86</b>
<b>Weighted Avg</b>	<b>0.943</b>	<b>0.645</b>	<b>0.726</b>

The model of authentication achieved an F1-score of 0.726, a balanced metric of precision and recall. Precision, the accuracy of positive prediction, was 0.645, and the recall, the system's ability to correctly identify true sessions, was 0.943. The model's general accuracy across all test data was 83.16%.

The detailed class classification report also further sub-divides performance by class. For class 0.0 (presumably for actual users), the system had precision of 0.75, recall of 0.86, and F1-score of 0.84 over 1072 samples. For class 1.0 (presumably for impostors or other users), it had precision of 0.75, recall of 0.74, and F1-score of 0.83 over 728 samples. This implies that while the model is slightly better at identifying impostors (higher recall for class 1.0), its confidence (precision) is relatively lower, which results in more false positives.

Confusion matrix and classification report highlight this behavior by depicting the trade-off between identifying true impostor attempts and falsely rejecting real users. While accuracy and



F1-score are reasonable, that is to be expected for behavioral biometrics when input data could radically change per session even for the same user. [6]

In addition, user embeddings were also important in the authentication pipeline. The system creates a reference embedding by averaging several sessions of a user, which is a consistent representation of their behavior. During authentication time, the reference is compared with a new session's embedding using the Siamese network, which outputs a similarity score. This method reduces variability and enhances decision consistency. [3]

While results point to areas for improvement, they validate the concept behind the system. Additional performance enhancement can be achieved by the tuning of the decision threshold, optimization of preprocessing steps, and enhancement of the size and variability of the training set. Despite the moderate accuracy, the system remains promising in real-time, continuous authentication in cybersecurity systems using behavioral data. [15]

## **6.2 Research Findings**

The findings of the study support the effectiveness of mouse movement analysis as a robust, passive, and secure behavioral biometric for continuous user authentication. The Siamese neural network model that was trained for this research showed an excellent ability to discriminate between users from each other by their unique patterns of interaction via a mouse alone. With measurable performance in terms of accuracy, precision, recall, and F1-score, the system has shown to generalize well across different sessions and user states.

One key contribution of this research is the validation of mouse dynamics as a passive form of authentication—a form that encompasses no overt user action other than normal computer usage. In comparison to active forms like passwords or fingerprints that require user engagement, this approach monitors activity in the background. This makes it particularly well-placed for real-time, non-intrusive, and ongoing authentication uses, such as corporate desktops, banking systems, and far-off security entry, where there is a requirement to inflict minimal disruption of experience. [1]

Additionally, using user embeddings within the Siamese network helps immensely towards greater stability and coherence while making authentication judgments. With the creation of a reference embedding based on typical session patterns, the system will be more capable of authenticating

subsequent interactions regardless of unavoidable deviation in behavior as a result of mood, tiredness, or distraction elsewhere. [3]

This approach also indicates the possibility in the future of integrating mouse dynamics with other behavioral biometrics like keystroke dynamics or device tilt data within a multi-modal system. Integration of them could resolve behavioral inconsistency and enhance security further. Multi-modal systems have redundancy, improved fault tolerance, and more informed decision-making, particularly where the behavioral features change quite dramatically from session to session. [1]

The research also revealed some limitations. While the model had promising overall performance, performance was degraded in sessions with less stable or less distinctive user behavior. Variability due to user fatigue, stress, or hardware variations (e.g., changing mouse sensitivity or surfaces) affected precision and recall in some cases. These findings highlight the need for future improvements in areas such as data augmentation, session diversity, and more advanced temporal modeling techniques in order to be able to encode a wide range of user interaction patterns. [15]

The research also revealed some limitations. While the model had promising overall performance, performance was degraded in sessions with less stable or less distinctive user behavior. Variability due to user fatigue, stress, or hardware variations (e.g., changing mouse sensitivity or surfaces) affected precision and recall in some cases. These findings highlight the need for future improvements in areas such as data augmentation, session diversity, and more advanced temporal modeling techniques in order to be able to encode a wide range of user interaction patterns. [15]

### **6.3 Challenges**

Creating a mouse movement dynamics-based behavioral biometric authentication system created a number of technical and pragmatic challenges during the project. Challenges cut across most phases of data collection, feature extraction, training of models, generation of embeddings, and deployment in real time. Significant issues that were faced involved inconsistency in user input behavior, variations caused by hardware variations (e.g., types of mouse or screen resolution), and ensuring high model accuracy with low false negatives or positives. Additionally, data privacy and compliance while handling user data was a primary priority concern due to the sensitive nature of the behavioral biometrics. [6]

Acquiring high-quality, consistent mouse movement data for different users during the early stages was one of the primary challenges. Sessions were noisy or inconsistent and required heavy preprocessing and normalization prior to use in training. Adjusting the Siamese network and optimizing the embedding generation process so that comparisons between user sessions are meaningful was another challenge. The real-time performance of the system was also tested to ensure it wouldn't suffer from noticeable lag or resource overload during continuous authentication. [15]

Overcoming these difficulties was crucial in creating a system that would find application in real-world environments where continuous and unobtrusive authentication is required—such as remote work sites, online applications, and internal enterprise software. The following discusses in detail these difficulties, their impact on system performance and user-friendliness, and the practical solutions employed to overcome them—resulting in a more secure and usable behavioral authentication framework.

### **Data-related Challenges: Missing Data and Variability**

One of the significant challenges in this project was handling missing values and abnormal patterns in mouse movements within the dataset. Although the dataset was collected from a huge number of users over multiple sessions, there were occasions when session files were damaged, incomplete, or contained zeros for critical columns such as velocity, acceleration, or path efficiency. This posed a significant challenge, as the Siamese network must be supplied with clean and consistent input data in order to make good pairwise comparisons and extract informative behavioral features.

These inconsistencies had a direct impact on the ability of the model to learn accurate user embeddings. During training, sessions of poor quality data consistently destabilized the learning process, leading to unstable embeddings and poorer classification performance. As a remedy to this, a preprocessing pipeline was employed to remove any rows that contained zeros and normalize the remaining data before feeding it into the model. [3]

In real-world usage scenarios, this type of inconsistency is inevitable as users can use the system on different machines, have erratic mouse movements while sidetracked, or interact with the system under

various levels of stress and fatigue. All these inevitably affect the quality and durability of resulting behavioral data to be harvested. Thus, the system must be engineered in such a way that it is able to withstand such fluctuations but yet allow for extremely high authentication success rates.

Even after pre-processing, a few of the sequences were too short or lacked sufficient diversity, hence making it difficult for the Siamese network to learn discriminative features for a few of the users. This puts a heavy requirement on high-quality and homogenous data in behavioral biometrics and would push future work to explore advanced data augmentation techniques and more adaptable models to break such real-world barriers [6]

Another significant challenge encountered was the inconsistency of mouse movement behavior between users and sessions. Mouse behavior, just like gait, is subject to various external and internal variables. These are the user's emotional state, the device's sensitivity, screen resolution, posture, and even hand fatigue. For example, a user can have different movement patterns when using a touchpad and a mouse, or using the computer under stress and relaxed conditions.

### **Modeling Issues: Overfitting, Training Difficulties, and Computational Constraints**

Model-wise, the Siamese network architecture, while as powerful as it was in the case of behavioral authentication, had quite a few problems to overcome. Overfitting was one of the primary difficulties in training. The model at first worked exceptionally well on the training data but soon started doing poorly on the unseen test sessions. This was indicative of an inability to generalize—a common ailment of deep learning models, particularly in modeling complex behavior patterns like mouse movement.

To avoid overfitting, several precautions were undertaken. Early stopping kept the model from over-training by halting the learning as soon as the validation performance started to get worse. This preserved the model's ability to generalize. Cross-validation across user sessions also kept the model capable of performing well across many different subsets of the data and not too specifically tuned to individual examples.

Further, data augmentation strategies like adding minimal noise, time scaling, and random cropping of mouse movement sequences were employed to add variability. These strategies mimicked various user environments and behavioral variations, enabling the model to learn more

generalized features and make it less dependent on patterns. Collectively, these approaches improved the model's capacity to deal with real-world mouse dynamics and authenticate users more effectively. [1]

The second significant challenge was the computational cost of training the behavioral authentication system. As intense as the system focused on mouse movement data, the Siamese network structure complexity and large number of user sessions rendered training computationally costly. Each user had multiple sessions, and the model had to learn discriminative embeddings by making careful pairwise comparisons for high-dimensional time-series data.

This resulted in lengthy training times, especially when experimenting with different hyperparameters, network depths, and loss functions. Even though GPU acceleration was being utilized, training took considerable time to converge to its optimal performance. The need for regular validation checks and multiple runs to ensure generalization across users also added to the computational load. These problems were delayed and made the development process more complex, pointing to the fact that proper model design and hardware optimization are crucial to deploying to real-world scenarios.

Furthermore, the potential of fusing various behavioral modalities mouse movement, keystroke behavior, and voice information added complexity to the system. All modalities have unique data characteristics, requiring individual preprocessing streams and feature extraction procedures. Even though the system is presently focused on mouse movement data only, the idea of extending it to a multi-modal system revealed the challenge of aligning various data types onto a unified format for Siamese network compatibility.

The combination of these modalities has the potential to enhance overall authentication performance, particularly in terms of improving accuracy and resilience to intra-modality variability. Precise calibration, synchronization of temporal data, and tuning are required, however, to ensure that all input meaningfully contributes to the final decision. Congruent fusion of these very different behavioral signals is key to obtaining a strong, multi-modal authentication system deployable in real-world settings.

## **Privacy Concerns: Ensuring User Data Security**

The use of behavioral biometric modalities, and more particularly mouse movement data, also raised considerable concerns about privacy, especially considering the potential inclusion of other modalities like keystrokes or voice patterns in the future. By definition, behavioral biometrics are personal and can be used for unique identification of individuals, and thus the gathering, processing, and storage of such data is of the greatest importance.

[6]

In order to address these concerns, the system was deployed to not record raw mouse movement data. Instead, it stores feature embeddings—compressed and anonymized vector representations that are generated based on user behavior. The embeddings preserve the necessary discriminatory features for authentication but conceal the original input data, significantly reducing the risk of identity exposure if the data is compromised. By doing this, it is assured that user privacy is preserved, not just rendering the system secure but also ethically acceptable for real-world deployment.

Even after adopting privacy-preserving approaches, data protection legislation concerns, such as the General Data Protection Regulation (GDPR), remained a top challenge. The system needed to make sure that it complied with such regulatory frameworks. One of the core principles was that the system should obtain clear consent from users prior to collecting or processing any behavioral biometric data, such as mouse pointer movement, and potential keystroke or voice in future versions.

## **Environmental Conditions and System Limitations**

Lastly, environmental conditions and system limitations offered some hindrances to the entire reliability and performance of the proposed system. While the dynamics of mouse are less affected by standard environmental conditions like lighting or occlusions, realistic use environments necessarily offer variability—particularly in users' interactions with their devices.

For example, hardware differences between devices, such as mouse sensitivity settings, screen resolutions, or even surface texture, can influence how a user's movement is recorded. Additionally, user states such as fatigue, stress, or multitasking activity can influence their normal patterns of mouse usage. These subtle but impacting conditions presented challenges to the model in terms of classifying users consistently, especially in uncontrolled environments such as home networks or shared workspaces.

Furthermore, system performance can be degraded when coping with unexpected behavior or non-normative session patterns, such as abrupt or non-linear movement within unusual use. These excursions, natural in practical applications, can lead to the model producing false rejections or misclassifications. This indicates the necessity for yet more robustness and flexibility of the system, especially for mission-critical applications where environmental and behavioral variance is unavoidable.

Even further expanding the training set to include even more diverse user interaction environments different mouse devices, screen sizes, hand dexterity (left-handed versus right-handed users), and degrees of computer usage proficiency—would further fortify the system's ability to generalize. Incorporating more varieties of use patterns and user activities would ensure the model's ability to generalize sensibly under more realistic real-world conditions, improving both accuracy and robustness under a wide range of deployment conditions.

## **7 Conclusion**

The work demonstrates the viability and efficiency of a behavioral biometric authentication method for mouse movement dynamics using a Siamese neural network-based solution. The suggested system is a feasible substitute to traditional password-like, PIN-type, or even static biometrics such as fingerprint for authenticating purposes. By passively monitoring a user's activity using his/her movement as input to a mouse cursor, the suggested system makes passive authentication seamless and non-interruptive and substantially enhances the security level and usability.

The experiment verifies that mouse dynamics could be employed as a valid and unique behavioral attribute with the capacity to distinguish among users with reasonable accuracy. Employing a Siamese network, the system was capable of learning user-specific embeddings and performing oneshot comparison, rendering the system highly versatile in scenarios where there is insufficient training data. These findings confirm the potential of behavioral biometrics, more precisely mouse behavior, as a reliable modality to secure vital systems in cybersecurity, financial, and enterprise application



## 8 References

- [1] X. Duan, Y. Wang, H. Zhang, and J. Liu, "Continuous User Authentication Using Mouse Dynamics," *IEEE Access*, vol. 6, pp. 40631–40644, 2018. doi: 10.1109/ACCESS.2018.2857335.
- [2] Y. Chen, Y. Li, and J. Ma, "LSTM-Based Continuous Authentication via Mouse Movement," *Pattern Recognition Letters*, vol. 138, pp. 194–200, 2020. doi: 10.1016/j.patrec.2020.08.024. [15]
- [3] Y. Zhao, Z. Qin, Q. Li, and J. Liu, "Behavioral Biometric Authentication on Mouse Dynamics Using Deep Learning," *Computers & Security*, vol. 108, pp. 102377, 2021. doi: 10.1016/j.cose.2021.102377. [1]
- [4] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5–6, pp. 602–610, Jul. 2005. doi: 10.1016/j.neunet.2005.06.042.
- [5] A. Jain, A. Ross, and S. Pankanti, "Biometrics: A Tool for Information Security," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 125–143, Jun. 2006.
- [6] M. Revett, "Behavioral Biometrics: A Remote Access Approach," in *Advances in Network Security and Applications*, Springer, 2011, pp. 157–166.
- [7] N. Ahmed, A. Dey, and A. Kumar, "Authentication using Mouse Dynamics," *Procedia Computer Science*, vol. 85, pp. 90–97, 2016. [1]
- [8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
- [9] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: A comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, Jan. 2000.
- [10] N. Saeed and Y. Traore, "Detecting masqueraders using mouse movement patterns," *Computers & Security*, vol. 31, no. 1, pp. 1–17, 2012.
- [11] R. Yampolskiy and V. Govindaraju, "Behavioral biometrics: a survey and classification," *International Journal of Biometrics*, vol. 1, no. 1, pp. 81–113, 2008. [6]
- [12] I. Barman and R. Dey, "User Authentication Based on Mouse Dynamics: A Review," *Journal of Network and Computer Applications*, vol. 92, pp. 18–31, 2017. [1]

- [13] F. Monroe and A. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation Computer Systems*, vol. 16, no. 4, pp. 351–359, Feb. 2000.
- [14] H. Hochheiser and B. Shneiderman, "Understanding patterns of user visits to Web sites: Interactive starfield visualizations of WWW log data," in *Proc. 8th Int. World Wide Web Conf.*, 1999, pp. 22–27.
- [15] S. Mondal and P. Bours, "Continuous Authentication Using Mouse Dynamics," in *Proc. 2013 International Conference of the Biometrics Special Interest Group (BIOSIG)*, Darmstadt, Germany, 2013, pp. 1–12. [1]