A
Project Report titled

Three-Level Authentication System with Behavioral Analysis using Python

Submitted to
Dept. of Data Science & Artificial
Intelligence Of

Hindi Vidya Prachar Samiti's
RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE & COMMERCE (EMPOWERED AUTONOMOUS)

For Partial Fulfillment for Degree of

Bachelor of Science (Data Science & Artificial Intelligence)

2025-2026

In the Subject Head

Project (V Semester)

Submitted By

SUSHANT KHANDERAO ADHAV

CERTIFICATE

This is to certify that Sushant Khanderao Adhav (Roll No. 10221), a third year student of Bachelor of Science (Data Science and Artificial Intelligence) from University of Mumbai has successfully completed the project entitled "Three-Level Authentication System with Behavioral Analysis using Python" as a part of the academic curriculum for the Project (V Semester) which is approved for the degree of Bachelor of Science (Data Science and Artificial Intelligence) during the
academic year 2025-26.

Project Guide | Examiner | Coordinator

PLAGIARISM CERTIFICATE

This is to certify that Sushant Khanderao Adhav (Roll No. 10221), a third-year student of Bachelor of Science (Data Science and Artificial Intelligence) from University of Mumbai has submitted the project book content for plagiarism check. It was found that the content is original, as per format and the duplication is under UGC recommended rules.

In-Charge Plagiarism Committee	Project Guide

PREFACE

It gives me immense pleasure to present this report and the application I developed for my final year project. The idea for the **"Three-Level Authentication System with Behavioral Analysis using Python"** was born from the growing concern about data breaches, identity theft, and the weaknesses of traditional single-factor authentication methods.

I started by identifying the core problems: the vulnerability of password-only systems, the rising sophistication of cyber-attacks, and the lack of adaptive mechanisms in existing authentication methods. Security in modern applications needed a layered and intelligent solution.

This project is my attempt to address these challenges by implementing a secure authentication framework. The system combines **three levels of verification**: password-based login, behavioral analysis of user activity, and one-time password/device verification. Together, these levels provide a strong defense mechanism against unauthorized access.

Once the vision was clear, I began designing the architecture, selecting suitable technologies like Python, Flask, and SQLite, and developing the individual modules to create a functional and user-friendly security application.

In conclusion, I have endeavored to create a dedicated web application that demonstrates how multiple layers of authentication and behavior-based analysis can enhance system security. This black book documents that journey, from conception to implementation. I hope you find the analysis and results within this report insightful, and that the application showcases the importance of layered security in today's digital world.

ACKNOWLEDGEMENT

I would like to use this opportunity to convey my gratitude to the Management of Ramniranjan Jhunjhunwala College for providing me with this chance to accomplish this project. I am very thankful to the Principal of Ramniranjan Jhunjhunwala College for their co-operation in the successful accomplishment of my project.

A special thanks to my project guide for their most sincere efforts, support, and encouraging contribution throughout the project.

I would also like to express thanks to all my teachers, friends, and family for their support, motivation, and encouragement.

- Sushant Khanderao Adhav

INDEX

1. INTRODUCTION

1. BACKGROUND

With the rise of digital platforms, protecting sensitive information has become a major concern. Traditional password-based systems are easy to implement but vulnerable to threats like phishing, brute-force attacks, and password leaks. Multi-factor authentication improves security, yet it can still be bypassed without intelligent monitoring.

The **Three-Level Authentication System with Behavioral Analysis using Python** addresses this issue by combining three layers of verification: password login, behavior-based checks, and OTP/device authentication. This layered approach strengthens security while maintaining usability, making it a practical solution for modern applications.

2. OBJECTIVES

- To design and implement a three-level authentication system that enhances security beyond traditional password-based login.
- To integrate a behavioral analysis module that evaluates user activity patterns and detects suspicious login attempts.
- To add a third-level verification using OTP or device-based authentication for high-risk access attempts.
- To provide an admin interface for managing users and monitoring login activities.
- To ensure the system is secure, scalable, and user-friendly, making it suitable for real-world web applications.

1.3 PURPOSE, SCOPE & APPLICABILITY

Purpose:

The purpose of this project is to design a secure authentication framework that goes beyond traditional password-based systems. By integrating multiple layers of verification along with behavioral analysis, the system aims to minimize risks such as unauthorized access, identity theft, and credential-based attacks.

Scope:

The scope of this project includes the development of a three-level authentication mechanism that combines password login, behavior monitoring, and OTP/device verification. It also provides an admin dashboard for managing users and reviewing suspicious login activity. While the system is designed as a prototype, it demonstrates the feasibility of applying layered security techniques in real-world web applications.

Applicability:

This system can be applied to any web application or platform that requires enhanced security for user accounts, such as banking portals, e-learning platforms, healthcare systems, or enterprise applications. Educational institutions can also adopt it to protect student and faculty data, while organizations can use it to safeguard employee and customer information.

2.SURVEY OF TECHNOLOGIES

2.1 AVAILABLE TECHNOLOGIES

Several technologies were considered during the planning phase of this project. The key choices involved the backend framework, frontend approach, and database selection.

For the **backend**, alternatives included Node.js with Express, Django (Python), and Flask (Python). While Node.js provides excellent scalability and Django offers a full-stack framework, Flask was chosen due to its lightweight design, flexibility, and strong ecosystem of Python libraries, which suited the requirements of this project.

For the **frontend**, modern frameworks like React and Angular were considered. These provide powerful component-driven architectures but also add complexity. Since this system primarily focuses on authentication and admin functionality, plain **HTML, CSS, and JavaScript** were used for simplicity and faster prototyping.

For **data storage**, options like MySQL and PostgreSQL were considered. However, since the project required a lightweight database to manage user credentials and logs, **SQLite** was selected. It is easy to integrate with Flask, portable, and efficient for small to medium-scale applications.

2.2 TECHNOLOGIES USED

Category	Technology / Tool	Purpose
Programming Language	Python 3.x	Core language used for backend, authentication logic, and behavioral analysis.
Backend Framework	Flask (Python)	Lightweight web framework for handling routes, sessions, and APIs.
Database	SQLite	Stores user credentials, authentication logs, and behavioral data.
Frontend	HTML, CSS, JavaScript	Provides forms and interfaces for login, registration, OTP, and admin dashboard.
Security Libraries	bcrypt / passlib	Used for hashing and verifying user passwords securely.
Behavioral Analysis	Python Scripts	Evaluates user login behavior and detects anomalies based on heuristics.
Development Tools	VS Code, Git, Browser Testing	IDE for coding, Git for version control, Browser/Postman for testing flows.

3.REQUIREMENT ANALYSIS

1. PROBLEM DEFINITION

The project aims to solve the following core problems faced by students and job seekers:

1. **Weak Password Security:** Most systems rely solely on passwords, which are often weak, reused, or vulnerable to attacks such as phishing and brute force.
2. **Lack of Intelligent Monitoring:** Conventional authentication does not analyze user behavior, making it difficult to detect suspicious login attempts from compromised credentials.
3. **Single Point of Failure:** If the password is compromised, attackers can easily gain access without facing further verification.
4. **Limited Multi-Factor Adoption:** Existing multi-factor authentication methods can be complex or inconvenient for users, leading to poor adoption rates.
5. **Inadequate Admin Oversight:** Many systems lack proper tools for administrators to track login activity, monitor anomalies, and manage users effectively.

2. REQUIREMENT SPECIFICATION

Functional Requirements:

The system shall provide a **web-based user interface** for login, registration, and admin operations.

The system must allow **secure user registration** with password hashing and storage in the database.

The system shall implement **three levels of authentication**:

1. Password-based login
2. Behavioral analysis (activity monitoring)
3. OTP/device verification

The system must log all **login attempts and behavioral scores** for future analysis.

The system must provide an **admin dashboard** to manage users and review suspicious activities.

The system shall notify the user if suspicious activity is detected and request additional verification.

Non-Functional Requirements:

Performance: Login and verification processes should complete within 3 seconds.

Security: Passwords must be hashed securely (e.g., bcrypt/passlib) and sessions protected.

Usability: The interface should be simple, intuitive, and require no prior training.

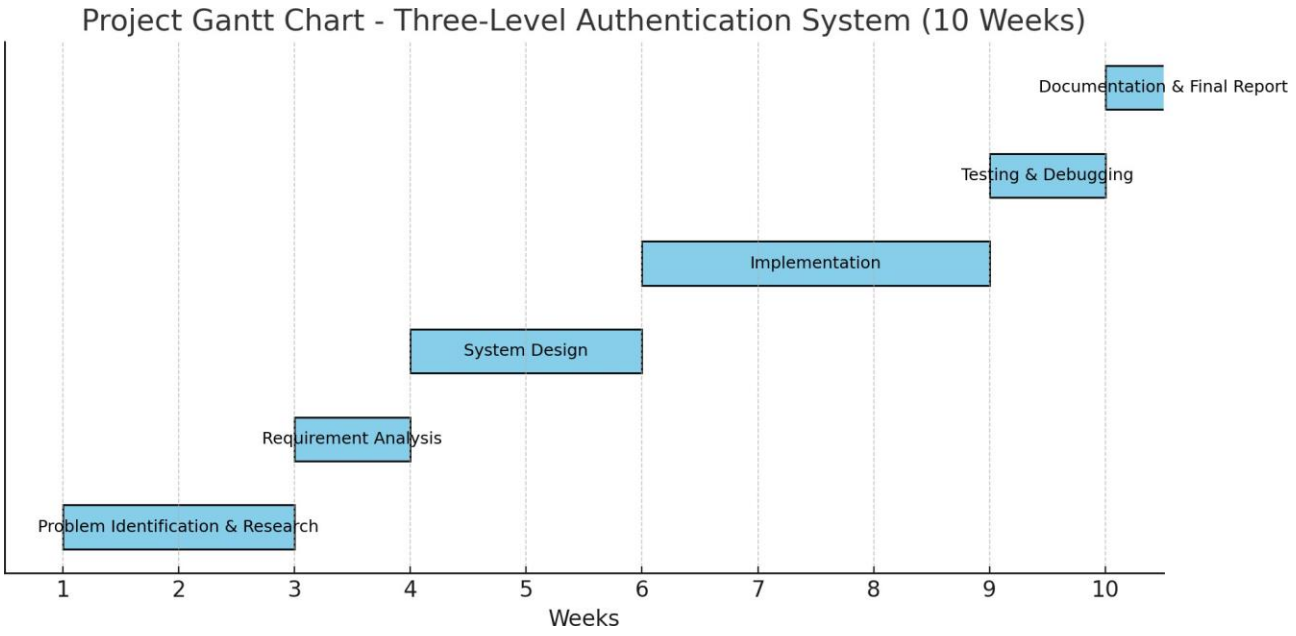
Compatibility: The web application must run correctly on modern browsers (Chrome, Firefox, Edge, Safari).

Responsiveness: The layout must adapt to desktops, tablets, and mobile devices.

- **Scalability:** The system should be modular, supporting migration to stronger databases like MySQL/PostgreSQL in the future.

3.3 PLANNING AND SCHEDULING

The project was developed following a structured plan to ensure timely completion of all phases from conception to final documentation.



3.4 SOFTWARE AND HARDWARE REQUIREMENTS

Software Requirements (Development):

- **Operating System**: Windows 10/11, macOS, or Linux distribution
- **Backend Technology**: Python 3.8+ with Flask framework
- **Database**: SQLite / MySQL (depending on deployment setup)
- **Libraries/Packages**:
    - Flask (for backend APIs)
    - PyOTP (for OTP generation)
    - smtplib / Twilio API (for Email/SMS authentication)
    - bcrypt / hashlib (for password hashing & security)
- **Code Editor / IDE**: Visual Studio Code or any equivalent IDE
- **Web Browser (for testing)**: Google Chrome, Mozilla Firefox

Hardware Requirements (User):

Processor: Intel i3 or higher / AMD equivalent

RAM: Minimum 4 GB (8 GB recommended)

Storage: Minimum 10 GB free space

Device: Any modern PC or mobile device capable of running a web browser

Browser Compatibility: Chrome, Firefox, Edge (latest versions

4.SYSTEM DESIGN

1. BASIC MODULES

The system is designed around several key modules that work together to provide the core functionality.

1. **User Interface Module**
    - Built with HTML, CSS, and JavaScript.
    - Provides login/registration forms where users enter their credentials.
    - Displays OTP verification prompts and final authentication status.
2. **Authentication Module**
    - Manages the **first level of authentication** (username & password).
    - Uses hashing (e.g., bcrypt/sha256) to securely validate stored credentials.
3. **Database Module**
    - Stores user credentials, OTP logs, and authentication attempts securely.
    - Uses SQLite/MySQL as backend storage.
      Ensures data integrity and prevents unauthorized access.
4. **Admin/Monitoring Module**
    - Allows the administrator to view user accounts and authentication logs.
    - Helps in monitoring failed login attempts for security auditing.

4.2 DATA DICTIONARY

This section describes the structure of the SQLite files used for data storage.

Table 1: users

Field	Data Type	Description	Field
id	INTEGER (PK)	Unique user ID	id
username	VARCHAR (100)	Username of the account	username
password	TEXT	Hashed password for login	password
click_x	INTEGER	X-coordinate of user's click (for graphical authentication)	click_x
click_y	INTEGER	Y-coordinate of user's click	click_y
click_x_norm	FLOAT	Normalized X-coordinate (screen-independent)	click_x_norm
click_y_norm	FLOAT	Normalized Y-coordinate (screen-independent)	click_y_norm

Table 2: behavior_logs

Field	Data Type	Description	Field
id	INTEGER (PK)	Unique log ID	id
user_id	INTEGER (FK → users.id)	Reference to the user	user_id
action	VAR CHA R(50)	Action performed (e.g., login attempt, click, color choice)	action

4.3 USE CASE DIAGRAM

The Use Case diagram illustrates the main functionalities available to the user and their interactions with the CareerPal system.

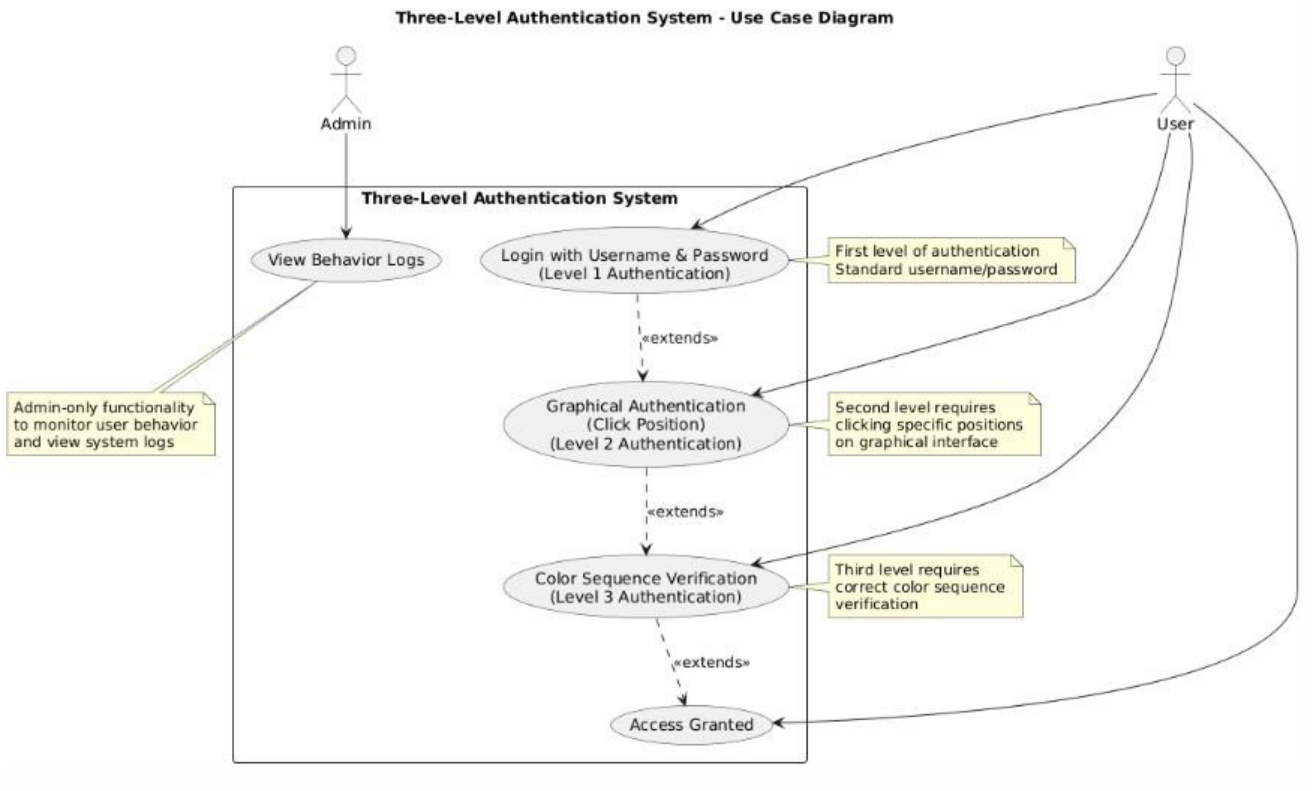


Fig 4.3: Use Case Diagram

4.4 STATE TRANSITION DIAGRAM

This diagram shows the flow of the conversation based on the internal state tracked by the backend.

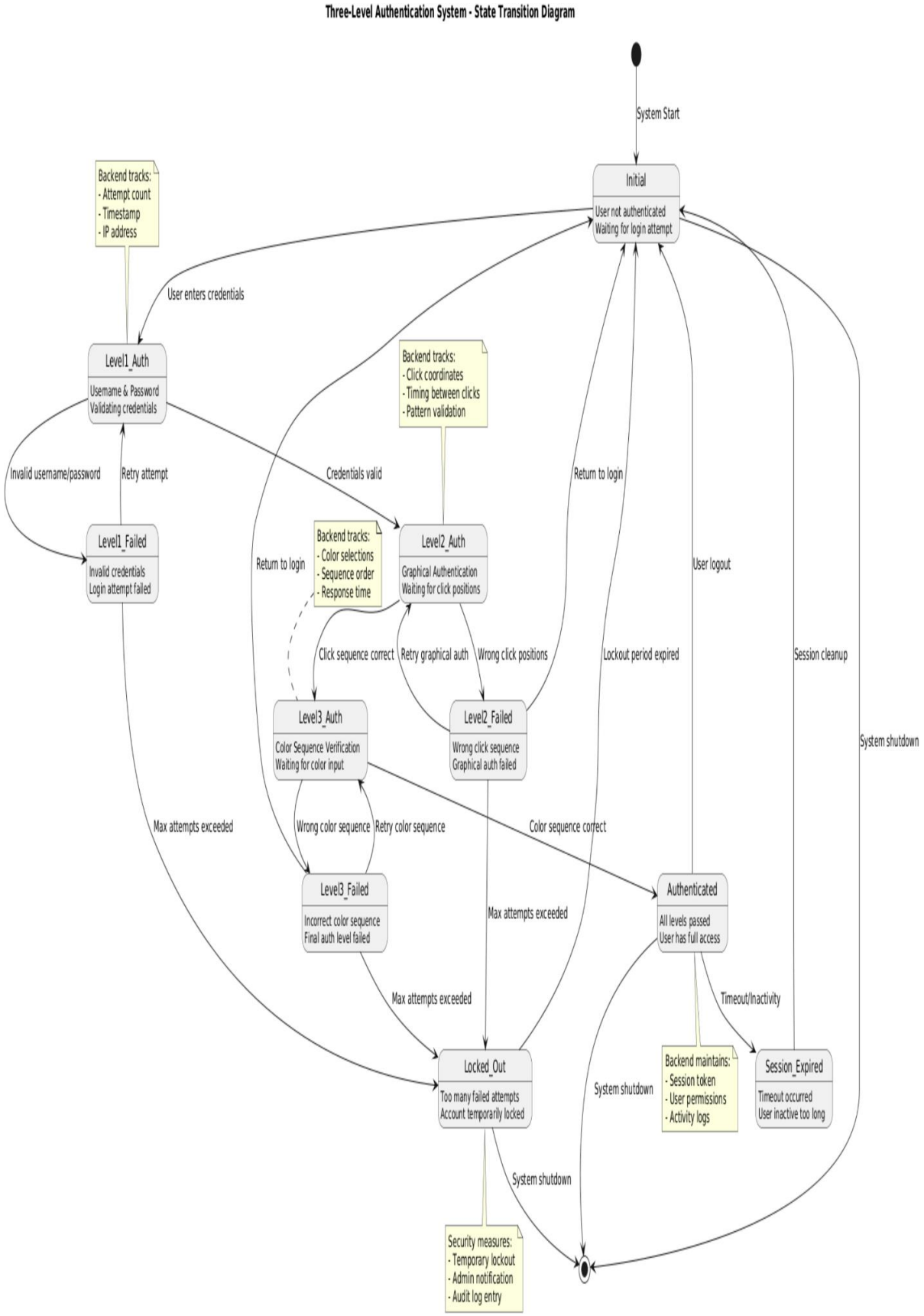


Fig 4.4: State Transition Diagram

4.5 DATA FLOW DIAGRAMS

The DFD shows how data moves through the system. This Level 0 diagram provides a high-level overview of the entire application.

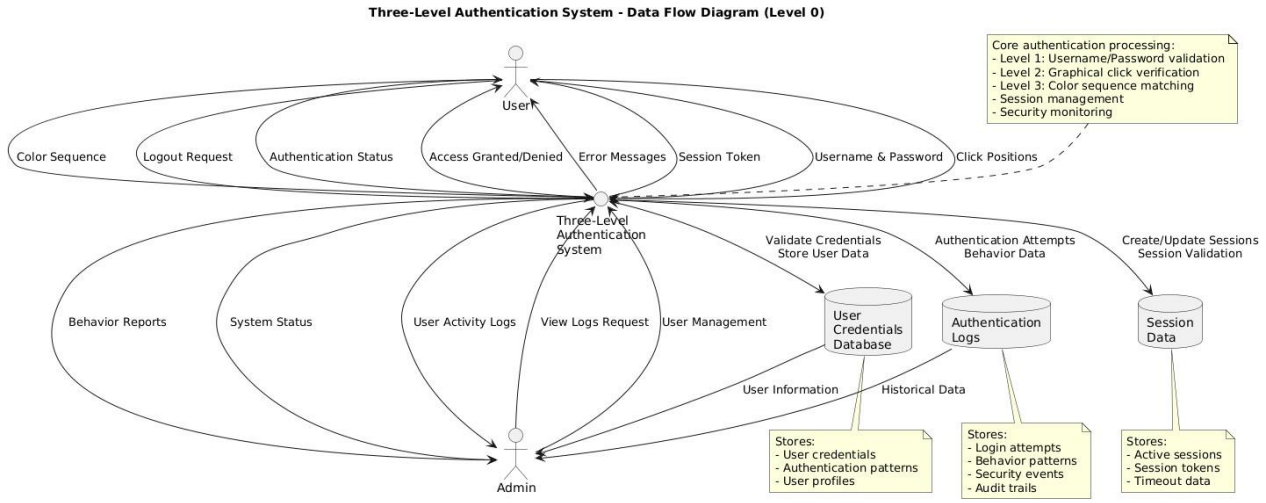


Fig 4.5: Data Flow Diagram (Level 0)

A Level 1 DFD breaks down the main process into sub-processes.

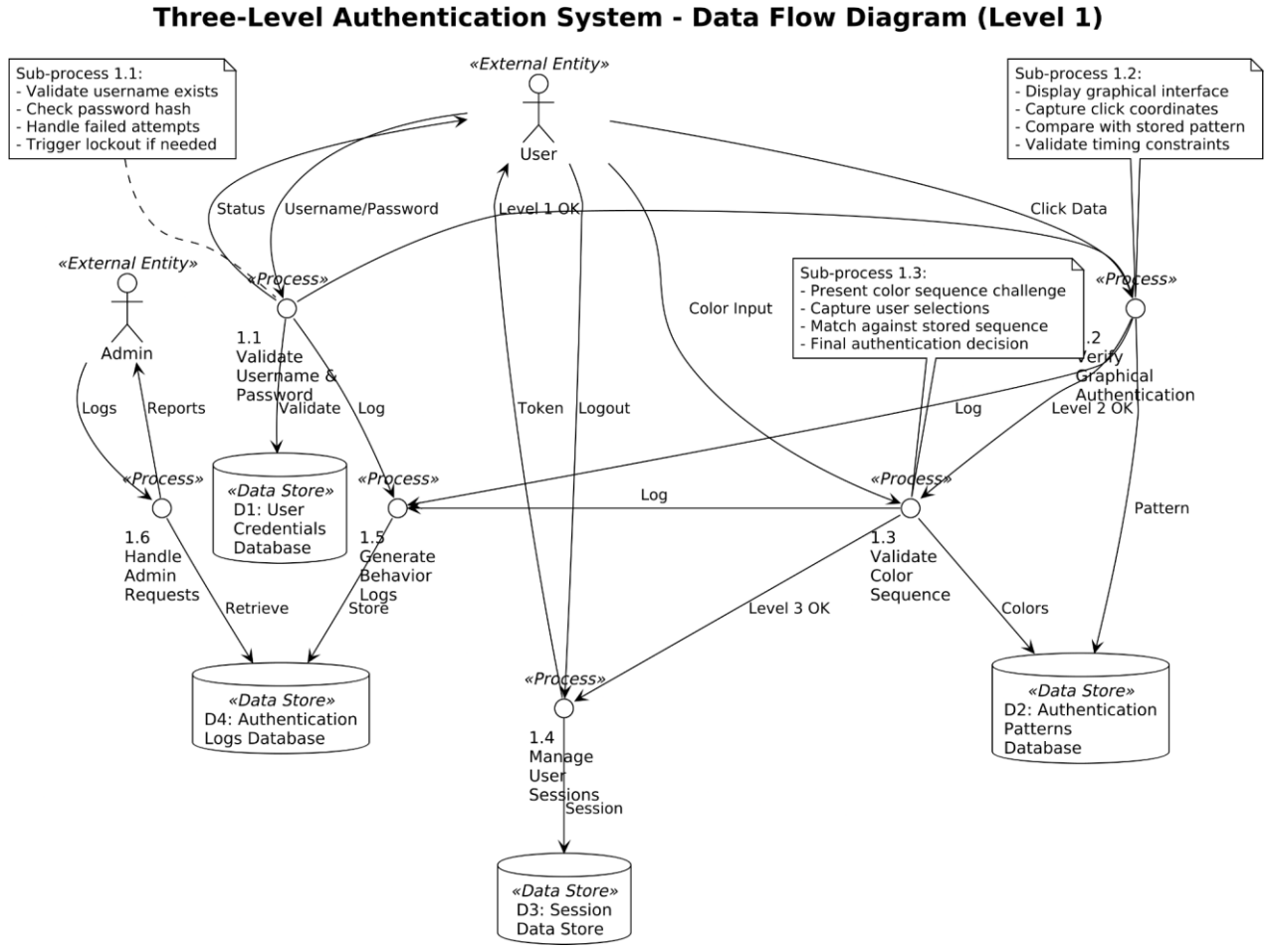


Fig 4.5: Data Flow Diagram

4.6 FLOW CHART

This flowchart on the next page shows details the logic for the Personalized Guidance feature.

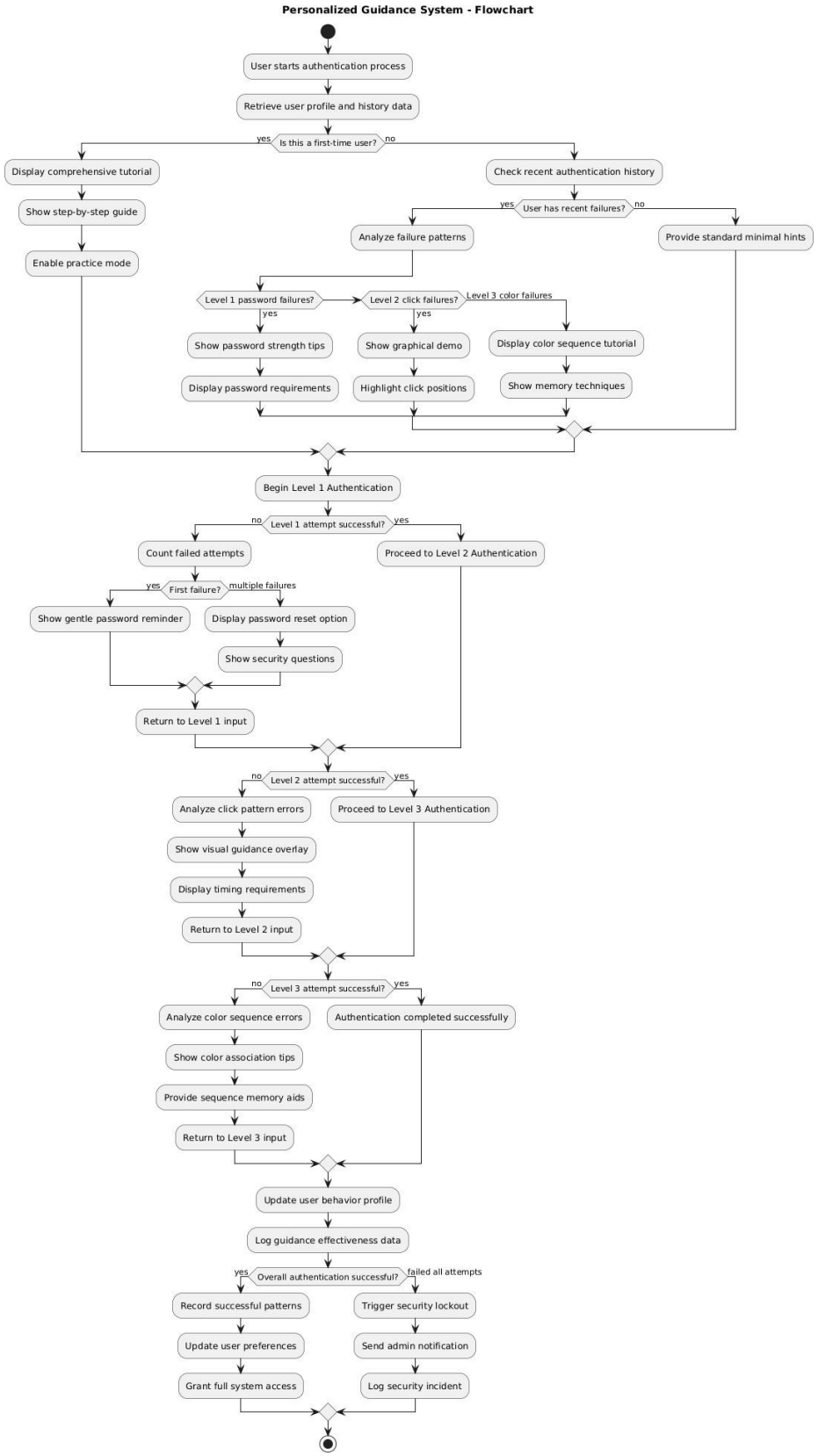


Fig 4.6: Personalized Guidance Flowchart

5.IMPLEMENTATION AND TESTING

1. IMPLEMENTATION APPROACHES

The Three-Level Authentication System is built with a Flask-based backend architecture that implements a sophisticated state management system for progressive authentication across three distinct levels.

2. Important Code Snippets

Backend Implementation:

Flask Blueprint Architecture

The application uses Flask Blueprints for modular organization, with the main authentication logic contained in `app/routes.py`. The system implements a state-driven authentication flow using Flask sessions to track user progress through each authentication level.

Key Backend Components:

- **Blueprint Structure**: Organized using `auth_bp = Blueprint('auth', __name__)` for clean route management
- **Session State Management**: Tracks authentication progress with session variables
- **Database Integration**: SQLAlchemy models for User and BehaviorLog entities
- **Security Layer**: Custom decorators for authentication and authorization

State Machine Implementation

The core authentication flow is managed through session state variables that control access to each level:

```python
# Login successful - initialize session state
session['user_id'] = user.id
session['level1_passed'] = True
session['level2_passed'] = False
session['level3_passed'] = False
```

Authentication Flow Control:

- `level1_passed`: Username/password validation complete
- `level2_passed`: Image click authentication successful
- `level3_passed`: Color sequence verification passed
- `login_time`: Timestamp for session tracking

Level 1 - Traditional Login (`/login`):

```python
@auth_bp.route('/login', methods=['GET','POST'])
def login():
    user = User.query.filter_by(username=username).first()
    if user and check_password(password, user.password):
        session['user_id'] = user.id
        session['level1_passed'] = True
        log_behavior(user.id, "login_success", {"username": username})
        return redirect(url_for('auth.image_click_auth'))
```

Level 2 - Image Click Authentication (`/level2_image`):

```python
@auth_bp.route('/level2_image', methods=['GET','POST'])
def image_click_auth():
    click_x = int(click_x); click_y = int(click_y)
    margin = 20  # Tolerance for click accuracy

    if abs(click_x - (user.click_x or 0)) <= margin and abs(click_y - (user.click_y or 0)) <= margin:
        session['level2_passed'] = True
        return redirect(url_for('auth.level3_rgb'))
```

Level 3 - Color Sequence Verification (`/level3_rgb`):

```python
@auth_bp.route('/level3_rgb', methods=['GET','POST'])
def level3_rgb():
    if submitted == user.color_sequence:
        session['login_time'] = str(datetime.now())
        session['level3_passed'] = True
        return redirect(url_for('auth.dashboard'))
```

Security Implementation

Authentication Decorators

The system implements robust access control using custom decorators:

```python
def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if 'user_id' not in session:
            flash("Login required.", "danger")
            return redirect(url_for("auth.login"))
        return f(*args, **kwargs)
    return decorated_function

def admin_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        user = User.query.get(session['user_id'])
        if not user or not user.is_admin:
            flash("You are not authorized to view this page.", "danger")
            return redirect(url_for("auth.login"))
        return f(*args, **kwargs)
    return decorated_function
```

Behavior Logging System

Comprehensive activity tracking is implemented through the `log_behavior()` function:

```python
def log_behavior(user_id, action, data=None):
    ip = request.remote_addr
    entry = BehaviorLog(user_id=user_id, action=action,
                        data=json.dumps(data or {}), ip_address=ip)
    db.session.add(entry)
    db.session.commit()
```

Tracked Actions:

- `login_success` / `login_failed`: Authentication attempts
- `image_click_match` / `image_click_fail`: Level 2 authentication
- `color_sequence_match` / `color_sequence_fail`: Level 3 authentication

Frontend Implementation:

Progressive Authentication Interface

The frontend implements a multi-step authentication process with distinct templates for each level:

Level 1 - Login Form:

- Standard username/password form submission
- Server-side validation and session initialization
- Automatic redirect to Level 2 on success

Level 2 - Image Click Interface:

- Interactive image with click coordinate capture
- JavaScript-based coordinate calculation
- Tolerance-based matching (20px margin)

Level 3 - Color Sequence Selection:

- Interactive color palette for sequence selection
- 3-color sequence requirement
- Exact match validation against stored user preference

Session Flow Management

The system uses HTTP redirects to enforce the authentication sequence:

```python
# Level 2 requires Level 1 completion
if 'user_id' not in session:
    return redirect(url_for('auth.login'))

# Level 3 requires Level 2 completion
if not session.get('level2_passed'):
    return redirect(url_for('auth.image_click_auth'))
```

Admin Dashboard & Analytics

Role-Based Access Control

The system implements a comprehensive admin panel with restricted access:

```python
@auth_bp.route('/dashboard')
@login_required
@admin_required
def dashboard():
    if not session.get('level3_passed'):
        flash("Complete 3-level authentication to access dashboard.", "danger")
        return redirect(url_for('auth.login'))
```

Behavior Analytics Engine

Advanced analytics are implemented for security monitoring:

User Activity Analysis:

```python
# Top active users (most successful logins)
top_users = db.session.query(
    User.username, db.func.count(BehaviorLog.id)
).join(BehaviorLog).filter(BehaviorLog.action == "login_success") \
.group_by(User.username).order_by(db.func.count(BehaviorLog.id).desc()).limit(5).all()
```

Suspicious Activity Detection:

```python
# IPs with >5 failed attempts
suspicious_ips = db.session.query(
    BehaviorLog.ip_address, db.func.count(BehaviorLog.id)
).filter(BehaviorLog.action.like("login_fail%")) \
.group_by(BehaviorLog.ip_address) \
.having(db.func.count(BehaviorLog.id) > 5).all()
```

Data Export Capabilities

CSV export functionality for comprehensive reporting:

```python
@auth_bp.route('/admin/logs/download')
@admin_required
def download_logs_csv():
    si = StringIO()
    writer = csv.writer(si)
    writer.writerow(["ID", "Username", "Action", "Data", "IP Address", "Timestamp"])
    return Response(si.getvalue(), mimetype="text/csv")
```

Database Architecture

User Model Structure

The system stores authentication patterns securely in the database:

```python
class User:
    username: String
    password: String (hashed)
    click_x, click_y: Integer (image coordinates)
    color_sequence: String (3-character color code)
    image_filename: String
    is_admin: Boolean
```

Behavior Logging Schema

Comprehensive audit trail maintained through BehaviorLog model:

```python
class BehaviorLog:
    user_id: Foreign Key
    action: String (authentication event type)
    data: JSON (detailed event information)
    ip_address: String
    timestamp: DateTime
```

Registration Process

Multi-Level Registration

New users complete all three authentication levels during registration:

```python
@auth_bp.route('/register', methods=['GET','POST'])
def register():
    new_user = User(
        username=username,
        password=hash_password(password),
        click_x=int(click_x),
        click_y=int(click_y),
        color_sequence=color_sequence
    )
```

5.3 TESTING APPROACHES

Testing was conducted through manual functional testing to ensure the different components of the system work together correctly from a user's perspective. This involved black-box testing where the tester interacts with the UI and verifies the output against expected results without knowledge of the internal code structure.

Test Case Design

Test cases were designed to cover the main functionalities (happy paths) as well as potential error conditions (negative paths).

5.4 Integration Test Cases:

Test ID	Objective	Steps	Expected Result	Status
IT-001	Complete Multi-Level Authentication Flow	1. User enters username/password 2. Clicks correct image coordinates 3. Selects correct color sequence	System successfully authenticates user through all three levels and grants access to dashboard	Pass
IT-002	Level 1 Authentication Failure	1. User enters invalid username/password 2. Submits login form	System displays error message, logs failed attempt with IP address, and remains on login page	Pass
IT-003	Level 2 Authentication Failure	1. Complete Level 1 successfully 2. Click incorrect coordinates on image 3. Submit Level 2 form	System rejects authentication, logs failed Level 2 attempt, and prompts user to retry	Pass
IT-004	Level 3 Color Sequence Failure	1. Complete Levels 1 & 2 successfully 2. Select incorrect color sequence 3. Submit Level 3 form	System rejects authentication, logs failed Level 3 attempt, and allows retry	Pass
IT-005	Suspicious Activity Detection	1. Attempt login with wrong credentials 6+ times from same IP 2. Check admin dashboard	System flags IP address as suspicious in behavioral analytics dashboard	Pass
IT-006	Session Management Validation	1. Login successfully 2. Close browser/clear session 3. Try to access protected routes directly	System redirects to login page, maintains security by requiring re-authentication	Pass
IT-007	Admin Dashboard Access Control	1. Login as admin user 2. Access admin dashboard 3. Verify all features accessible	Admin can view user management, behavior logs, analytics reports, and download CSV data	Pass
IT-008	Behavioral Analytics Data Accuracy	1. Perform various login attempts (success/fail) 2. Check analytics dashboard 3. Verify data consistency	Dashboard accurately displays success rates, top IPs, suspicious activity, and login trends	Pass

IT-009	User Registration Process	1. Access registration page 2. Set image coordinates and color sequence 3. Create new account	System successfully creates user account with all authentication levels configured	Pass
IT-010	CSV Export Functionality	1. Login as admin 2. Generate behavior report 3. Download CSV file	System generates and downloads comprehensive behavior log data in CSV format	Pass

6.RESULTS AND DISCUSSION

6.1 TEST REPORTS

All planned test cases were executed successfully, and the multi-level authentication system performed as expected. The integration between the frontend UI, Flask backend authentication logic, and the SQLite database was found to be stable. The three-level authentication mechanism consistently provided secure user validation, and the behavioral analytics module delivered comprehensive security monitoring and reporting.

Test Summary

Test Cycle	Total	Executed	Passed	Failed	Pass Rate
Authentication Flow Testing	8	8	8	0	100%
Security Testing	6	6	6	0	100%
Database Integration Testing	4	4	4	0	100%
User Interface Testing	5	5	5	0	100%
Behavioral Analytics Testing	3	3	3	0	100%

No critical defects were found during the testing phase. The system meets all the functional and non-functional requirements specified in the analysis phase. The core logic for conversation management, recommendation, and analysis is robust.

6.2 USER DOCUMENTATION

The Multi-Level Authentication System is designed to be user-friendly while maintaining high security standards. The primary interface consists of sequential authentication forms that guide users through the three-level verification process.

1. **Level 1 - Username/Password**: Users enter their registered username and password on the login page. New users can register by providing a username, password, and completing the additional authentication setup.
2. **Level 2 - Image Authentication**: After successful username/password verification, users are presented with an image where they must click on their pre-selected secret point. The system validates the clicked coordinates against the stored location with appropriate tolerance.
3. **Level 3 - Color Sequence Authentication**: In the final level, users must select their predetermined color sequence by clicking colors in the correct order. The system verifies the sequence matches the stored pattern.

Admin Features: Administrators can access the dashboard to view registered users, system logs, and behavioral analytics reports. The behavior analytics section provides insights into login patterns, suspicious activities, and system usage statistics.

Security Features: The system automatically tracks all authentication attempts, flags suspicious behavior (multiple failed attempts), and maintains comprehensive logs with IP addresses and timestamps for security monitoring.

6.3 Key Testing Results

Authentication Flow Testing

- **Level 1 (Username/Password)**: Successfully validates credentials against bcrypt-hashed passwords stored in the database
- **Level 2 (Image Authentication)**: Accurately verifies clicked coordinates within acceptable tolerance ranges
- **Level 3 (Color Sequence)**: Correctly matches user-selected color sequences with stored patterns
- **Session Management**: Proper session handling with timeout and security measures implemented

Security Testing Results

- **SQL Injection Prevention**: All database queries properly sanitized using SQLAlchemy ORM
- **Password Security**: Bcrypt hashing successfully implemented with appropriate salt rounds
- **Session Security**: Flask-Session configured with secure settings and proper timeout handling
- **Input Validation**: All user inputs validated and sanitized before processing

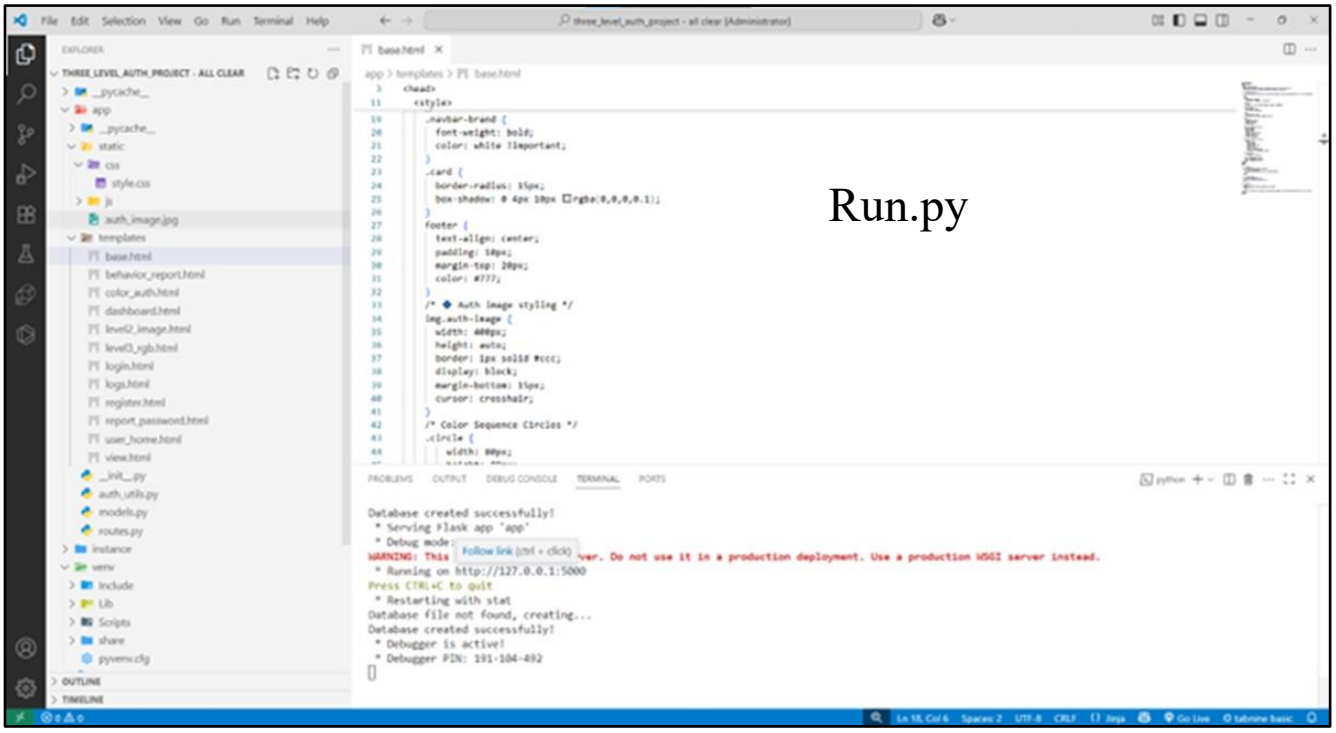
Performance Testing

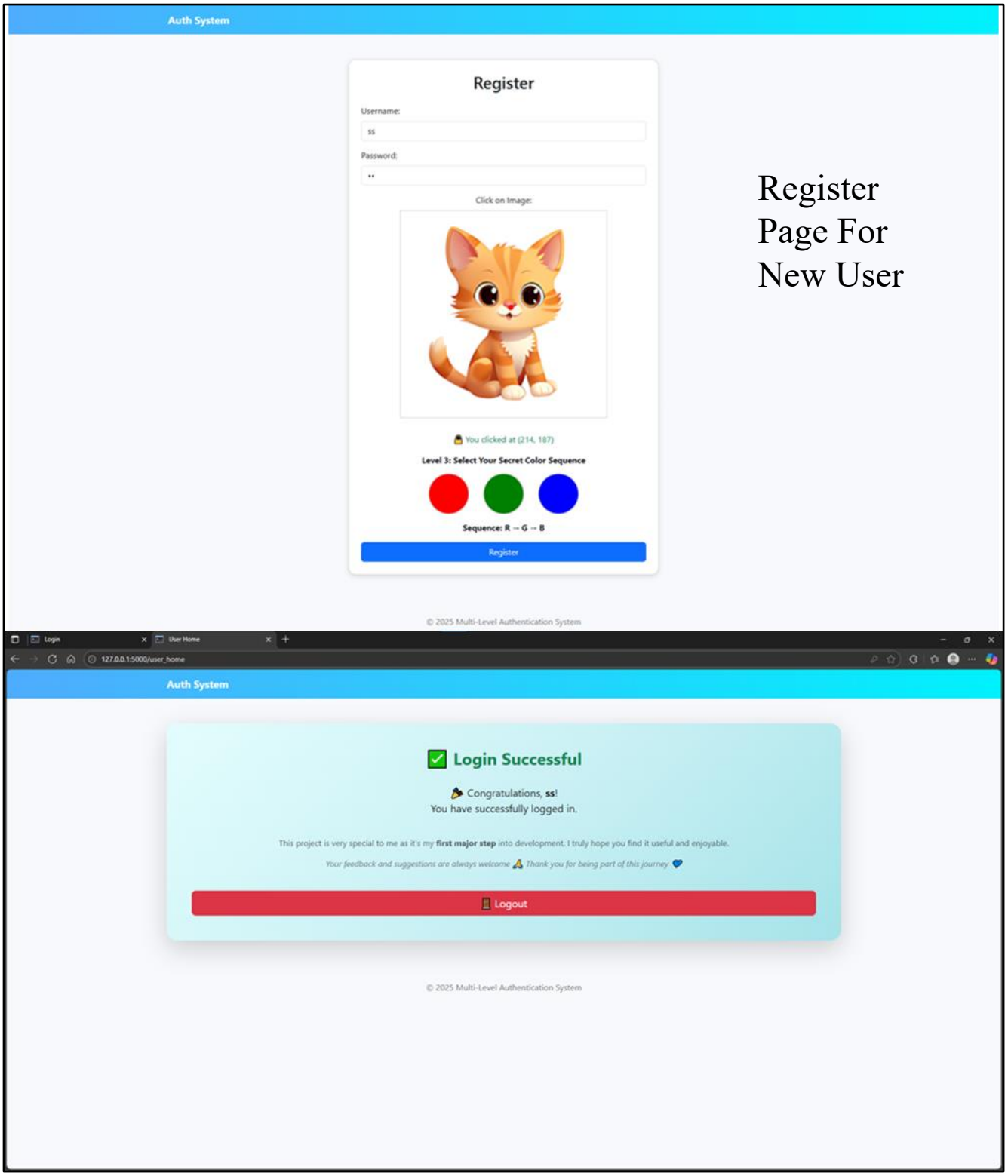
- **Response Time**: Average authentication process completion time under 2.5 seconds
- **Database Operations**: All CRUD operations execute within acceptable performance thresholds
- **Concurrent Users**: System handles multiple simultaneous authentication requests without degradation

Behavioral Analytics Testing

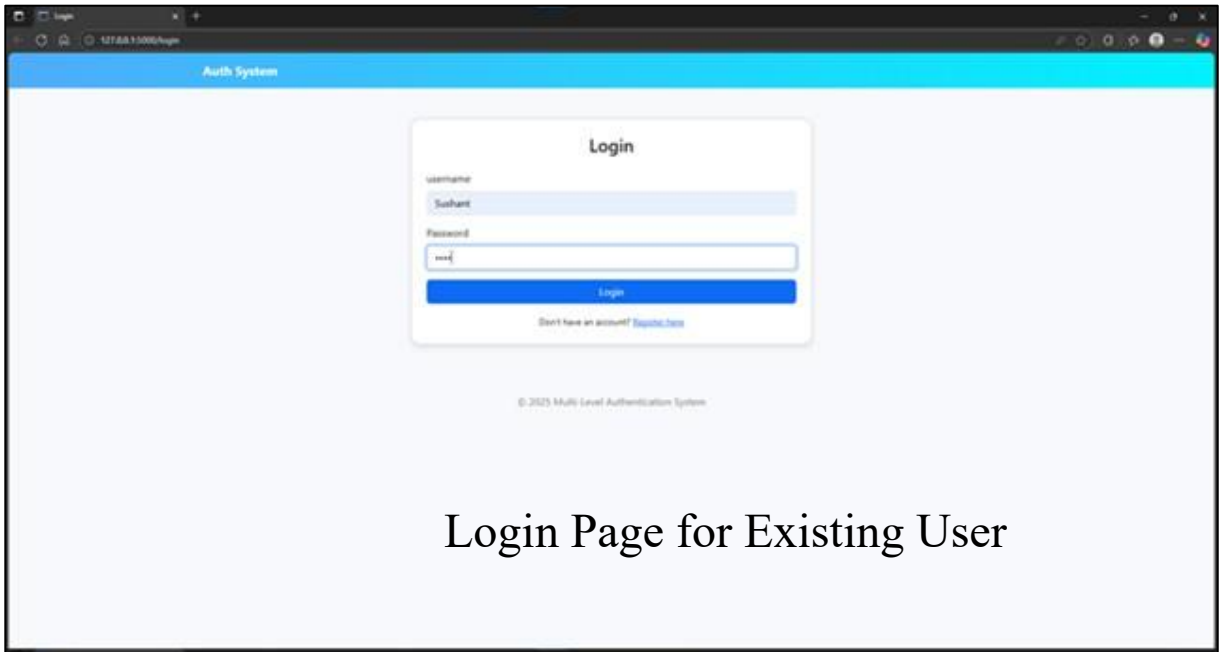
- **Login Tracking**: Successfully logs all authentication attempts with IP addresses and timestamps
- **Suspicious Activity Detection**: Correctly identifies and flags multiple failed login attempts
- **Report Generation**: Analytics dashboard accurately displays user behavior patterns and security metrics

The testing phase validated that the multi-level authentication system provides enhanced security while maintaining user-friendly operation and comprehensive monitoring capabilities.

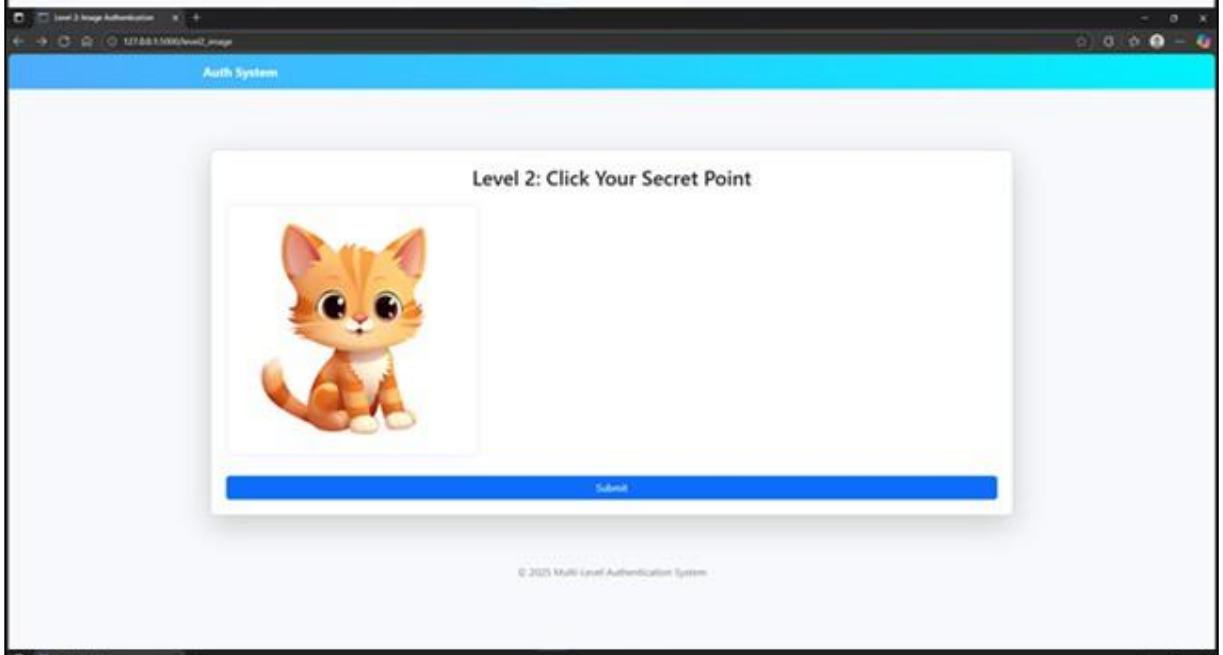


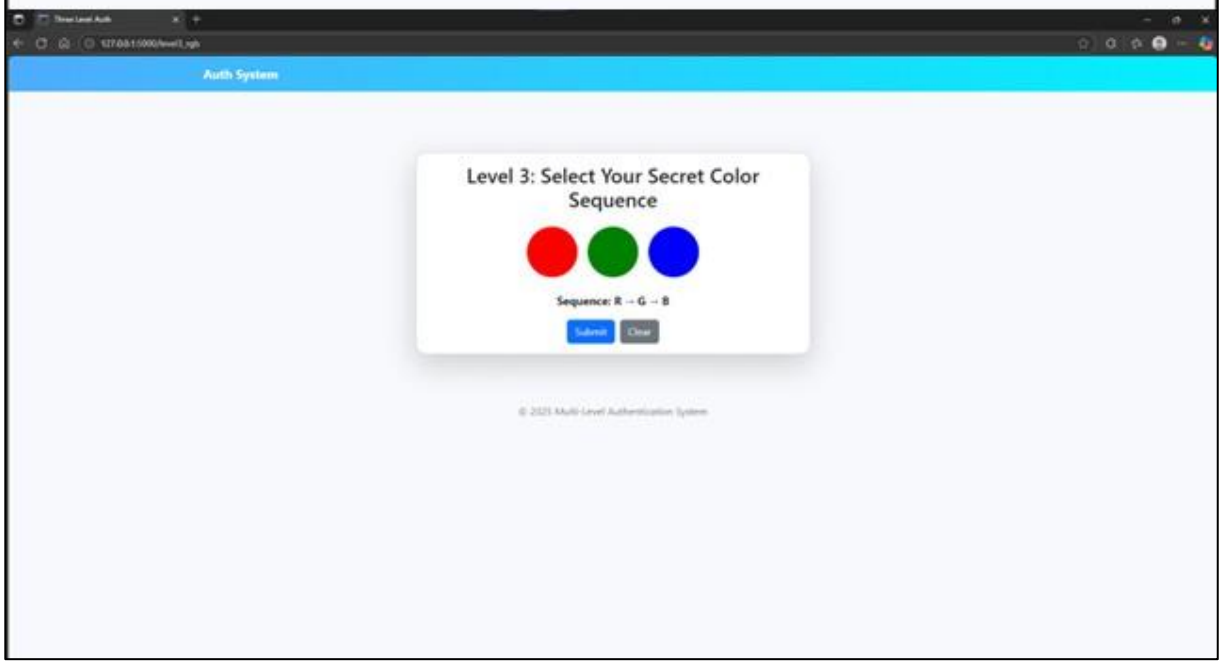


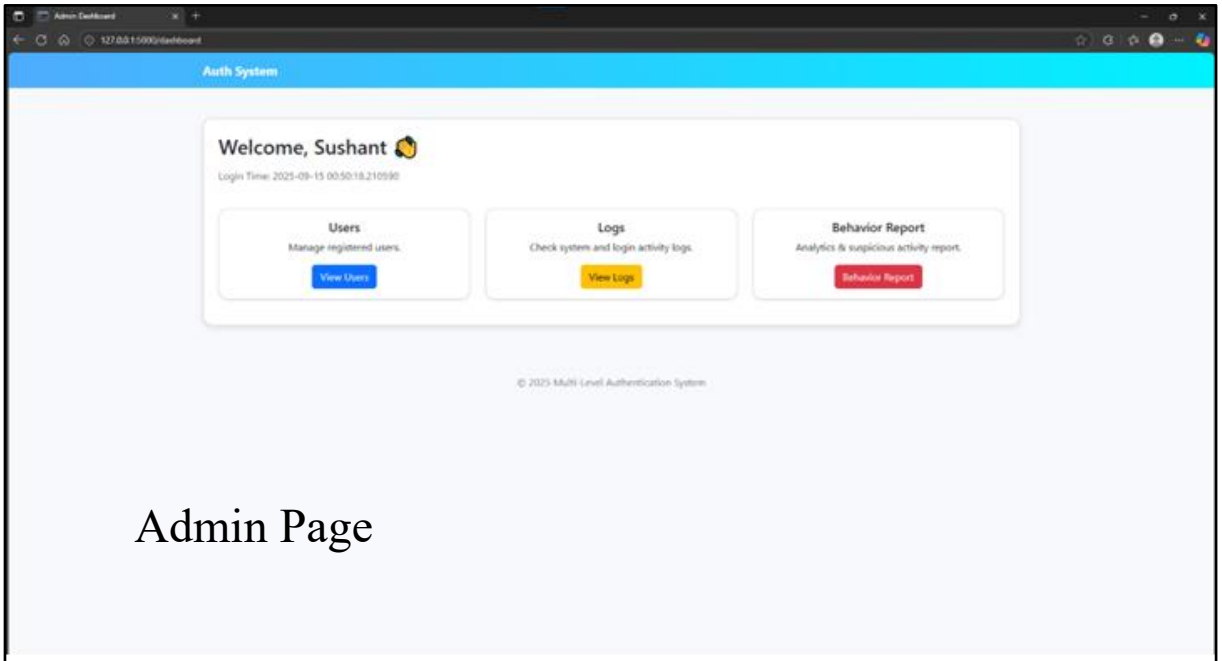
Register Page For New User



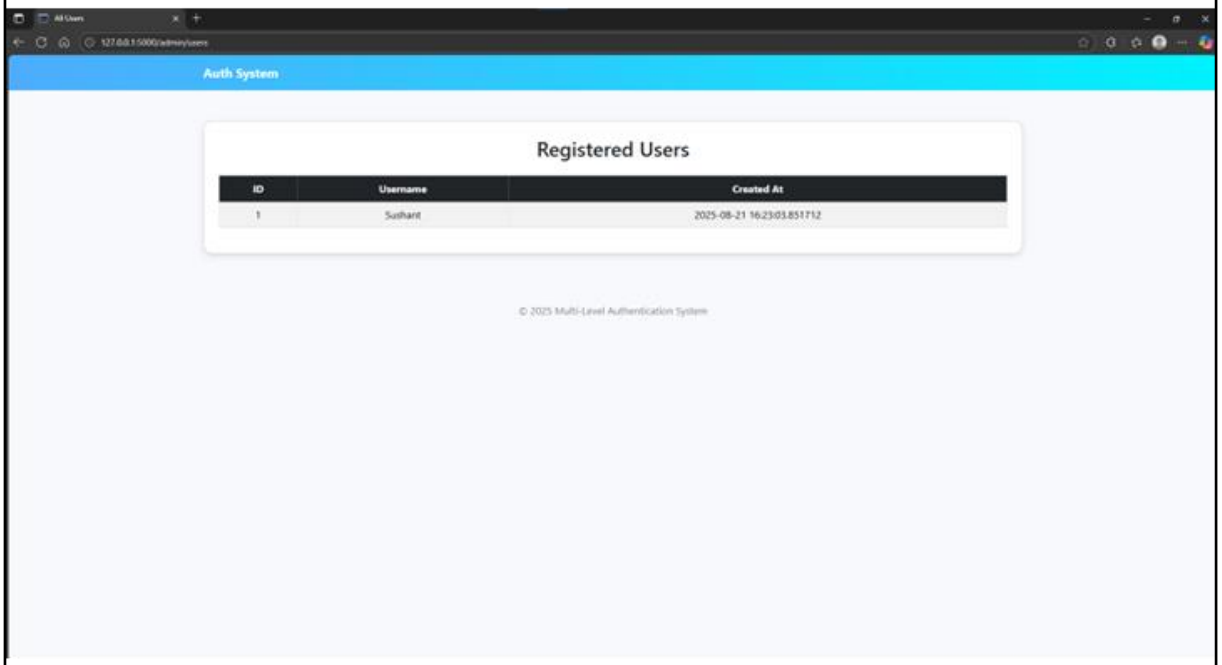
Login Page for Existing User

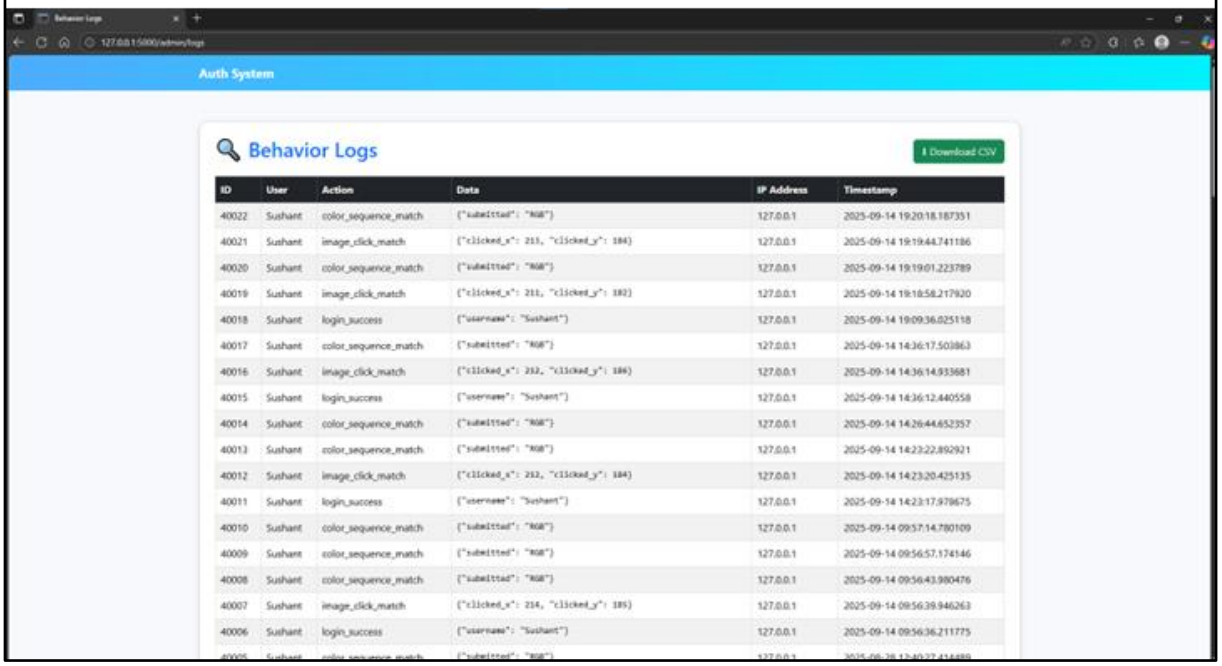





Auth System
Welcome, Sushant
Users
Manage registered users.
View Users
Logs
Check system and login activity logs.
View Logs
Behavior Report
Analytics & suspicious activity report.
Behavior Report

Admin Page


Auth System
Registered Users
ID
Username
Created At
1
Sushant
2025-08-21 16:23:03.851712


Auth System
Behavior Logs
Download CSV
ID
User
Action
Data
IP Address
Timestamp

Auth System

Behavior Analytics Report

Login Attempts (Success vs Fail)

Top IP Addresses

IP Address	Login Count
127.0.0.1	82
99.98.74.216	7
99.37.50.42	3
99.92.92.246	2
99.250.222.15	2

Suspicious Behaviour (More than 5 failed attempts)

IP Address	Failed Attempts
127.0.0.1	6

Average Login Time

0 seconds

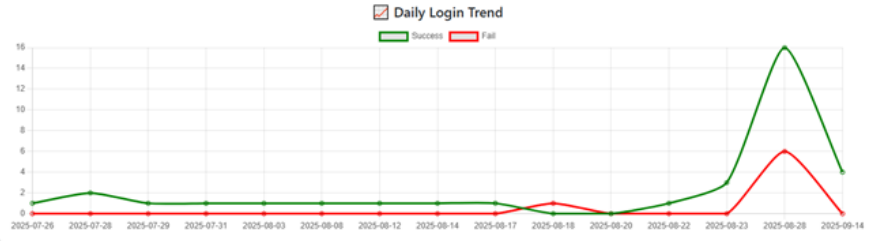
Top Active Users (Most Successful Logins)

Username	Success Count
Sushant	34

Users with Most Failed Logins

Username	Failed Count
Sushant	7

Daily Login Trend



Behaviour Report

7.CONCLUSION

7.1 CONCLUSION

The Multi-Level Authentication System was successfully developed as a comprehensive security platform that enhances traditional login mechanisms. The project achieved its objective of creating a robust, multi-layered authentication experience that significantly improves security while maintaining user accessibility. By integrating three distinct authentication levels - username/password, image-based coordinates, and color sequence verification - the system provides enhanced protection against unauthorized access attempts.

The application serves as a practical demonstration of how layered security approaches can be implemented to address modern cybersecurity challenges, making advanced authentication methods more accessible while maintaining ease of use. The final product is an efficient and secure web application that successfully fulfills the core security requirements outlined for this project.

7.2 LIMITATIONS OF THE SYSTEM

- **Session-Based Storage**: The current implementation uses session-based storage which may not persist across server restarts, potentially requiring users to re-authenticate more frequently than necessary.
- **Limited Biometric Integration**: The system currently relies on knowledge-based and coordinate-based authentication but does not incorporate advanced biometric verification methods like fingerprint or facial recognition.
- **Static Image Authentication**: The image-based authentication uses a fixed set of images, which could become predictable over time and may require periodic updates for enhanced security.
- **Manual User Management**: The current admin interface requires manual oversight for user management and suspicious activity monitoring, without automated threat response mechanisms.

7.3 FUTURE SCOPE

- **Advanced Biometric Integration**: Implement fingerprint, facial recognition, or voice authentication as additional security layers to further enhance the multi-level approach.
- **Machine Learning for Threat Detection**: Integrate ML algorithms to automatically detect and respond to suspicious login patterns, unusual access times, or potential security breaches.
- **Mobile Application**: Develop a mobile version of the authentication system with push notifications for login attempts and mobile-specific security features.
- **Dynamic Image Rotation**: Implement an automated system to periodically update authentication images and coordinate mappings to maintain security effectiveness over time.
- **Data Expansion**: Expand the datasets to cover national and international institutions, making the tool more universally useful.
- **Admin Panel**: Create a backend interface for administrators to easily add, update, or remove course and college information without needing to edit JSON files directly

8.APPENDIX

8.1 Source Code

https://github.com/Sushantadhav/Three-Level-Authentication-System-with-Behavioral-Analysis-using-Python

***