**1.1 TITLE OF THE PROJECT:**

Graphical Password by Image Segmentation

**1.2 INTRODUCTION OF THE PROJECT:**

In the age of digital transformation, ensuring the security of online accounts has become more critical than ever. Traditional password systems, which rely on alphanumeric characters, are vulnerable to attacks such as brute force and phishing. To address these concerns, innovative authentication methods like Graphical Passwords have gained traction. These systems leverage images or visual elements instead of text-based passwords, offering a more user-friendly and potentially more secure alternative.

One such method involves Image Segmentation, where an image is divided into multiple segments, and the user is required to select specific regions or objects within the image as their password. This adds an additional layer of security because, unlike text passwords, graphical passwords are harder to guess or intercept. Moreover, they make use of cognitive recognition, which is more intuitive for users, especially in cases where they may struggle to remember complex alphanumeric strings.

**1.3 OBJECTIVE OF THE PROJECT:**

Enhanced Security through Graphical Passwords:

To provide a more secure alternative to traditional alphanumeric passwords by using images as the basis for user authentication.

Image segmentation adds a layer of unpredictability, making it difficult for attackers to use conventional password-cracking techniques such as brute force.

User-Friendly and Intuitive Interface:

To design a simple, interactive, and visually appealing frontend that allows users to easily select image segments for both registration and login.

To enhance user experience by providing an authentication method that is easier to remember and use than typical text-based passwords.

Implementation of Image Segmentation for Dynamic Password Creation:

To break down an image into smaller segments, which users can select during registration to create a unique password.

The segmentation method ensures that the password is not only based on the image but also on specific selected regions, increasing the complexity of the authentication.

Secure User Registration and Authentication:

To develop a backend system using PHP that securely stores user credentials, including the selected image segments.

To implement user authentication that checks the correctness of selected image segments during login, ensuring only authorized users can access their accounts.

Responsive Web Design:

To create a responsive and adaptive user interface using HTML,CSS and java script ensuring the system functions smoothly across both desktop and mobile devices.

To provide a consistent experience regardless of the device being used for authentication.

Database Integration for Storing Credentials:

To implement a secure database (e.g., MySQL) that stores user details, including the hashed image segment choices, ensuring privacy and preventing unauthorized access.

Promote Ease of Use while Maintaining Robust Security:

To design a system that balances the ease of use of graphical passwords with strong security measures that make it difficult for attackers to compromise the system.

**1.4 SCOPE OF THE PROJECT:**

**1. Authentication Mechanism:**

Image-based Authentication: The primary focus of the system is to develop a password authentication system based on selecting segments from a grid of images, replacing traditional text passwords.

Image Segmentation: The images used for authentication will be divided into smaller segments (like a 3x3 or 4x4 grid), and users will select specific segments as their "password." These selections will be stored in the backend.

Password Creation and Authentication: The system will allow users to create and authenticate their passwords based on the sequence of selected image segments. This will serve as their unique login credentials.

**2. User Registration and Login Flow:**

User Registration: When a user registers, the system will prompt them to select a base image, which will be segmented into smaller sections. The user will then choose specific sections of the image as their password.

User Login: On subsequent logins, users will be presented with the same image and will need to select the same image segments they chose during registration for successful authentication.

**3. Backend Development:**

PHP Backend: The PHP server will handle all the logic for storing user data (including the segmented image selections), processing login requests, and authenticating users. It will also manage user sessions once authenticated.

Database Integration: The system will use a MySQL database (or any relational database) to store user credentials securely, including the hashed representations of their selected image segments.

**4. Frontend Design:**

HTML/CSS Interface: The frontend will be designed using HTML5 and CSS3 to ensure a clean, responsive, and visually appealing user interface. The image grid will be displayed in a way that is easy for users to interact with, on both desktop and mobile platforms.

Interactive Grid: The users will interact with the segmented image grid, selecting the specific regions for authentication. The frontend will ensure the selected regions are correctly highlighted and sent to the backend for validation.

**5. Security Measures:**

Hashing and Encryption: The image selections (which form the "password") will be securely hashed and stored in the database to prevent unauthorized access. This ensures that even if the database is compromised, the actual image selection data remains secure.

Session Management: After authentication, the system will use secure sessions to manage user logins and track their activities.

Brute Force Protection: While the image selection process increases complexity, additional mechanisms (such as limiting the number of login attempts) may be implemented to prevent brute-force attacks.

**6. Responsive and Adaptive Design:**

Mobile and Desktop Compatibility: The frontend design will be responsive, allowing the system to work seamlessly on both desktop and mobile devices. This will ensure users can access the authentication system regardless of the device they are using.

**7. Testing and Validation:**

Usability Testing: The project will involve testing the user interface to ensure the image selection process is intuitive and easy to follow.

Security Testing: The security of the system will be tested by attempting to bypass the authentication mechanism to identify any potential vulnerabilities.

Cross-Browser Compatibility: The system will be tested on various browsers to ensure consistent behavior and visual appearance.

**8. Project Limitations:**

Image Diversity: The system’s security depends on the diversity of images available for users to choose from. The images provided for selection should be varied enough to prevent predictability in user selections.

Device Limitations: On mobile devices, the interaction with the image grid could potentially be less precise compared to desktop platforms due to the smaller screen sizes, but responsive design will help alleviate this issue.

User Experience vs. Security: While graphical passwords offer enhanced security over traditional passwords, they are still subject to certain weaknesses, such as shoulder surfing (where someone watches the user select their image segments). Additional features like visual obfuscation could mitigate this risk.

**9. Future Enhancements:**

Multi-Factor Authentication (MFA): Future versions of the system could integrate multi-factor authentication (e.g., SMS or email verification) to add an extra layer of security.

Dynamic Image Selection: Users could be provided with different sets of images for each login session, enhancing security by preventing potential attackers from predicting user selections.

Advanced Image Recognition: The system could evolve to incorporate more advanced image recognition techniques to allow for more complex authentication methods, such as combining multiple images or selecting patterns within an image.

**10. Potential Applications:**

User Account Login Systems: The graphical password authentication system can be integrated into various web applications, e-commerce websites, or online banking services to improve the user experience while ensuring higher security levels.

Mobile Applications: This graphical password system can be adapted for use in mobile apps where users can authenticate using touch-based gestures or touchscreens to select image segments.

**4.1 HARDWARE REQUIREMENT:**

**Minimum (Client/Browser Side)**

Processor: Any modern CPU (e.g., Intel i3 or better)

RAM: 2 GB minimum

Display: 1024×768 resolution

Input: Mouse or touchscreen (for image selection)

**Server-Side (for hosting)**

Processor: 2 GHz dual-core or better

RAM: 4 GB minimum

Storage: 20 GB (depends on user and image storage)

Network: Stable internet connection for user access

**1.4.2 SOFTWARE REQUIREMENT:**

**Client Side**

Web Browser: Chrome, Firefox, Edge, Safari (latest versions)

Language Support: HTML, CSS, JavaScript

**Server Side**

Web Server: Apache, Nginx, or Node.js

Language: PHP, Python (Flask/Django), or Node.js

Database: MySQL, PostgreSQL, or MongoDB

Image Handling: Basic server-side image storage and access

**1.4.3 PROGRAMMING LANGUAGES**

Front End: html, css, javascript

Back End: php

Database: MySQL

**1.5 PROJECT MODULES:**

User Interface (Frontend Module)

**Objective:**

Design the frontend for users to select image regions as their password.

**Key Components:**

HTML & CSS: Structure and style the page.

JavaScript: Handle user interactions, image rendering, and sending data to the backend.

Image Rendering: Display and segment the image for password selection.

**Sub-Modules:**

1. Image Display:

1. Show the image with clickable regions.

2. Selection Handling: Capture and highlight selected regions.

3. Data Submission: Send selected coordinates to the backend.

4. Validation Feedback: Show success or failure messages based on authentication.

2. Image Segmentation Module

**Objective:**

Break the image into segments or predefined regions that will be used as part of the password. This module can either be part of the backend or handled on the frontend.

**Key Components:**

Grid Segmentation: Divide the image into a grid pattern (e.g., 3x3, 4x4).

Custom Segmentation: Allow for specific regions of the image to be selected (e.g., user-defined or predetermined hotspots).

**Sub-Modules:**

Grid Segmentation: Create clickable areas in a grid.

Custom Reqion Detection: Define non-uniform regions if needed.

Image Preparation: Process the image into segments.

3. User Registration & Authentication Module (Backend)

**Objective:**

Handle user registration and login by comparing selected image regions.

**Key Components:**

User Registration: Store user-selected image regions.

Login Authentication: Compare login selections to stored data.

**Sub-Modules:**

1. User Registration: Capture and store selected image regions.

2. Password Storage: Store coordinates securely in the database.

3. Login Validation: Compare user selections during login.

4. Password Validation: Authenticate the user if selections match.

**4. PHP Backend & Server-Side Logic Module**

**Objective:**

Manage server-side operations for user authentication and session handling.

**Key Components:**

User Data Handling: Store and retrieve user data.

Password Verification: Compare login selections to stored coordinates.

Security: Implement security measures like rate-limiting and CAPTCHA.

**Sub-Modules:**

1. User Authentication: Handle session creation and validation.

2. Data Validation: Compare coordinates with stored values.

3. Database Integration: Store and retrieve user data.

4. Error Handling: Provide error messages on failed authentication.

**5. Database Module**

**Objective:**

Store user data and manage sessions.

Key Components:

User Information: Store user data and selected image coordinates.

Session Management: Track and manage user sessions.

**Sub-Modules:**

1. User Data Storage: Store user info and password segments.

2. Data Retrieval: Retrieve and compare password segments during login.

3. Session Data: Manage user sessions.

4. Security Measures: Securely store and retrieve data

**6. Error Handling & Security Module**

**Objective:**

Implement error handling and security measures.

Key Components:

Error Feedback: Provide feedback on authentication failure.

Security Measures: Prevent brute-force attacks and session hijacking.

**Sub-Modules:**

1. Input Validation: Ensure valid image selections.

2. Security Layers: Implement CAPTCHA, encryption, and rate-limiting.

3. Session Management: Handle session expiration and secure cookie management.

**7. Testing & Debugging Module**

**Objective:**

Ensure the system works correctly and securely.

**Key Components:**

Unit Testing: Test individual modules for correctness.

Integration Testing: Ensure all modules work together.

Security Testing: Test for vulnerabilities.

**Sub-Modules:**

1. Unit Tests: Test backend logic.

2. UI Testing: Ensure frontend usability.

3. Security Testing: Perform penetration testing to detect vulnerabilities

**1.6 Future Scope of the Graphical Password System Using Image Segmentation**

**Advanced Image Segmentation:**

Use machine learning and AI for dynamic, context-aware segmentation of images.

Allow users to define custom regions or implement object detection for segmentation.

**Multi-Factor Authentication (MFA):**

Integrate with text-based passwords, biometrics (fingerprint, face recognition), or voice authentication for stronger security.

**Mobile & Touchscreen Compatibility:**

Optimize the system for mobile devices and touchscreen interfaces, allowing tap and drag interactions.

Explore Augmented Reality (AR) for selecting regions in real-world objects.

**Enhanced Security:**

Encrypt stored image segment data and integrate intrusion detection systems.

Implement adaptive security features like increased authentication requirements for suspicious logins.

**User Experience Improvements:**

Create interactive tutorials and enhance accessibility (voice guidance, keyboard navigation).

Allow users to upload custom images for password creation.

**Scalability:**

Use cloud computing, microservices, and load balancing for large-scale deployments.

Explore decentralized authentication with blockchain for increased privacy.

Integration with IoT Devices:

Implement authentication across IoT devices (e.g., smartwatches, home assistants) using graphical passwords.

**Privacy-Preserving Techniques:**

Use Zero-Knowledge Proofs (ZKPs) to allow secure authentication without revealing passwords.

Ensure privacy by processing data locally, reducing central data storage.

**Emerging Technologies:**

Adapt to AI-powered authentication and explore brain-computer interfaces for a hands-free, ultra-secure login method.

**Advantages:**

**Enhanced Security**: Harder to guess than alphanumeric passwords, especially with image segmentation.

**Usability:** More intuitive and memorable for users, especially for those with limited typing skills.

**Phishing Resistance:** Less vulnerable to phishing attacks due to reliance on images.

**Reduced Keylogging Risk:** No need to type, reducing chances of keylogger attacks.

**Backend Scalability:** PHP can handle image segmentation and integrate easily with web apps.

**Disadvantages:**

**Complex Implementation:** Requires advanced logic for image segmentation, which can slow down performance.

**UI Complexity:** Can be harder to use for people with visual impairments or those unfamiliar with graphical interfaces.

**Security Risks:** Storing and manipulating images can expose vulnerabilities if not done securely.

**Resource Consumption:** More bandwidth and storage are needed for image-based data.

Scalability Issues: Managing a large volume of image data can become challenging as the user base grows.

**2. Introduction**

**2.1 Purpose**

This Software Requirements Specification (SRS) defines the functional and non-functional requirements of a software system that involves graphical password-based authentication and image segmentation. The system aims to provide secure, user-friendly access and image processing mechanisms. This document is intended for developers, testers, designers, and stakeholders involved in the development lifecycle.

**2.2 Overall Description**

**Product Perspective**

* This software is a standalone web-based or desktop-based application. It follows a modular architecture with defined subsystems:
* Authentication module (Graphical password)
* Image Segmentation module
* User Interface
* Data Management subsystem
* Product Functions
* User registration with graphical password selection
* Secure login using selected image patterns
* Uploading and segmenting images for processing
* Data storage and retrieval through backend databases
* Administrative panel for managing users and logs

**2.3 Project modules:**

**2.3.1 User Interface (Frontend Module)**

**Objective:**

Design the frontend for users to select image regions as their password.

**Key Components:**

HTML & CSS: Structure and style the page.

JavaScript: Handle user interactions, image rendering, and sending data to the backend.

Image Rendering: Display and segment the image for password selection.

**Sub-Modules:**

1.Image Display: Show the image with clickable regions.

2. Selection Handling: Capture and highlight selected regions.

3. Data Submission: Send selected coordinates to the backend.

4. Validation Feedback: Show success or failure messages based on authentication.

**2. 3.2 Image Segmentation Module**

**Objective:**

Break the image into segments or predefined regions that will be used as part of the password. This module can either be part of the backend or handled on the frontend.

Key Components:

Grid Segmentation: Divide the image into a grid pattern (e.g., 3x3, 4x4).

Custom Segmentation: Allow for specific regions of the image to be selected (e.g., user-defined or predetermined hotspots).

**Sub-Modules:**

1. Grid Segmentation: Create clickable areas in a grid.
2. Custom Reqion Detection: Define non-uniform regions if needed.
3. Image Preparation: Process the image into segments.

**2.3.3 User Registration & Authentication Module (Backend)**

**Objective:**

* Handle user registration and login by comparing selected image regions.
* Key Components:
* User Registration: Store user-selected image regions.
* Login Authentication: Compare login selections to stored data.

**Sub-Modules:**

1. User Registration: Capture and store selected image regions.

2. Password Storage: Store coordinates securely in the database.

3. Login Validation: Compare user selections during login.

4. Password Validation: Authenticate the user if selections match.

**2.3.4 PHP Backend & Server-Side Logic Module**

**Objective:**

Manage server-side operations for user authentication and session handling.

**Key Components:**

* User Data Handling: Store and retrieve user data.
* Password Verification: Compare login selections to stored coordinates.
* Security: Implement security measures like rate-limiting and CAPTCHA.

**Sub-Modules:**

1. User Authentication: Handle session creation and validation.

2. Data Validation: Compare coordinates with stored values.

3. Database Integration: Store and retrieve user data.

4. Error Handling: Provide error messages on failed authentication.

**2.3.5 Database Module**

**Objective:**

Store user data and manage sessions.

**Key Components:**

* User Information: Store user data and selected image coordinates.
* Session Management: Track and manage user sessions.

**Sub-Modules:**

1. User Data Storage: Store user info and password segments.

2. Data Retrieval: Retrieve and compare password segments during login.

3. Session Data: Manage user sessions.

4. Security Measures: Securely store and retrieve data

**2.3.6 Error Handling & Security Module**

**Objective:**

Implement error handling and security measures.

**Key Components:**

* Error Feedback: Provide feedback on authentication failure.
* Security Measures: Prevent brute-force attacks and session hijacking.

**Sub-Modules:**

1. Input Validation: Ensure valid image selections.

2. Security Layers: Implement CAPTCHA, encryption, and rate- limiting.

3. Session Management: Handle session expiration and secure cookie management.

**2.3.7 Testing & Debugging Module**

**Objective:**

Ensure the system works correctly and securely.

**Key Components:**

* Unit Testing: Test individual modules for correctness.
* Integration Testing: Ensure all modules work together.
* Security Testing: Test for vulnerabilities.

**Sub-Modules:**

1. Unit Tests: Test backend logic.

2. UI Testing: Ensure frontend usability.

3.Security Testing: Perform penetration testing to detect vulnerabilities

**3.1 Future Scope of the Graphical Password System Using Image Segmentation**

**1. Authentication Mechanism:**

Image-based Authentication: The primary focus of the system is to develop a password authentication system based on selecting segments from a grid of images, replacing traditional text passwords.

Image Segmentation: The images used for authentication will be divided into smaller segments (like a 3x3 or 4x4 grid), and users will select specific segments as their "password." These selections will be stored in the backend.

Password Creation and Authentication: The system will allow users to create and authenticate their passwords based on the sequence of selected image segments. This will serve as their unique login credentials.

**2. User Registration and Login Flow:**

User Registration: When a user registers, the system will prompt them to select a base image, which will be segmented into smaller sections. The user will then choose specific sections of the image as their password.

User Login: On subsequent logins, users will be presented with the same image and will need to select the same image segments they chose during registration for successful authentication.

**3. Backend Development:**

PHP Backend: The PHP server will handle all the logic for storing user data (including the segmented image selections), processing login requests, and authenticating users. It will also manage user sessions once authenticated.

Database Integration: The system will use a MySQL database (or any relational database) to store user credentials securely, including the hashed representations of their selected image segments.

**4. Frontend Design:**

HTML/CSS Interface: The frontend will be designed using HTML5 and CSS3 to ensure a clean, responsive, and visually appealing user interface. The image grid will be displayed in a way that is easy for users to interact with, on both desktop and mobile platforms.

Interactive Grid: The users will interact with the segmented image grid, selecting the specific regions for authentication. The frontend will ensure the selected regions are correctly highlighted and sent to the backend for validation.

**5. Security Measures:**

* Hashing and Encryption: The image selections (which form the "password") will be securely hashed and stored in the database to prevent unauthorized access. This ensures that even if the database is compromised, the actual image selection data remains secure.
* Session Management: After authentication, the system will use secure sessions to manage user logins and track their activities.
* Brute Force Protection: While the image selection process increases complexity, additional mechanisms (such as limiting the number of login attempts) may be implemented to prevent brute-force attacks.

**6. Responsive and Adaptive Design:**

Mobile and Desktop Compatibility**:** The frontend design will be responsive, allowing the system to work seamlessly on both desktop and mobile devices. This will ensure users can access the authentication system regardless of the device they are using.

**7. Testing and Validation:**

* Usability Testing: The project will involve testing the user interface to ensure the image selection process is intuitive and easy to follow.
* Security Testing: The security of the system will be tested by attempting to bypass the authentication mechanism to identify any potential vulnerabilities.
* Cross-Browser Compatibility: The system will be tested on various browsers to ensure consistent behavior and visual appearance.

**8. Project Limitations:**

* Image Diversity: The system’s security depends on the diversity of images available for users to choose from. The images provided for selection should be varied enough to prevent predictability in user selections.
* Device Limitations: On mobile devices, the interaction with the image grid could potentially be less precise compared to desktop platforms due to the smaller screen sizes, but responsive design will help alleviate this issue.
* User Experience vs. Security: While graphical passwords offer enhanced security over traditional passwords, they are still subject to certain weaknesses, such as shoulder surfing (where someone watches the user select their image segments). Additional features like visual obfuscation could mitigate this risk.

**9. Future Enhancements:**

* Multi-Factor Authentication (MFA): Future versions of the system could integrate multi-factor authentication (e.g., SMS or email verification) to add an extra layer of security.
* Dynamic Image Selection: Users could be provided with different sets of images for each login session, enhancing security by preventing potential attackers from predicting user selections.
* Advanced Image Recognition: The system could evolve to incorporate more advanced image recognition techniques to allow for more complex authentication methods, such as combining multiple images or selecting patterns within an image.
* 10. Potential Applications:
* User Account Login Systems: The graphical password authentication system can be integrated into various web applications, e-commerce websites, or online banking services to improve the user experience while ensuring higher security levels.
* Mobile Applications: This graphical password system can be adapted for use in mobile apps where users can authenticate using touch-based gestures or touchscreens to select image segments.

**4.1 General Constraints (Easy Version)**

**1. Password Composition**

The password must consist of 3 to 6 image segments.

The selected segments must follow a specific sequence (order matters).

**2. Image Segmentation Rules**

The original image should be divided into a uniform grid (e.g., 4×4 or 5×5).

Each segment is uniquely identifiable by its position (row/column or ID).

Segments should be visually distinct enough to avoid confusion.

**3. Security Constraints**

Enforce selection from distinct zones (e.g., top-left, bottom-right) to reduce guess ability and prevent shoulder surfing.

**4. Usability Guidelines**

Users must be able to easily recognize and select segments with a click or tap.

Encourage users to select personally meaningful segments for better memorability.

**5. Randomized Segment Positions**

Prevents attackers from learning the password based on mouse movement or screen position.

**4.2 HARDWARE REQUIREMENT:**

**Minimum (Client/Browser Side)**

Processor: Any modern CPU (e.g., Intel i3 or better)

RAM: 2 GB minimum

Display: 1024×768 resolution

Input: Mouse or touchscreen (for image selection)

**Server-Side (for hosting)**

Processor: 2 GHz dual-core or better

RAM: 4 GB minimum

Storage: 20 GB (depends on user and image storage)

Network: Stable internet connection for user access

**4.3 SOFTWARE REQUIREMENT:**

**Client Side**

Web Browser: Chrome, Firefox, Edge, Safari (latest versions)

Language Support: HTML, CSS, JavaScript

**Server Side**

Web Server: Apache, Nginx, or Node.js

Language: PHP, Python (Flask/Django), or Node.js

Database: MySQL, PostgreSQL, or MongoDB

Image Handling: Basic server-side image storage and access

**System Attributes**

* **Reliability**

1. The system should work without crashing or errors.
2. It must provide consistent performance across sessions.
3. It should ensure data is not lost during operations like login or image upload.

* **Availability**

1. The system must be available for use at least 99% of the time.
2. It should support multiple users accessing the system at the same time.

* **Security**

1. Passwords (graphical or text) must be stored in encrypted format.
2. The system should prevent unauthorized access by using CAPTCHA and session timeouts.

* **Usability**

1. The interface should be user-friendly and easy to navigate.
2. Graphical password selection should be clear and intuitive.

* **Maintainability**

1. Errors should be logged for easy debugging and system improvement.

* **Portability**
  1. The system should work on both desktop and laptop environments.
  2. It should support multiple web browsers (Chrome, Firefox, Edge).
  3. Should be installable and usable on both Windows and Linux OS.

**3.1 INTRODUCTION**

System Design is the process or art of defining the architecture, components modules, interfaces and data for a system to satisfy specified requirements. One could see it as the application of the systems theory to system analysis, systems architecture and systems engineering. In system design focus is on deciding which module are needed for the system. the specification of this modules should be interconnected is called system design

System Design is also called top-level design. Here we consider a system to be set of components with clearly defined behaviour that interact with each other in a fixed manner to produce sum behaviour. In a system design, the design consists of module definition with each module supporting a functional abstraction.

**3.2 ASSUMPTION AND CONSTRAINTS:**

**3.2.1 ASSUMPTION:**

In system design, assumptions are the foundational beliefs or expectations that help guide the design process by simplifying complex problems and focusing on key factors. These assumptions set boundaries for how the system will be designed, implemented, and used. They are necessary but should be validated during testing and deployment.

**3.2.2 CONSTRAINTS:**

Graphical passwords and image segmentation are two important fields that enhance user authentication and image processing, respectively. Graphical passwords utilize images for secure and user-friendly authentication, addressing usability and security challenges. Meanwhile, image segmentation involves dividing an image into meaningful parts, crucial for various applications like object recognition and image analysis. Both areas must navigate constraints related to usability, security, algorithm complexity, and variability in content to deliver effective solutions.

**3.3 FUNCTIONAL DECOMPOSITION**

Functional decomposition refers broadly to the process of dissolving a functional relationship into its constituent parts in such a way that the original function can be reconstructed (i.e., recomposed) from those parts by function composition. In general, this process of decomposition is undertaken either for the purpose of gaining inside into the identity of the constituent components or for the purpose of obtaining a compressed representation of global function, a task which a feasible only when the constituent processes a certain level of modularity. This section identifies functional components of the software package.

**3.3.1 SYSTEM SOFTWARE ARCHITECTURE**

**SYSTEM ARCHITECTURE**

Store Image

Segment Storage

Store Image

Store Image Data

Segment Storage

Retrieve Segments

Image Input

Credential Storage

Password Verification

Authentication Server

Graphical Password System

Stores Segments

Password

Image Database

Segment Storage

Registration

User

Fig.3.3.1 System Software Architecture

**3.3.2 SYSTEM TECHNIQUE ARCHITECTURE**

Technique architecture is a form of IT architecture that is used to design the computer system. It involves the development of technical blueprint with regard to arrangement interaction, interdependence of all elements so that system-relevant requirement is met.

**3.3.3 SYSTEM HARDWARE ARCHITECTURE**

Hardware architecture is the representation of an engineered (or to be engineered) electronic or electromechanical hardware system, and the process and discipline for effectively implementing the design for such system.

**3.4 DESCRIPTION OF PROGRAMS**

**3.4.1 CONTEXT FLOW DIAGRAM (LEVEL 0)**

Context flow diagram is atop-level data flow diagram. It only contains one process node that generalizes the function of the entire system in relationship to external entities. In context flow diagram the entire system is treated as a single process and all its inputs, outputs, sinks and sources are identified and shown.

Fig.3.4.1 Context Flow Diagram

Graphical Password of Authentication System

Log in

Registration

Step 6

User gets login screen

Step 1

Users enters details

Step 7

Select images & click grid point in sequence

Step 2

User selects one image from five of his/her choice

Step 8

Data sends to the Server

Step 3

Divide an Image into Segment and select grid points in sequence

Step 9

Server produces encrypted Password by accessing Grid values from fixed Database

Step 4

The same image is shown again. The user must select the same segment they selected in Step 3.

Password matches?

Go to step 5

Step 5

If both selections match Registration is Successful

Yes

Step 11

Allow surfing on application

**3.4.2 DATA FLOW DIAGRAM**

A Data Flow Diagram is a graph showing the flow of data values from their sources in objects through processes that transform them to their destination in other objects.

A DFD Also known as “bubble chart”, has the purpose of clarifying the system requirements and identifying major transformations that will become programs in system design. So, it is the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformations and the lines represent data flows in the system. A DFD consists of processes that transform data, data flows that move data, actor objects that produce and consume data, and data store objects that store data passively.

**Data Flow Diagram Symbol**

|  |  |  |
| --- | --- | --- |
| Name | Symbol | Description |
| Process |  | A process shows a transformation or manipulation of data flows within the system. A process transforms incoming data flow into outgoing data flow |
| Data Flow |  | A data flow shows flow of information from source to destination. A data flow represented by a line, with arrowhead showing the direction of flow |
| External Entities |  | External entities are outside the system. em, but they either supply input data into the system or use system output. External entities represented by a rectangle |
| Data Store |  | A data store indicates to which database a particular data is stored and from which database the data is retrieved |

**DATA FLOW DIAGRAM (LEVEL 1)**

INFORMATION REQUEST

**GPIS**

**USER**

**Data Flow Diagram (Level 2)**

Retrieve Picture

Registration

Information

IMAGE

DATABASE

Password

**USER**

information

Login page

LOG

Request

information

Login result

Retrieve information

**3.5 UML DIAGRAM**

**3.5.1 USE CASE DIAGRAM**

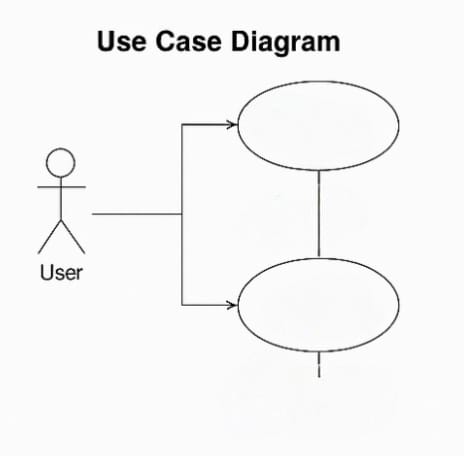
Use-case diagrams model the behaviour of a system and help to capture the requirements of the system.

Use-case diagrams describe the high-level functions and scope of a system.

These diagrams also identify the interactions between the system and its actors.

Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system.





Select segments

Register

Fig.3.5.1 Use Case Diagram

**3.5.2 Sequence Diagram**

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios.

It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time.

In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page.

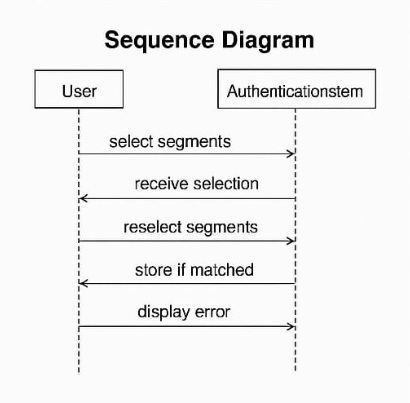


Fig. 3.5.2Sequence Diagram

**3.5.3 Activity Diagram**

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.

The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs.

The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc

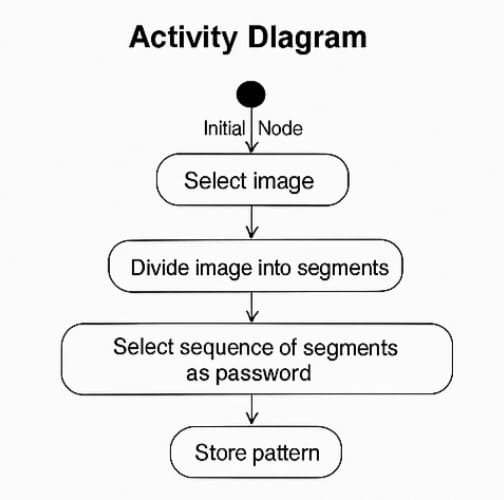




Fig.3.5.3 Activity diagram

**3.5.4 Class Diagram**

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them.

A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system.

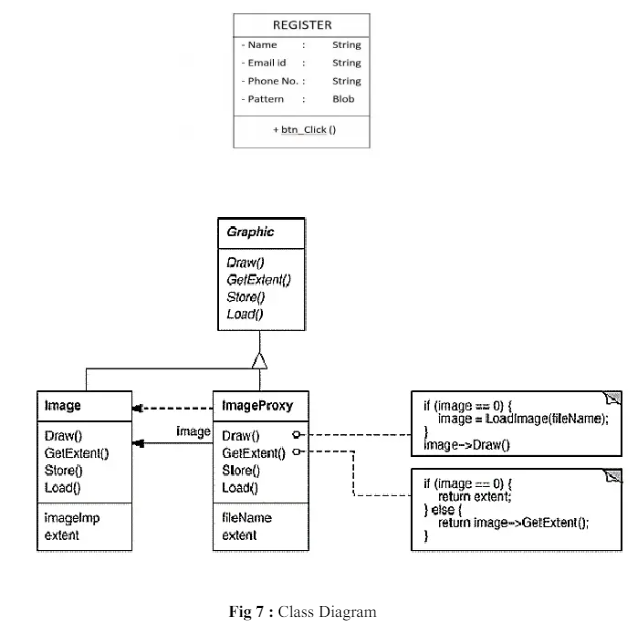


    Fig. 3.5.4 Class Diagram

4.1 DATABASE DESIGN

A database design is a collection of stored data organized in such a way that the data requirements are satisfied by the database. The general objective is to make information access easy, quick, inexpensive and flexible for the user. There are also some specific objectives like controlled redundancy from failure, privacy, security and performance. A collection of relative records make up a table. To design and store data to the needed forms database tables are prepared. Two essential settings for a database are:

Primary key: The field that is unique for all the record occurrences.

Foreign key: - The field used to set relation between tables. Normalization is a technique to avoid redundancy in the tables.

4.2 PURPOSE AND SCOPE

PURPOSE

a good data base design is one that: divides your information into subject based tables to reduce radiant data. Provides access worth the information it requires to join the information in the tables together as needed. Helps support and ensure the accuracy and integrity of our information

SCOPE:

Easy to locate the data or information in no time.

No redundant data

No reception

More Security like if one is accessing or changing the data other cannot change the data at that time.

Table references are easy to maintain

4.3 DATABASE IDENTIFICATION

Microsoft sq. server and SQLite are two different data base management system, developed by Microsoft and SQLite Consortium respectively. Microsoft sq. server is a robust and scalable …

extension of .db.SQLite or .sqLIte3. Microsoft soft sql server databases Commonly use the file extensions.mad (primary data file) and if(transaction log file). database connectivity: SQLite databases are generally accessed through local file paths, while Microsoft sq. server databases are typically accessed through network connections using connection string that include server names, usernames and passwords

4.4 Schema information

Database schema design organizes the data into separate entities, determines how to create relationships between organized entities, and how to apply the constrains on the data. designers create database schemas to give other database users, such as programmers and analysts, a logical understanding of the data.

**4.5** **ER Diagram**

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

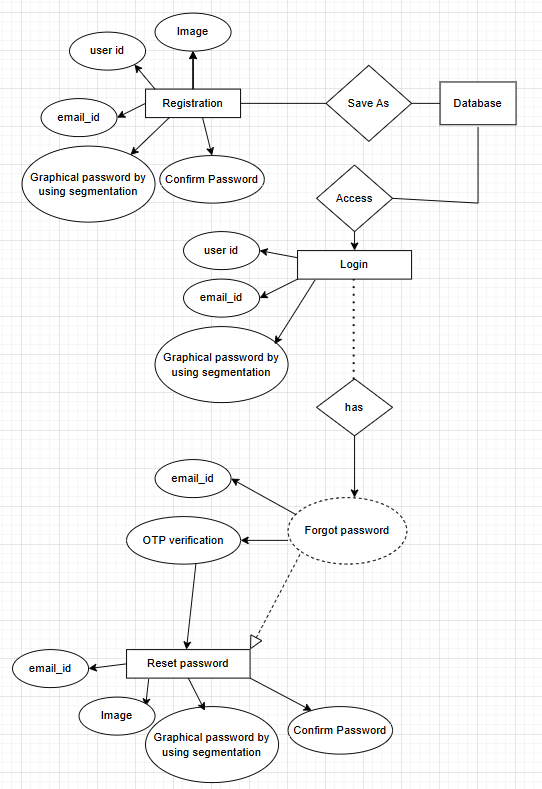


Fig. 4.5 ER Diagram

**5.1 Registration page: (html)**

**register.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Graphical Password Registration</title>

<style>

body {

font-family: Arial, sans-serif;

background: #f4f6f8;

margin: 0;

padding: 20px;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

background: #fff;

padding: 30px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

width: 400px;

}

h2 {

text-align: center;

}

.form-group {

margin-bottom: 20px;

}

input[type="text"], input[type="email"], input[type="submit"] {

width: 100%;

padding: 10px;

margin-top: 5px;

}

.image-selection img {

width: 60px;

height: 60px;

margin: 5px;

cursor: pointer;

border: 2px solid transparent;

border-radius: 5px;

}

.image-selection img.selected {

border-color: green;

}

.grid-container {

position: relative;

width: 240px;

height: 240px;

display: grid;

grid-template-columns: repeat(4, 1fr);

grid-template-rows: repeat(4, 1fr);

grid-gap: 5px;

margin-top: 10px;

background-size: cover;

background-position: center;

background-repeat: no-repeat;

}

.grid-item {

background: rgba(255,255,255,0.5);

border-radius: 5px;

cursor: pointer;

border: 1px solid #ccc;

}

.grid-item.selected {

background: rgba(0, 128, 0, 0.5);

}

.buttons {

display: flex;

justify-content: space-between;

margin-top: 20px;

}

button, input[type="submit"] {

background: #4CAF50;

color: white;

padding: 10px;

border: none;

border-radius: 5px;

cursor: pointer;

width: 45%;

}

.submit-btn:disabled {

background: gray;

}

.status {

font-size: 12px;

margin-top: 5px;

}

</style>

</head>

<body>

<div class="container">

<h2>Create Your Account</h2>

<form action="register.php" method="POST" onsubmit="return validateForm();">

<!-- Step 1 -->

<div class="step active" id="step1">

<div class="form-group">

<label>Username:</label>

<input type="text" name="userid" id="userid" onkeyup="checkUsername(this.value)" required>

<div id="userid-status" class="status"></div>

</div>

<div class="form-group">

<label>Email (Gmail):</label>

<input type="email" name="email" id="email" onkeyup="checkGmail(this.value)" required>

<div id="email-status" class="status"></div>

</div>

<div class="buttons">

<button type="button" onclick="history.back();">Back</button>

<button type="button" onclick="nextStep(1)">Next</button>

</div>

</div>

<!-- Step 2 -->

<div class="step" id="step2" style="display: none;">

<div class="form-group image-selection">

<label>Select Image:</label><br>

<img src="images/img1.jpeg" onclick="selectImage(this)">

<img src="images/img2.jpeg" onclick="selectImage(this)">

<img src="images/img3.jpeg" onclick="selectImage(this)">

<img src="images/img4.jpeg" onclick="selectImage(this)">

<img src="images/img5.jpeg" onclick="selectImage(this)">

</div>

<div class="buttons">

<button type="button" onclick="history.back();">Back</button>

<button type="button" onclick="nextStep(2)">Next</button>

</div>

</div>

<!-- Step 3 -->

<div class="step" id="step3" style="display: none;">

<div class="form-group">

<label>Select Graphical Password:</label>

<div id="password-grid" class="grid-container"></div>

</div>

<div class="form-group">

<label>Confirm Graphical Password:</label>

<div id="confirm-password-grid" class="grid-container"></div>

</div>

<div id="match-status" class="status"></div>

<input type="hidden" name="selected\_image" id="selected\_image">

<input type="hidden" name="password\_segments" id="password\_segments">

<input type="hidden" name="confirm\_password\_segments" id="confirm\_password\_segments">

<div class="buttons">

<button type="button" onclick="history.back();">Back</button>

<input type="submit" value="Register" class="submit-btn" disabled>

</div>

</div>

</form>

</div>

<script>

let usernameValid = false;

let gmailValid = false;

let passwordsMatch = false;

let imageSelected = false;

let selectedImageSrc = "";

function checkUsername(value) {

if (value.length < 3) {

document.getElementById('userid-status').textContent = '';

usernameValid = false;

updateSubmitButton();

return;

}

fetch('check\_username.php?userid=' + value)

.then(response => response.text())

.then(data => {

document.getElementById('userid-status').textContent = data;

usernameValid = data.includes("available");

updateSubmitButton();

});

}

function checkGmail(value) {

if (value.length < 5) {

document.getElementById('email-status').textContent = '';

gmailValid = false;

updateSubmitButton();

return;

}

fetch('check\_gmail.php?email=' + value)

.then(response => response.text())

.then(data => {

document.getElementById('email-status').textContent = data;

gmailValid = data.includes("allowed");

updateSubmitButton();

});

}

function selectImage(img) {

const imgs = document.querySelectorAll('.image-selection img');

imgs.forEach(i => i.classList.remove('selected'));

img.classList.add('selected');

selectedImageSrc = img.getAttribute('src');

document.getElementById('selected\_image').value = selectedImageSrc;

imageSelected = true;

generateGrids();

updateSubmitButton();

}

function generateGrids() {

createGrid('password-grid', selectedPassword, 'password\_segments');

createGrid('confirm-password-grid', selectedConfirmPassword, 'confirm\_password\_segments');

}

function createGrid(gridId, selectedArray, hiddenInputId) {

const grid = document.getElementById(gridId);

grid.innerHTML = '';

grid.style.backgroundImage = url('${selectedImageSrc}');

for (let i = 0; i < 16; i++) {

const cell = document.createElement('div');

cell.classList.add('grid-item');

cell.dataset.index = i + 1;

cell.addEventListener('click', () => toggleSelection(cell, selectedArray, hiddenInputId));

grid.appendChild(cell);

}

}

let selectedPassword = [];

let selectedConfirmPassword = [];

function toggleSelection(cell, selectedArray, hiddenInputId) {

const idx = parseInt(cell.dataset.index);

const index = selectedArray.indexOf(idx);

if (index === -1) {

// Add the grid index to the array

selectedArray.push(idx);

cell.classList.add('selected');

} else {

// Remove the grid index from the array

selectedArray.splice(index, 1);

cell.classList.remove('selected');

}

document.getElementById(hiddenInputId).value = selectedArray.join(',');

comparePasswords();

}

function comparePasswords() {

const password = document.getElementById('password\_segments').value;

const confirmPassword = document.getElementById('confirm\_password\_segments').value;

const matchStatus = document.getElementById('match-status');

if (password === confirmPassword && password.length > 0) {

matchStatus.textContent = "Passwords match ✅";

matchStatus.style.color = "green";

passwordsMatch = true;

} else {

matchStatus.textContent = "Passwords do not match ❌";

matchStatus.style.color = "red";

passwordsMatch = false;

}

updateSubmitButton();

}

function nextStep(stepNumber) {

const steps = document.querySelectorAll('.step');

steps.forEach(step => step.style.display = 'none');

document.getElementById('step' + (stepNumber + 1)).style.display = 'block';

}

function updateSubmitButton() {

const submitBtn = document.querySelector('.submit-btn');

submitBtn.disabled = !(usernameValid && gmailValid && passwordsMatch && imageSelected);

}

function validateForm() {

if (!passwordsMatch) {

alert('Password and Confirm Password do not match.');

return false;

}

if (!imageSelected) {

alert('Please select an image.');

return false;

}

return true;

}

</script>

</body>

</html>

**5.2 Registration page: (php)**

**register.php**

<?php

// Include database connection file

include('db\_connect.php');

// Check if the form is submitted

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

// Get POST values safely

$userid = trim($\_POST['userid']);

$email = trim($\_POST['email']);

$selected\_image = trim($\_POST['selected\_image']);

$password\_segments = trim($\_POST['password\_segments']);

$confirm\_password\_segments = trim($\_POST['confirm\_password\_segments']);

// Simple validation

if (empty($userid) || empty($email) || empty($selected\_image) || empty($password\_segments) || empty($confirm\_password\_segments)) {

die("Please fill out all the fields.");

}

// Validate that the password sequences match

if ($password\_segments !== $confirm\_password\_segments) {

die("Passwords do not match.");

}

// Check if User ID already exists

$stmt = $conn->prepare("SELECT \* FROM users WHERE userid = ?");

$stmt->bind\_param("s", $userid);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

die("User ID already exists.");

}

// Check if Email already exists

$stmt = $conn->prepare("SELECT \* FROM users WHERE email = ?");

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

die("Email already registered.");

}

// Insert new user into database

$stmt = $conn->prepare("INSERT INTO users (userid, email, selected\_image, password\_segments) VALUES (?, ?, ?, ?)");

$stmt->bind\_param("ssss", $userid, $email, $selected\_image, $password\_segments);

if ($stmt->execute()) {

// Redirect to login page after successful registration

header('Location: login.html');

exit();

} else {

echo "Error: " . $stmt->error;

}

// Close everything

$stmt->close();

$conn->close();

}

?>

**5.3 Configuration php :-**

**check\_username.php**

<?php

include('db\_connect.php');

if (isset($\_GET['userid'])) {

$userid = $\_GET['userid'];

$stmt = $conn->prepare("SELECT \* FROM users WHERE userid = ?");

$stmt->bind\_param("s", $userid);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

echo "❌ User ID is already taken.";

} else {

echo "✅ User ID is available.";

}

$stmt->close();

$conn->close();

}

?>

check\_gmail.php

<?php

include('db\_connect.php');

if (isset($\_GET['email'])) {

$email = $\_GET['email'];

$stmt = $conn->prepare("SELECT \* FROM users WHERE email = ?");

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

echo "❌ Gmail already registered.";

} else {

echo "✅ Gmail is allowed.";

}

$stmt->close();

$conn->close();

}

?>

check\_user.php

<?php

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

require\_once 'db\_connect.php';

header('Content-Type: application/json');

if (isset($\_POST['email'])) {

$email = $\_POST['email'];

// Adjust the query to check for email

$stmt = $conn->prepare("SELECT \* FROM users WHERE email = ?");

if ($stmt) {

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

echo json\_encode(["valid" => true]);

} else {

echo json\_encode(["valid" => false]);

}

$stmt->close();

} else {

echo json\_encode(["valid" => false, "error" => $conn->error]);

}

} else {

echo json\_encode(["valid" => false, "error" => "Missing email"]);

}

?>

**5.4 Login page(html)**

**login.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Login - Graphical Password</title>

<style>

body {

font-family: Arial, sans-serif;

background: #f4f6f8;

margin: 0;

padding: 20px;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

background: #fff;

padding: 30px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

width: 400px;

}

h2 {

text-align: center;

}

.form-group {

margin-bottom: 20px;

}

input[type="text"], input[type="email"], input[type="password"], input[type="submit"] {

width: 100%;

padding: 10px;

margin-top: 5px;

}

.image-selection img {

width: 60px;

height: 60px;

margin: 5px;

cursor: pointer;

border: 2px solid transparent;

border-radius: 5px;

}

.image-selection img.selected {

border-color: green;

}

.grid-container {

position: relative;

width: 240px;

height: 240px;

display: grid;

grid-template-columns: repeat(4, 1fr);

grid-template-rows: repeat(4, 1fr);

grid-gap: 5px;

margin-top: 10px;

background-size: cover;

background-position: center;

background-repeat: no-repeat;

}

.grid-item {

background: rgba(255,255,255,0.5);

border-radius: 5px;

cursor: pointer;

border: 1px solid #ccc;

}

.grid-item.selected {

background: rgba(0, 128, 0, 0.5);

}

.status {

font-size: 12px;

margin-top: 5px;

}

.submit-btn:disabled {

background: gray;

}

</style>

</head>

<body>

<div class="container">

<h2>Login to Your Account</h2>

<form action="login.php" method="POST" id="login-form">

<div class="form-group">

<label for="username">User ID:</label>

<input type="text" id="username" name="username" required>

<div id="username-status" class="status"></div>

</div>

<div class="form-group">

<label for="email">Email (Gmail):</label>

<input type="email" id="email" name="email" required>

<div id="email-status" class="status"></div>

</div>

<div class="form-group image-selection">

<label>Select Image:</label><br>

<img src="images/img1.jpeg" onclick="selectImage(this)">

<img src="images/img2.jpeg" onclick="selectImage(this)">

<img src="images/img3.jpeg" onclick="selectImage(this)">

<img src="images/img4.jpeg" onclick="selectImage(this)">

<img src="images/img5.jpeg" onclick="selectImage(this)">

</div>

<div class="form-group">

<label>Select Graphical Password:</label>

<div id="login-grid" class="grid-container"></div>

</div>

<input type="hidden" name="selected\_image" id="selected\_image">

<input type="hidden" name="password\_sequence" id="password\_sequence">

<input type="submit" value="Login" class="submit-btn" id="submit-btn" disabled>

<div class="status">

Don't have an account? <a href="register.html">Register here</a><br>

<a href="forgot\_password.html">Forgot Password?</a>

</div>

<div id="password-error" class="status"></div>

</form>

</div>

<script>

let selectedImageSrc = "";

let selectedPassword = [];

const form = document.getElementById('login-form');

const submitButton = document.getElementById('submit-btn');

const usernameInput = document.getElementById('username');

const emailInput = document.getElementById('email');

const usernameStatus = document.getElementById('username-status');

const emailStatus = document.getElementById('email-status');

const passwordError = document.getElementById('password-error');

function selectImage(img) {

const imgs = document.querySelectorAll('.image-selection img');

imgs.forEach(i => i.classList.remove('selected'));

img.classList.add('selected');

selectedImageSrc = img.getAttribute('src');

document.getElementById('selected\_image').value = selectedImageSrc;

generateGrid();

checkFormValidity();

}

function generateGrid() {

const grid = document.getElementById('login-grid');

grid.innerHTML = '';

grid.style.backgroundImage = url('${selectedImageSrc}');

for (let i = 0; i < 16; i++) {

const cell = document.createElement('div');

cell.classList.add('grid-item');

cell.dataset.index = i + 1;

cell.addEventListener('click', () => toggleSelection(cell));

grid.appendChild(cell);

}

}

function toggleSelection(cell) {

const idx = parseInt(cell.dataset.index);

if (!selectedPassword.includes(idx)) {

selectedPassword.push(idx);

cell.classList.add('selected');

} else {

selectedPassword = selectedPassword.filter(item => item !== idx);

cell.classList.remove('selected');

}

document.getElementById('password\_sequence').value = selectedPassword.join(',');

checkFormValidity();

}

function checkFormValidity() {

const username = usernameInput.value;

const email = emailInput.value;

const image = document.getElementById('selected\_image').value;

const sequence = document.getElementById('password\_sequence').value;

if (username && email && image && sequence) {

submitButton.disabled = false;

} else {

submitButton.disabled = true;

}

}

form.addEventListener('submit', function(e) {

e.preventDefault(); // Stop normal form submission

const userid = usernameInput.value;

const email = emailInput.value;

const selectedImage = document.getElementById('selected\_image').value;

const passwordSequence = document.getElementById('password\_sequence').value;

const xhr = new XMLHttpRequest();

xhr.open("POST", "login.php", true);

xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xhr.onload = function() {

if (xhr.status === 200) {

const response = JSON.parse(xhr.responseText);

if (response.valid) {

window.location.href = 'college.html';

} else {

passwordError.textContent = "Invalid password. Retry it again.";

}

}

};

xhr.send(`username=${encodeURIComponent(userid)}&email=${encodeURIComponent(email)}&selected\_image=${encodeURIComponent(selectedImage)}&password\_sequence=${encodeURIComponent(passwordSequence)}`);

});

usernameInput.addEventListener('input', checkFormValidity);

emailInput.addEventListener('input', checkFormValidity);

</script>

</body>

</html>

**5.5 Login page: (php)**

**login.php**

<?php

// db\_connect.php connection

require\_once 'db\_connect.php';

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$userid = $\_POST['username'];

$email = $\_POST['email'];

$selected\_image = $\_POST['selected\_image'];

$password\_sequence = $\_POST['password\_sequence'];

$stmt = $conn->prepare("SELECT \* FROM users WHERE userid = ? AND email = ?");

$stmt->bind\_param("ss", $userid, $email);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

$user = $result->fetch\_assoc();

if ($user['selected\_image'] === $selected\_image && $user['password\_segments'] === $password\_sequence) {

echo json\_encode(['valid' => true]); // ONLY success flag

} else {

echo json\_encode(['valid' => false]); // Wrong image or sequence

}

} else {

echo json\_encode(['valid' => false]); // No such user

}

$stmt->close();

$conn->close();

}

?>

**5.6 configuration html**

**forgot\_password.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Forgot Password</title>

<style>

body {

font-family: Arial, sans-serif;

background: #f4f6f8;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

background: white;

padding: 30px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

width: 350px;

}

h2 {

text-align: center;

}

input[type="text"], input[type="email"], input[type="submit"] {

width: 100%;

padding: 10px;

margin-top: 10px;

}

</style>

</head>

<body>

<div class="container">

<h2>Forgot Password</h2>

<form method="POST" action="send\_code.php">

<label for="username">User ID:</label>

<input type="text" id="username" name="username" required>

<label for="email">Enter your registered Gmail:</label>

<input type="email" id="email" name="email" required>

<input type="submit" value="Send Verification Code">

</form>

</div>

</body>

</html>

**send\_code.php**

<?php

// Include PHPMailer's autoloader

require 'vendor/autoload.php';

use PHPMailer\PHPMailer\PHPMailer;

use PHPMailer\PHPMailer\Exception;

session\_start();

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Safely get email input

$email = $\_POST['email'] ?? '';

$username = explode('@', $email)[0]; // Derive a username from email if not provided

// Basic validation

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

echo "Invalid email address.";

exit();

}

// Generate a random 6-digit verification code

$verificationCode = rand(100000, 999999);

// Store in session

$\_SESSION['verification\_code'] = $verificationCode;

$\_SESSION['email'] = $email;

$\_SESSION['username'] = $username;

// Send the email

$mail = new PHPMailer(true);

try {

$mail->isSMTP();

$mail->Host = 'smtp.gmail.com';

$mail->SMTPAuth = true;

$mail->Username = 'shalushalini11223344@gmail.com'; // Your Gmail

$mail->Password = 'lyxq xqfm eqpd rcjv'; // App password

$mail->SMTPSecure = PHPMailer::ENCRYPTION\_STARTTLS;

$mail->Port = 587;

$mail->setFrom('shalushalini11223344@gmail.com', 'GraphicalPassword');

$mail->addAddress($email, $username);

$mail->isHTML(true);

$mail->Subject = 'Your OTP for Password Reset';

$mail->Body = "

<p>Hello <b>$username</b>,</p>

<p>Your OTP for password reset is:</p>

<h2>$verificationCode</h2>

<p>If you didn’t request this, you can ignore this email.</p>";

$mail->send();

// Redirect to verify\_code.php

header("Location: verify\_code.php");

exit();

} catch (Exception $e) {

echo "Error sending email: {$mail->ErrorInfo}";

}

}

?>

**verify\_code.php**

<?php

session\_start();

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$enteredCode = $\_POST['code'] ?? '';

$sessionCode = $\_SESSION['verification\_code'] ?? '';

if ($enteredCode == $sessionCode) {

header("Location: reset\_password.html");

exit();

} else {

$error = "❌ Invalid verification code. Please try again.";

}

}

?>

<!-- verify\_code.php -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Enter OTP</title>

<style>

body {

font-family: Arial, sans-serif;

background: #f4f6f8;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

background: white;

padding: 30px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

width: 350px;

}

h2 {

text-align: center;

}

input[type="text"], input[type="submit"] {

width: 100%;

padding: 10px;

margin-top: 10px;

}

.error {

color: red;

text-align: center;

margin-top: 10px;

}

</style>

</head>

<body>

<div class="container">

<h2>Verify OTP</h2>

<?php if (!empty($error)): ?>

<div class="error"><?php echo $error; ?></div>

<?php endif; ?>

<form method="POST" action="verify\_code.php">

<label for="code">Enter the OTP sent to your email:</label>

<input type="text" id="code" name="code" required>

<input type="submit" value="Verify OTP">

</form>

</div>

</body>

</html>

**reset\_password.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Reset Password</title>

<style>

body {

font-family: Arial, sans-serif;

background: #f4f6f8;

margin: 0;

padding: 20px;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

background: #fff;

padding: 30px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

width: 400px;

}

h2 {

text-align: center;

}

.image-selection img {

width: 60px;

height: 60px;

margin: 5px;

cursor: pointer;

border: 2px solid transparent;

border-radius: 5px;

}

.image-selection img.selected {

border-color: green;

}

.grid-container {

width: 240px;

height: 240px;

display: grid;

grid-template-columns: repeat(4, 1fr);

grid-gap: 5px;

margin-top: 10px;

background-size: cover;

background-position: center;

background-repeat: no-repeat;

}

.grid-item {

background: rgba(255,255,255,0.5);

border-radius: 5px;

cursor: pointer;

border: 1px solid #ccc;

}

.grid-item.selected {

background: rgba(0, 128, 0, 0.5);

}

.submit-btn {

margin-top: 20px;

width: 100%;

background: #4CAF50;

color: white;

padding: 10px;

border: none;

border-radius: 5px;

cursor: pointer;

}

</style>

</head>

<body>

<div class="container">

<h2>Reset Password</h2>

<form action="reset\_password.php" method="POST" onsubmit="return validateForm();">

<!-- Email Input -->

<label for="email">Email:</label>

<input type="email" name="email" id="email" required placeholder="Enter your email"><br><br>

<!-- Image Selection -->

<div class="image-selection">

<label>Select Image:</label><br>

<img src="images/img1.jpeg" onclick="selectImage(this)">

<img src="images/img2.jpeg" onclick="selectImage(this)">

<img src="images/img3.jpeg" onclick="selectImage(this)">

<img src="images/img4.jpeg" onclick="selectImage(this)">

<img src="images/img5.jpeg" onclick="selectImage(this)">

</div>

<!-- Hidden Fields -->

<input type="hidden" name="selected\_image" id="selected\_image">

<input type="hidden" name="password\_segments" id="password\_segments">

<input type="hidden" name="confirm\_password\_segments" id="confirm\_password\_segments">

<!-- Password and Confirm Password Grid -->

<div>

<label>Password:</label>

<div id="password-grid" class="grid-container"></div>

</div>

<div>

<label>Confirm Password:</label>

<div id="confirm-password-grid" class="grid-container"></div>

</div>

<div id="match-status" style="margin-top:10px; font-size: 12px;"></div>

<input type="submit" value="Reset" class="submit-btn" disabled>

</form>

</div>

<script>

let selectedImageSrc = "";

let selectedPassword = [];

let selectedConfirmPassword = [];

function selectImage(img) {

document.querySelectorAll('.image-selection img').forEach(i => i.classList.remove('selected'));

img.classList.add('selected');

selectedImageSrc = img.getAttribute('src');

document.getElementById('selected\_image').value = selectedImageSrc;

generateGrid('password-grid', selectedPassword, 'password\_segments');

generateGrid('confirm-password-grid', selectedConfirmPassword, 'confirm\_password\_segments');

}

function generateGrid(gridId, selectedArray, hiddenInputId) {

const grid = document.getElementById(gridId);

grid.innerHTML = '';

grid.style.backgroundImage = url('${selectedImageSrc}');

for (let i = 0; i < 16; i++) {

const cell = document.createElement('div');

cell.classList.add('grid-item');

cell.dataset.index = i + 1;

cell.addEventListener('click', () => toggleSelection(cell, selectedArray, hiddenInputId));

grid.appendChild(cell);

}

}

function toggleSelection(cell, selectedArray, hiddenInputId) {

const idx = parseInt(cell.dataset.index);

const index = selectedArray.indexOf(idx);

if (index === -1) {

selectedArray.push(idx);

cell.classList.add('selected');

} else {

selectedArray.splice(index, 1);

cell.classList.remove('selected');

}

document.getElementById(hiddenInputId).value = selectedArray.join(',');

comparePasswords();

}

function comparePasswords() {

const pass = document.getElementById('password\_segments').value;

const confirm = document.getElementById('confirm\_password\_segments').value;

const matchStatus = document.getElementById('match-status');

const submitBtn = document.querySelector('.submit-btn');

if (pass === confirm && pass.length > 0) {

matchStatus.textContent = "Passwords match ✅";

matchStatus.style.color = "green";

submitBtn.disabled = false;

} else {

matchStatus.textContent = "Passwords do not match ❌";

matchStatus.style.color = "red";

submitBtn.disabled = true;

}

}

function validateForm() {

if (!selectedImageSrc) {

alert("Please select an image.");

return false;

}

if (document.getElementById('password\_segments').value === "") {

alert("Please select password segments.");

return false;

}

return true;

}

</script>

</body>

</html>

**5.7 Website page: (html)**

**college.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>XYZ College</title>

<link rel="stylesheet" href="college.css">

</head>

<body>

<!-- Header Section -->

<header>

<div class="logo">

<img src="logo.jpeg" alt="College Logo" />

</div>

<nav>

<ul>

<li><a href="#about-us">About Us</a></li>

<li><a href="#courses">Courses</a></li>

<li><a href="#faculty">Faculty</a></li>

<li><a href="#admissions">Admissions</a></li>

<li><a href="#events">Events</a></li>

<li><a href="#contact-us">Contact Us</a></li>

</ul>

</nav>

<!-- Link to Login Page -->

<a href="login.html" class="cta-button">Logout</a>

</header>

<!-- Main Banner Section -->

<section class="main-banner">

<div class="slider">

<div class="slide">

<div class="banner-text">

<h1>Welcome to Alva' s Degree College</h1>

<p>Your journey to success begins here!</p>

</div>

</div>

</div>

</section>

<!-- About Us Section -->

<section id="about-us" class="section-content">

<h2>About Us</h2>

<p>Alva’s College, Moodubidire, a highly sought-after center for higher education in the Indian state of Karnataka, nestles snugly on a grassy knoll overlooking a verdant landscape that brims with breathtaking scenic beauty. Armed with the vision of moulding a better tomorrow for the young brains through educational, cultural and sporting excellence, the college has been rendering yeoman service to society for more than two and a half decades. The institution, which is managed by Alva’s Education Foundation® (AEF), a brainchild of the visionary educationist, Dr. M. Mohan Alva, came into existence in 1998 with just two programmes having hardly twenty-five students. Over the years, Alva’s College has scaled dizzy heights spreading its name and fame at the international level attracting students not just from all over India but from different parts of the globe as well.</p>

<br><h2>Vision</h2><br><p>Moulding better tomorrow through educational, cultural and sports excellence.</p>

</section>

<!-- Courses Section -->

<section id="courses" class="section-content">

<h2>Featured Courses</h2>

<div class="course">

<h3>Computer Science</h3>

<p>Explore the latest trends in tech and software development.</p>

<ul><li>B.C.A. – Bachelor of Computer Application</li>

<li>B.Sc. – Microbiology, Chemistry, Zoology</li>

<li>B.Sc. – Microbiology, Bio-Chemistry, Botany</li>

<li>B.Sc. – Biotechnology, Chemistry, Botany</li>

<li>B.Sc. – Biotechnology, Chemistry, Zoology</li>

<li>B.Sc. – Botany, Zoology, Chemistry</li>

<li>B.Sc. – Physics, Mathematics, Computer Science

</li></ul>

</div>

<div class="course">

<h3>Arts</h3>

<p>Build the future with innovative engineering solutions.</p>

</div>

</section>

<!-- Events Section -->

<section id="events" class="section-content">

<h2>Upcoming Events</h2>

<ul>

<li><span>June 20, 2025</span> - Annual Sports Meet</li>

<li><span>July 10, 2025</span> - College Fest</li>

<li><span>August 15, 2025</span> - Admission Open Day</li>

</ul>

</section>

<!-- Footer Section -->

<footer>

<div class="contact-info">

<p>Alva's Degree College</p>

<p>Vidyagiri, Moodubidire, Dakshina Kannada – 574227, Karnataka, India.</p>

<p>Email: admission@alvascollege.org<br>

principal@alvascollege.org</p>

<p>Phone: <br>Contact Office : +91 8197617333<br>

Principal : +91 8147760394<br>

Admission Officer : +91 8884455526</p>

</div>

<div class="social-media">

<a href="https://facebook.com" target="\_blank">Facebook</a>

<a href="https://twitter.com" target="\_blank">Twitter</a>

<a href="https://instagram.com" target="\_blank">Instagram</a>

<a href="https://linkedin.com" target="\_blank">LinkedIn</a>

</div>

</footer>

<script>

document.querySelectorAll("nav a").forEach(link => {

link.addEventListener("click", function (e) {

e.preventDefault();

const targetId = this.getAttribute("href");

showSection(targetId);

});

});

function showSection(sectionId) {

// Hide all sections

const sections = document.querySelectorAll('.section-content');

sections.forEach(section => {

section.style.display = "none";

});

// Show the clicked section

const targetSection = document.querySelector(sectionId);

if (targetSection) {

targetSection.style.display = "block";

}

}

// Initially show the about us section

document.addEventListener('DOMContentLoaded', function () {

showSection('#about-us');

});

</script>

</body>

</html>

college.css

/\* Basic reset and typography \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

color: #333;

}

header {

background-color: #0044cc;

padding: 20px;

display: flex;

justify-content: space-between;

align-items: center;

}

header .logo img {

height: 100px;

}

nav ul {

display: flex;

gap: 20px;

}

nav ul li a {

color: white;

text-decoration: none;

font-size: 18px;

}

.cta-button {

background-color: #ff7f50;

padding: 10px 20px;

text-decoration: none;

color: white;

border-radius: 5px;

}

/\* Main Banner \*/

.main-banner {

position: relative;

height: 200px;

background-color: #ddd;

text-align: center;

}

.main-banner .slider .slide img {

width: 100%;

height: 100%;

object-fit: cover;

}

.banner-text {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

color: white;

}

.banner-text h1 {

font-size: 40px;

}

.banner-text p {

font-size: 20px;

}

.banner-text button {

background-color: #ff7f50;

padding: 10px 20px;

border: none;

color: white;

border-radius: 5px;

}

/\* Section Styles \*/

.section-content {

display: none; /\* Hide all sections initially \*/

padding: 40px;

margin: 20px 0;

}

.course {

display: inline-block;

width: 45%;

margin: 20px 2.5%;

background-color: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

text-align: center;

}

.course img {

width: 100%;

border-radius: 10px;

}

.course h3 {

margin-top: 15px;

font-size: 22px;

}

footer {

background-color: #333;

color: white;

padding: 20px 0;

text-align: center;

}

footer .contact-info {

margin-bottom: 20px;

}

footer .social-media a {

color: white;

margin: 0 10px;

text-decoration: none;

font-size: 18px;

}

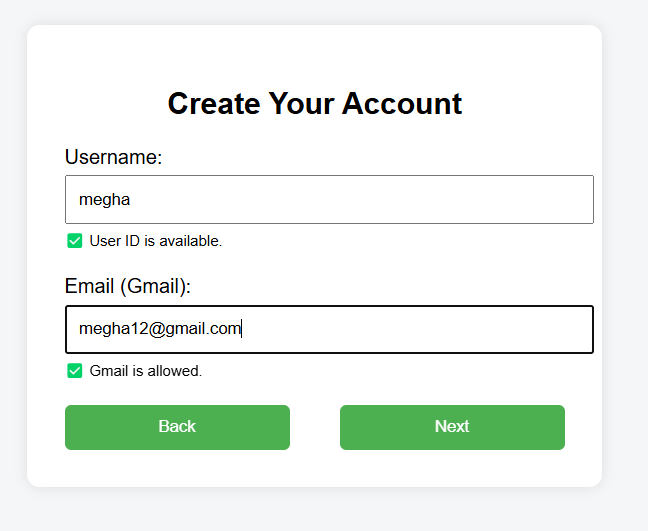
**5.8 Database connection (php)**

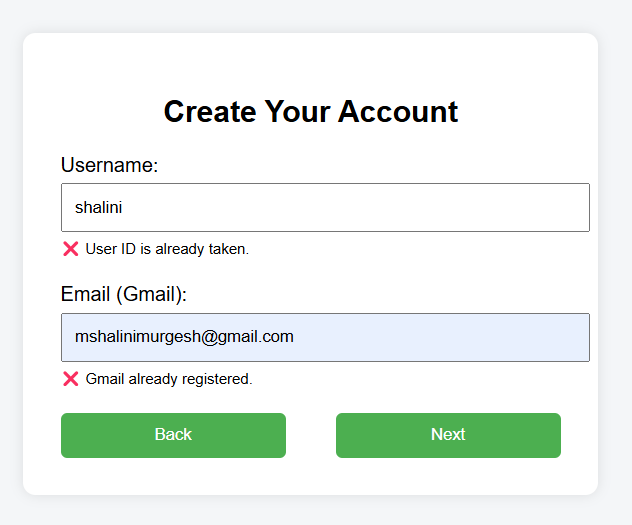
MySQL

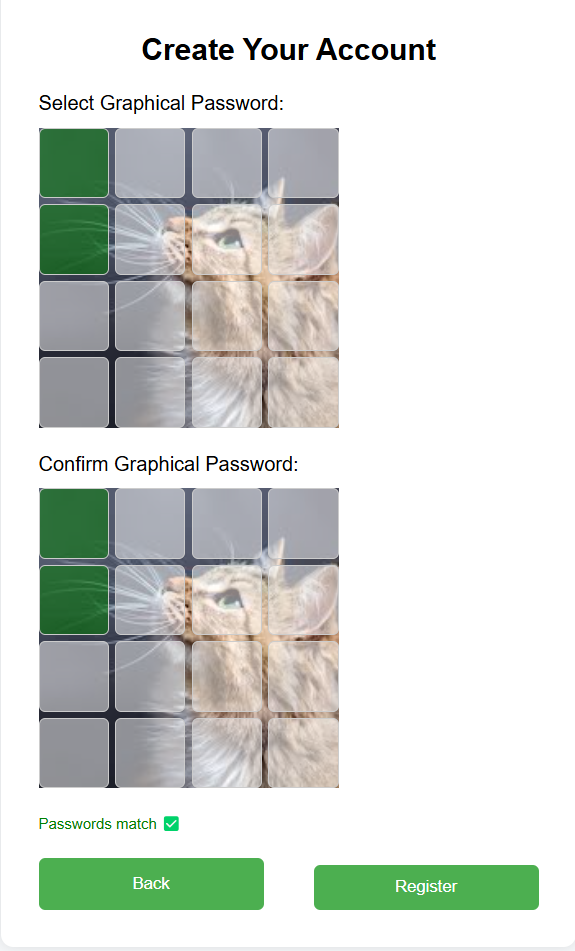
CREATE TABLE graphical\_pass (  
    id INT(11) NOT NULL AUTO\_INCREMENT,  
    userid VARCHAR(255) NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    selected\_image VARCHAR(255) NOT NULL,  
    password\_segments TEXT NOT NULL,  
    created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
    PRIMARY KEY (id)  
);

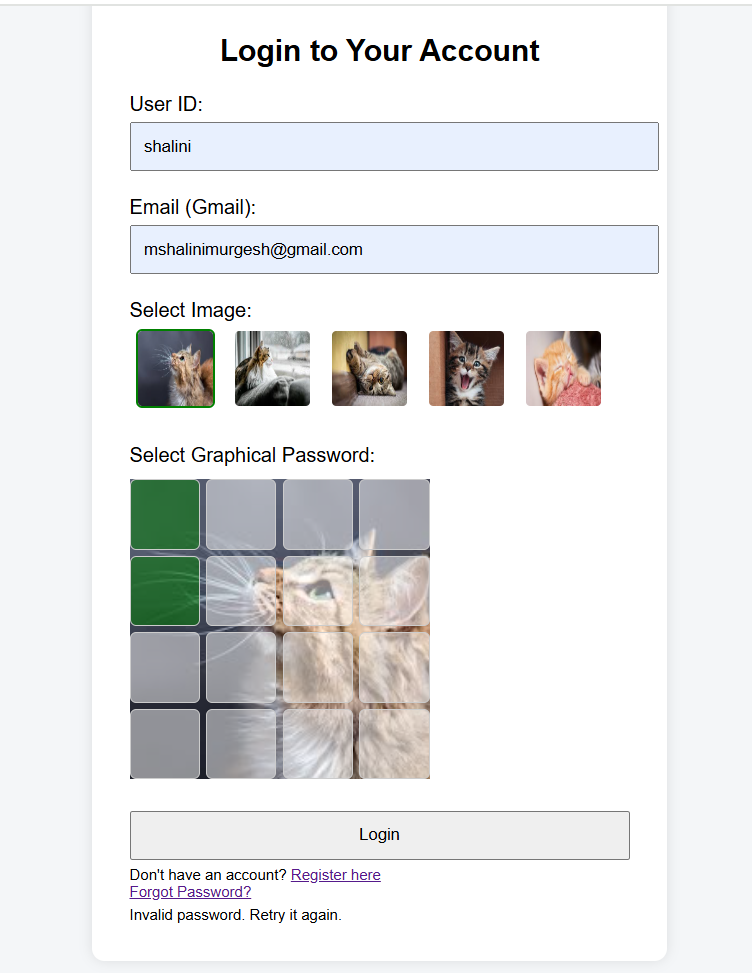
db\_connect.php

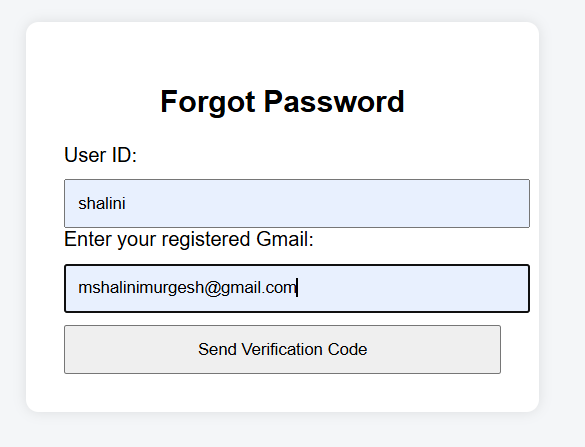
<?php  
// Database connection settings  
$servername = "localhost";    // if you're using XAMPP/LAMP, localhost is correct  
$username = "root";           // default username for XAMPP is 'root'  
$password = "";               // default password for XAMPP MySQL is empty  
$database = "graphical\_pass"; // your database name  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $database);  
  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
?>

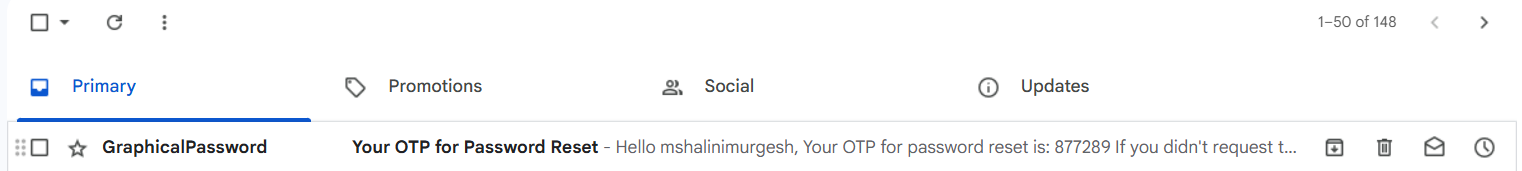


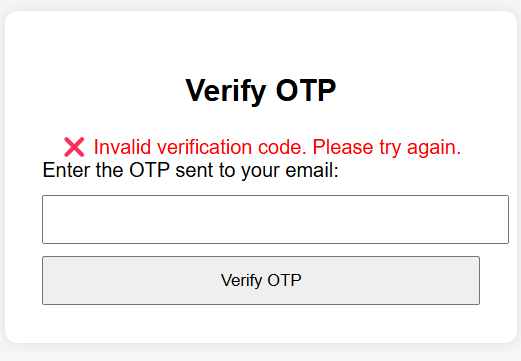


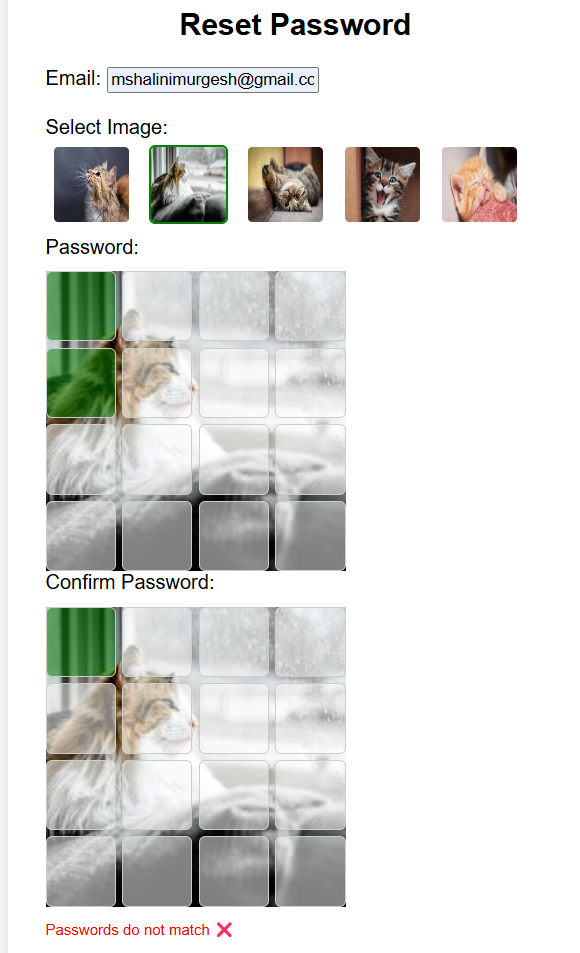




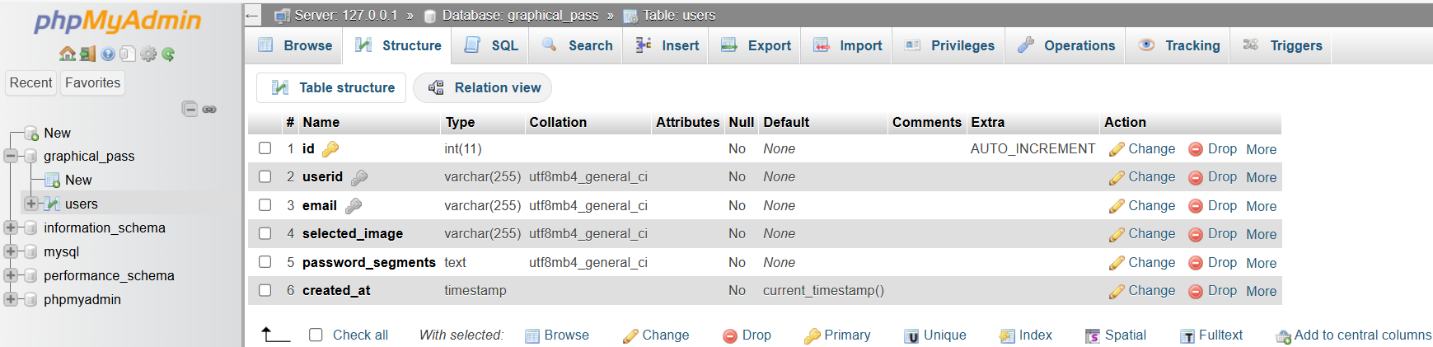


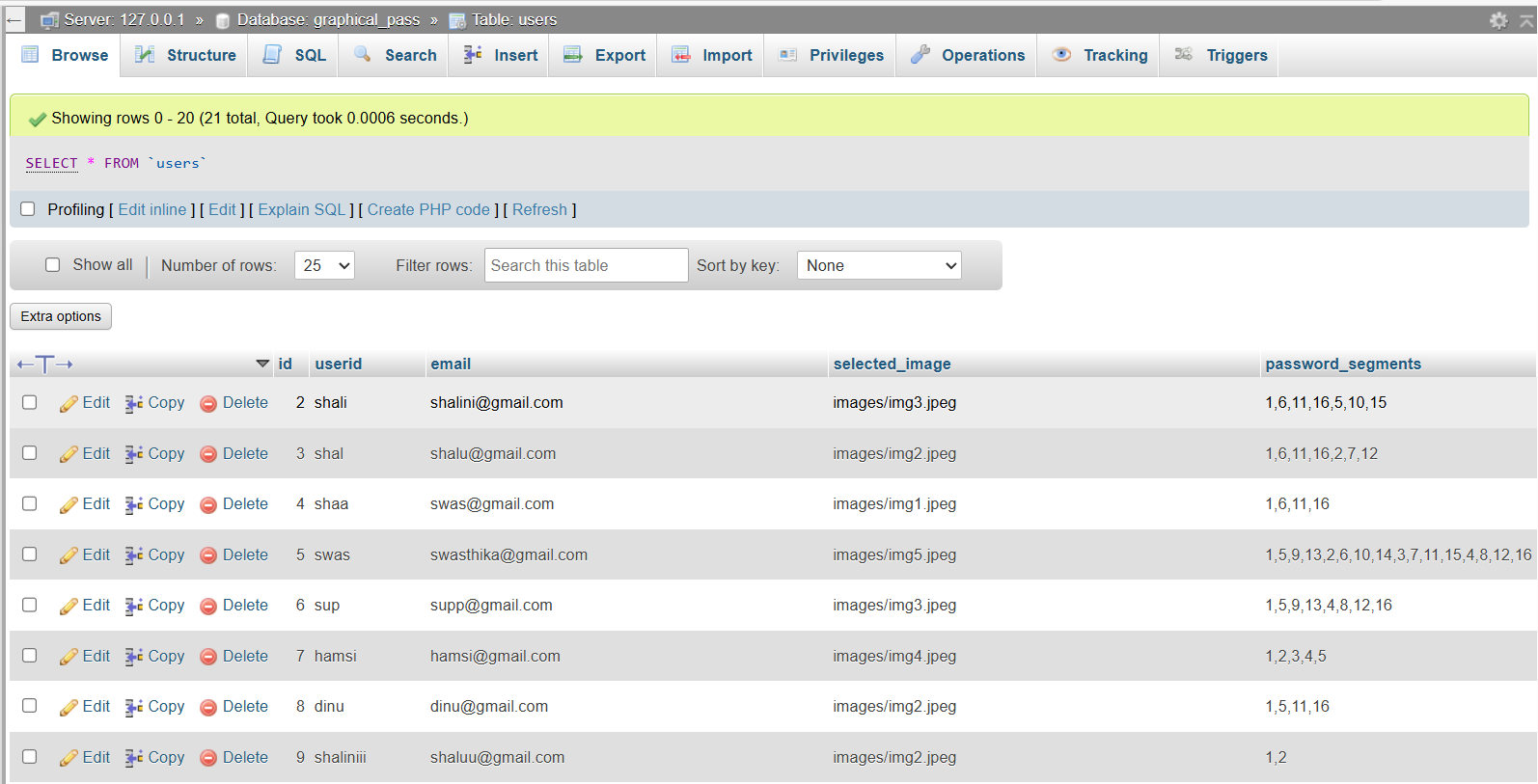


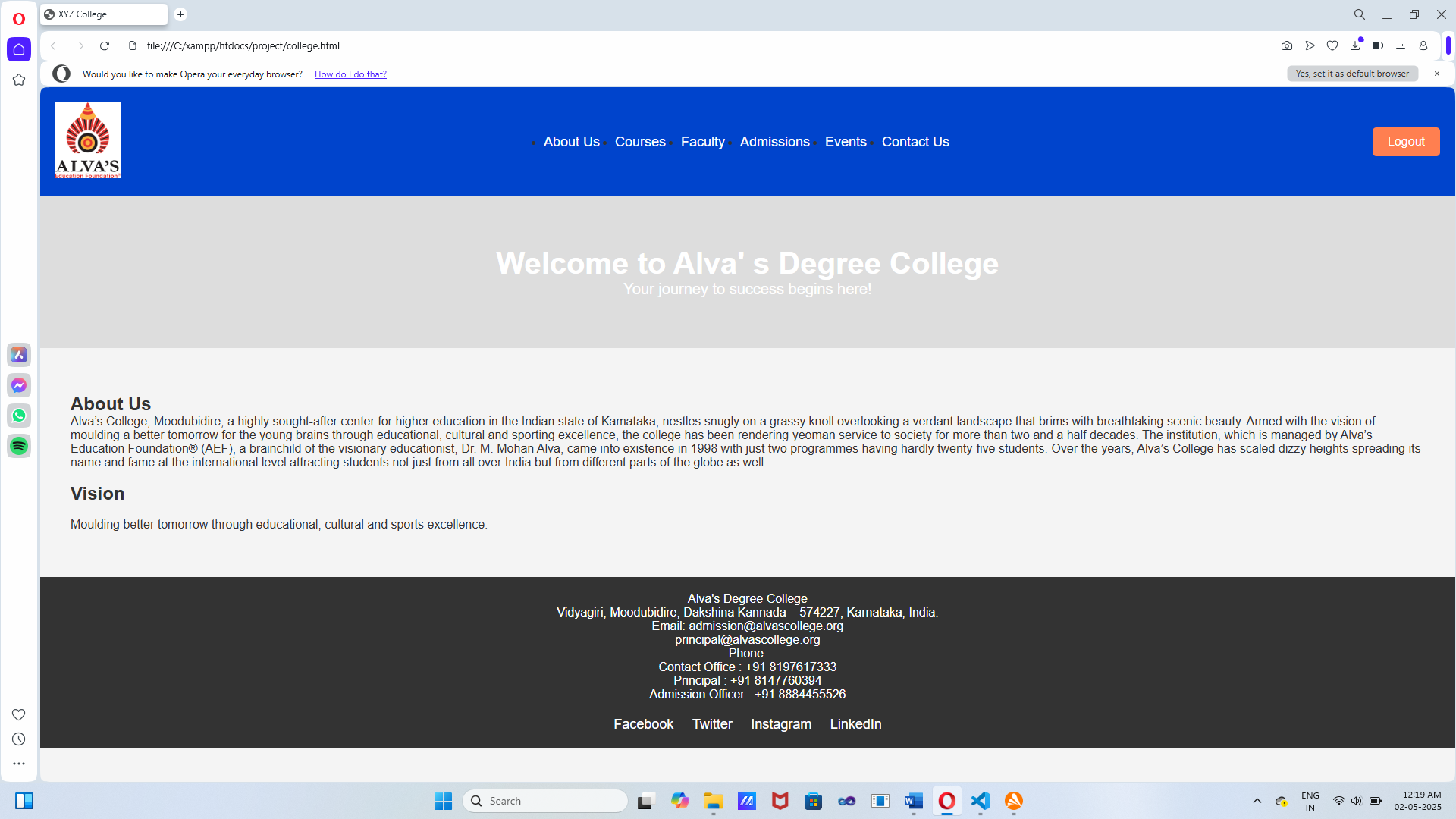


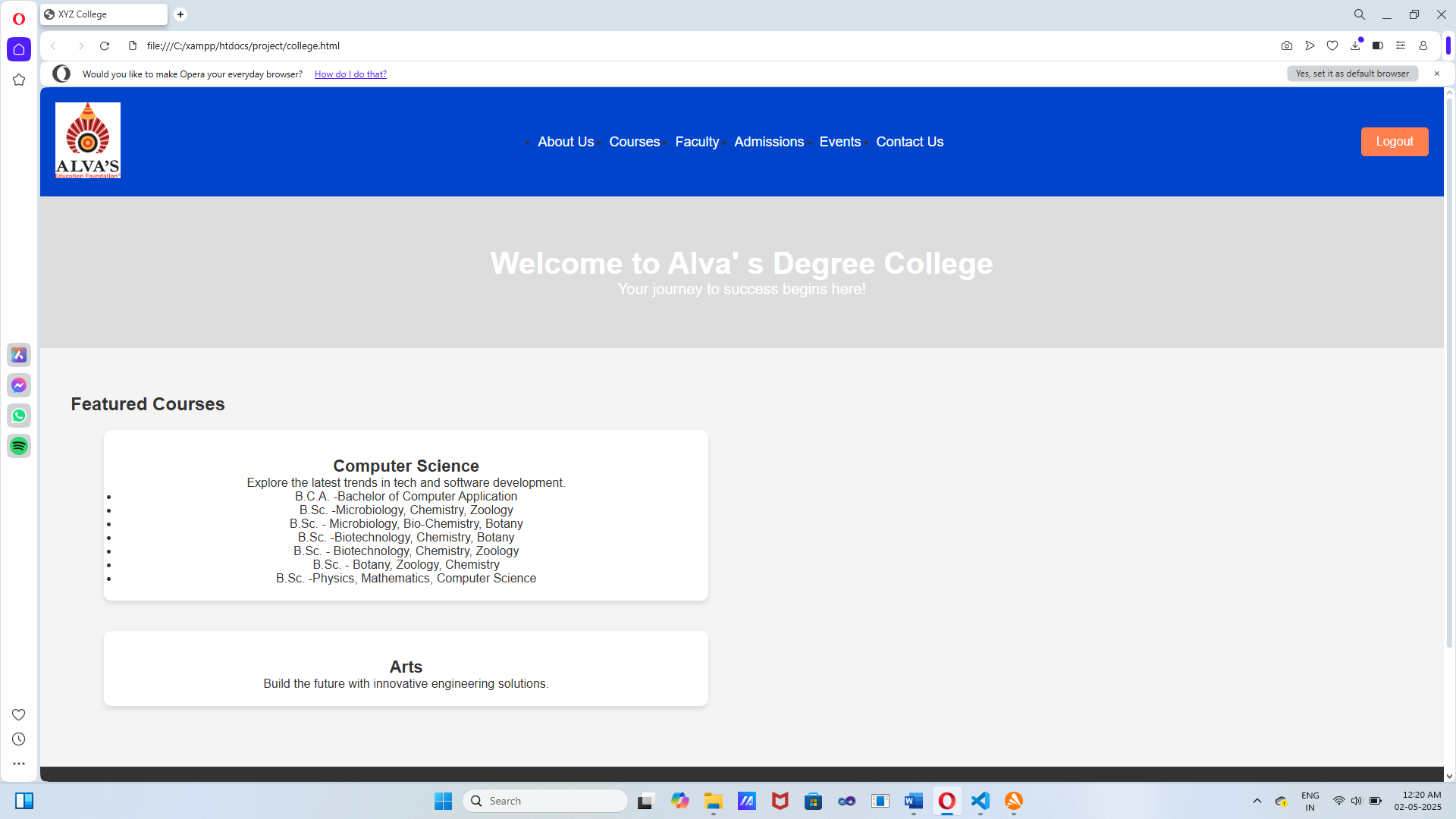


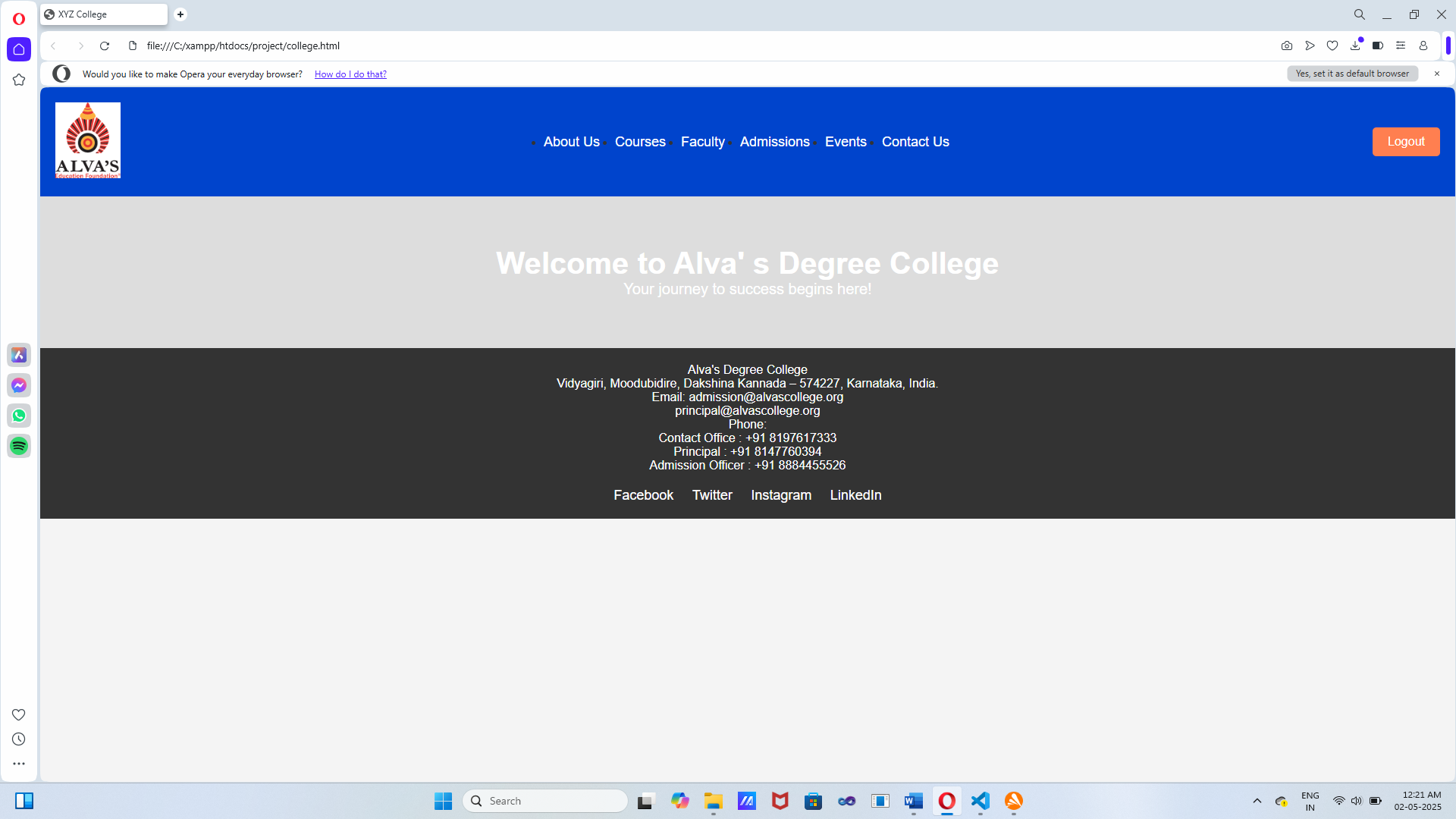


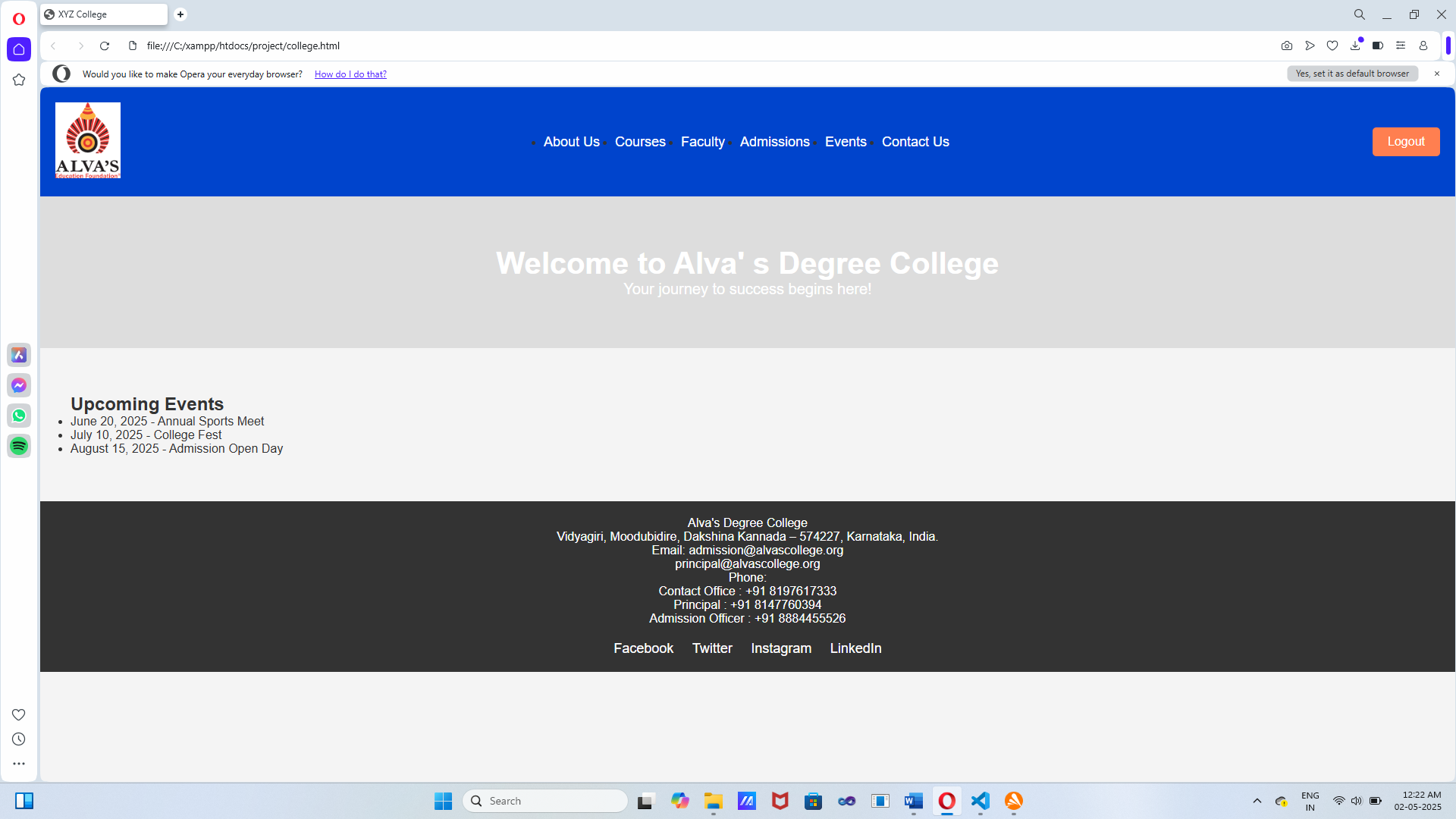












**7.1 Introduction:**

Testing is a process of running a system with the intension of finding errors. Testing enhance the integrity of the system by detecting the deviations in designs and errors in the system. Testing aims at errors \_prom areas. This helps in the prevention of errors in the system. Testing also adds values to the product by confirming to the user requirement. The main purpose of is to detect errors and get error-prom area in the system. Testing must be through and well-planned.

**7.2 Objectives of testing:**

The objective of testing are:

* Testing is a process of executing a program with intent of finding errors.
* A successful test case is one that uncovers an as yet undiscovered error.

**7.3Testing Methods:**

System testing is a stage of Implementation. This is to check whether the system works accurately and efficient before live operation commences. Testing is vital to the success of the system.

* **Unit Testing:**

Unit testing focuses effort on the smallest unit of software decision. This is known as module testing. The modules are tested separately.

* **Integration Testing:**

Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface.

* **User acceptance testing:**

User acceptance of the system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the perspective system users at the time of developing and making changes whenever required.

* **Validation:**

At the completion of the integration testing, software is completely assembled as a package interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing.

* **Output Testing:**

After performing validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format.

**Test Unit: registration form:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Input | Expected Output | Status |
| 1. Username Validation | Short or existing usernames | Error message, button disabled | Successful |
| 2. Gmail Validation | Non-Gmail or already-used | "Not allowed", button disabled | Successful |
| 3. Image Selection | Click on an image | Image gets green border, image stored | Successful |
| 4. Graphical Password Matching | Select same grid cells in both grids | "Passwords match ✅", enable submit | Successful |
| 5. Password Mismatch | Different grid selections | "Passwords do not match ❌" | Successful |
| 6. Submit Disabled | Incomplete or invalid form | Submit button remains disabled | Successful |

**Test Unit: Configuration form:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| "john123" | "❌ User ID is already taken." | Test with an existing username. | Successful |
| "newuser" | "✅ User ID is available." | Test with a unique username. | Successful |
| "us" | No output (empty response) | Test with a short username (should be invalid). | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| "[example@gmail.com](mailto:example@gmail.com)" | "❌ Gmail already registered." | Test with an existing Gmail address. | Successful |
| "[newuser@gmail.com](mailto:newuser@gmail.com)" | "✅ Gmail is allowed." | Test with a new, unique Gmail address. | Successful |
| "invalidemail.com" | No output (empty response) | Test with an incorrectly formatted email. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| "[existing@gmail.com](mailto:existing@gmail.com)" | {"valid": true} | Test with a registered email address. | Successful |
| "[nonexistent@gmail.com](mailto:nonexistent@gmail.com)" | {"valid": false} | Test with a non-registered email address. | Successful |
| "invalidemail.com" | {"valid": false, "error": "Missing email"} | Test with an invalid or missing email field. | Successful |

**Test Unit: Database form:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| Valid database credentials | Connection successful (no error) | Ensure correct database credentials are provided. | Successful |
| Invalid database credentials | Connection failed error message | Test with incorrect credentials (username, password) | Successful |
| Missing database server | Error: Connection failed | Test with an unreachable or non-existing database server. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| SHOW TABLES | Table users exists | Test if the users table exists. | Successful |
| DESCRIBE users; | Correct columns (userid, email, etc.) | Verify the table has the necessary fields. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| ' OR 1=1 -- | No error or access granted | Test to check for SQL injection vulnerability. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| <script>alert('XSS')</script> | No input error or script execution | Test if input sanitization prevents XSS attacks. | Successful |

**Test Unit: Login form:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| Empty Username and Email fields | Submit button remains disabled | Test if the form is disabled when no fields are filled. | Successful |
| Empty Graphical Password selected | Submit button remains disabled | Test if the form is disabled when no password grid is selected. | Successful |
| Username filled, Email filled, Image selected, Password selected | Submit button enabled | Test if the submit button is enabled when all fields are correctly filled. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| Click on image (e.g., img1.jpeg) | Grid background changes to selected image | Test if the grid background image updates when an image is selected. | Successful |
| Click on any grid cell | Grid cell is highlighted when selected | Test if clicking on a grid cell adds it to the password sequence. | Successful |
| Deselect grid cells | Grid cell is deselected and removed from password sequence | Test if deselecting a grid cell updates the password sequence. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| Select multiple grid cells | Selected cells highlight, sequence is updated | Test if selecting multiple cells updates the password sequence. | Successful |
| Deselect grid cells | Deselecting grid cells removes from sequence | Test if deselecting grid cells correctly removes them from the sequence. | Successful |
| Submit form with valid password sequence | Successful login with valid credentials | Test if a valid graphical password sequence logs the user in. | Successful |
| Submit form with invalid sequence | Error message displayed, retry option presented | Test if invalid password sequence triggers error handling. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| Correct Username, Gmail, Image, and Password | Redirect to college.html after successful login | Test if valid credentials submit and redirect to college.html. | Successful |
| Incorrect Username or Password | Error message displayed: "Invalid password. Retry it again." | Test if invalid credentials display the error message. | Successful |
| No Username or Password entered | Submit button remains disabled, no request sent | Test if submit button stays disabled when mandatory fields are empty. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| Empty username | Form cannot be submitted, warning message appears | Test if username input is validated before form submission. | Successful |
| Invalid Gmail address | Form cannot be submitted, warning message appears | Test if the email input is validated before form submission. | Successful |
| Valid username and Gmail address | Form can be submitted successfully | Test if the form submission works with valid inputs. | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Input | Expected Output | Description | Status |
| Correct Username and Password | Successful login, redirect to college.html | Test if the server-side PHP script correctly validates the user credentials. | Successful |
| Incorrect Username or Password | Error message: "Invalid password" | Test if invalid credentials are rejected and the correct error message is displayed. | Successful |

**Conclusion:**

As a practical and memorable authentication technique, the suggested Cued Click Points scheme has promise. GPIS is more usable than Pass Points because it makes use of users' capacity to recognise images and the memory trigger connected to seeing a new image. It seems simpler to be cued when each image is displayed and to only have to remember one click point per image as opposed to having to remember an ordered succession of clicks on a single image.

A safer substitute for Pass Points is GPIS. By requiring attackers to first obtain image sets for each user and then perform hotspot analysis on each of these photos, GPIS increases the effort for attackers. We can also include challenge response interactivity in future developments. In challenge-response interactions, the server will pose a challenge, and the client will then need to respond in line with the requirements. Access is allowed if the response is accurate. We can also restrict the number of incorrect password entries a user can make.