CS232 - Project Report

Secure Face Recognition Authentication System

Team Members:

Name	Registration Number	Contribution
M. Daniyal	2023406	Frontend Development,Database Integration
Muhammad Bin Waseem	2023403	Backend Development, OTP configuration
Mahad Aqeel	2023286	Documentation,Face Authentication

1. Project Overview

The Secure Face Recognition Authentication System (D.E.E.Z) is a web-based application for secure user authentication using email/password, OTP verification, and facial recognition. It uses PostgreSQL for data storage, PHP for backend logic, and HTML/CSS/JS/TS for the frontend. It integrates EmailJS for OTP delivery and BlazeFace (TensorFlow.js) for facial recognition, featuring a responsive UI with animations and scalable design.

2. Technology Stack

Directory Structure

Root: .bolt (with config.json), .gitignore, HTML (dashboard.html, face-setup.html, index.html, login.html, registration.html, verify-otp.html), SQL (database.sql), PHP (db.php, face-login.php, face.php, login.php, otp.php, register.php), configs (eslint.config.js, vite.config.ts, tailwind.config.js, tsconfig.json).

- CSS: auth.css, dashboard.css, face-scan.css, login.css, registration.css, styles.css, verification.css.
- JS: app.js, auth.js, dashboard.js, face-recognition.js, face-setup.js, login.js, otp.js, registration.js, ui.js, utils.js.

- Node Modules: Dependencies (TensorFlow.js, Tailwind CSS, Vite).

Key Components

Frontend

- HTML: Templates for dashboard, login, registration, face setup, OTP verification.
- CSS: Global styles with animations; page-specific styles for forms, face scanning, dashboard.
- JS/TS:
- app.js: Initializes app, session checks.
- auth.js: Manages sessions, face data.
- dashboard.js: Adds animations, logout.
- face-recognition.js: Implements BlazeFace.
- face-setup.js: Handles face ID setup.
- login.js: Manages login flows.
- otp.js: Processes OTP verification.
- registration.js: Validates registration, sends OTP.
- ui.js: Loads page templates.
- utils.js: Utility functions.

Backend:

- Database: Postgresql
- Users: Stores user info, email index.
- Face Authentication: Stores face descriptors.
- Secure Password: Password Storage with Salt and Pepper Hashing

PHP:

- db.php: Database connection.
- face-login.php, face.php: Face authentication.
- login.php: Email/password login.
- otp.php: OTP verification
- register.php: User registration.

Dependencies

- Node Modules: TensorFlow.js (BlazeFace), Tailwind CSS, Vite, ESLint, EmailJS.
- EmailJS: Sends OTPs (service_jdtk3x4 for Gmail, service_xt65nbs for GIKI).
- BlazeFace: Facial recognition.
- Cloudflare: Security scripts.
- JSON Usage

Configuration: package.json, package-lock.json, tsconfig.json (and variants), bolt/config.json define dependencies, scripts, TypeScript settings.
Data Exchange: Frontend (fetch) and backend (PHP) use JSON for API responses (e.g., { success: true }).
Local Storage: JSON data stored in localStorage (e.g., pendingRegistration, currentOTP).
EmailJS: Sends JSON payloads for OTPs (e.g., { email: "user@gmail.com", passcode: "123456" }).

3. Entity-Relationship Diagram (ERD)

Entities:

User,Email,Password,Gmail,Outlook,Face,Session Logs,OTP(gmail service,outlook service)

Relationships:

1. **User → OTP:**

   A user can generate multiple OTPs, each linked to them via user ID.

2. **User → Email:**

   Each user has one unique email used for login and communication.

3. **User → Password:**

   A user is associated with one securely hashed password.

4. **User → Face:**

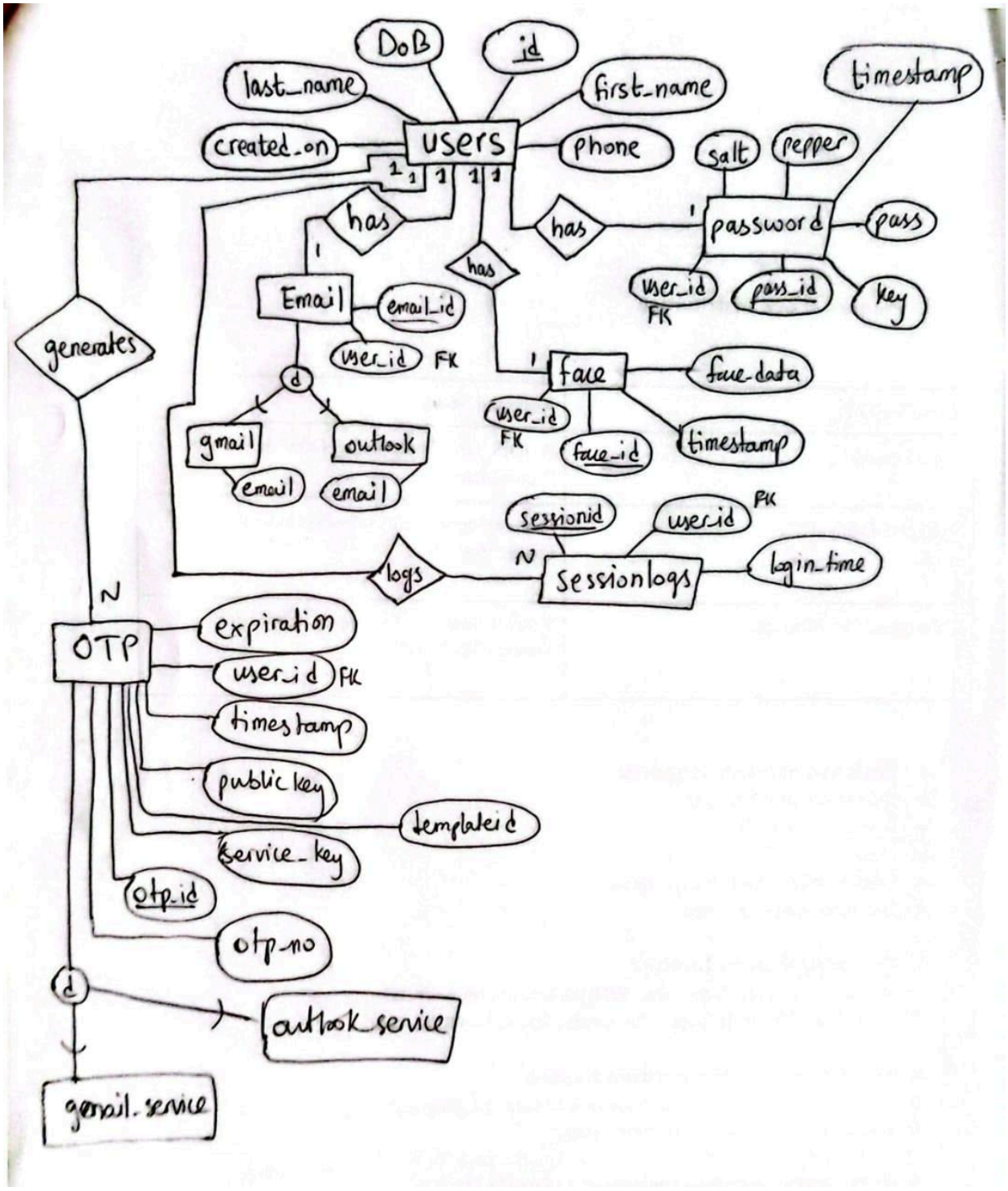
   Each user has one stored facial descriptor used for biometric login.

5. **User → Session Logs:**

   Every user may have multiple session logs recording login activity.

6. **OTP → Email Service:**

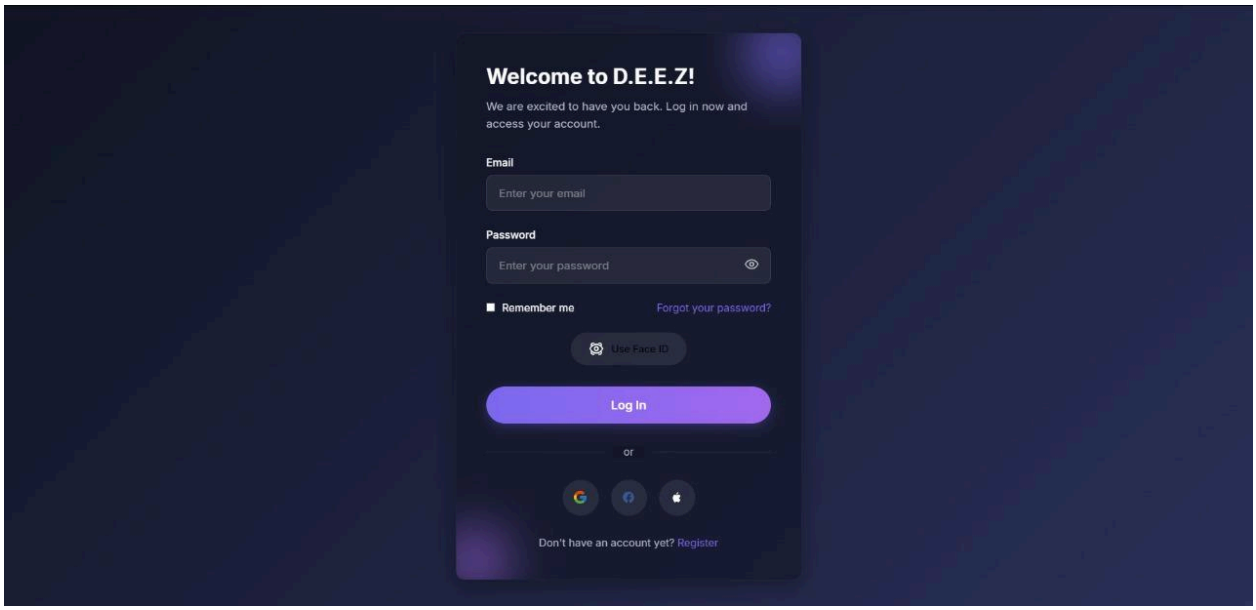
   Each OTP is sent through either Gmail or Outlook, identified by a service type.

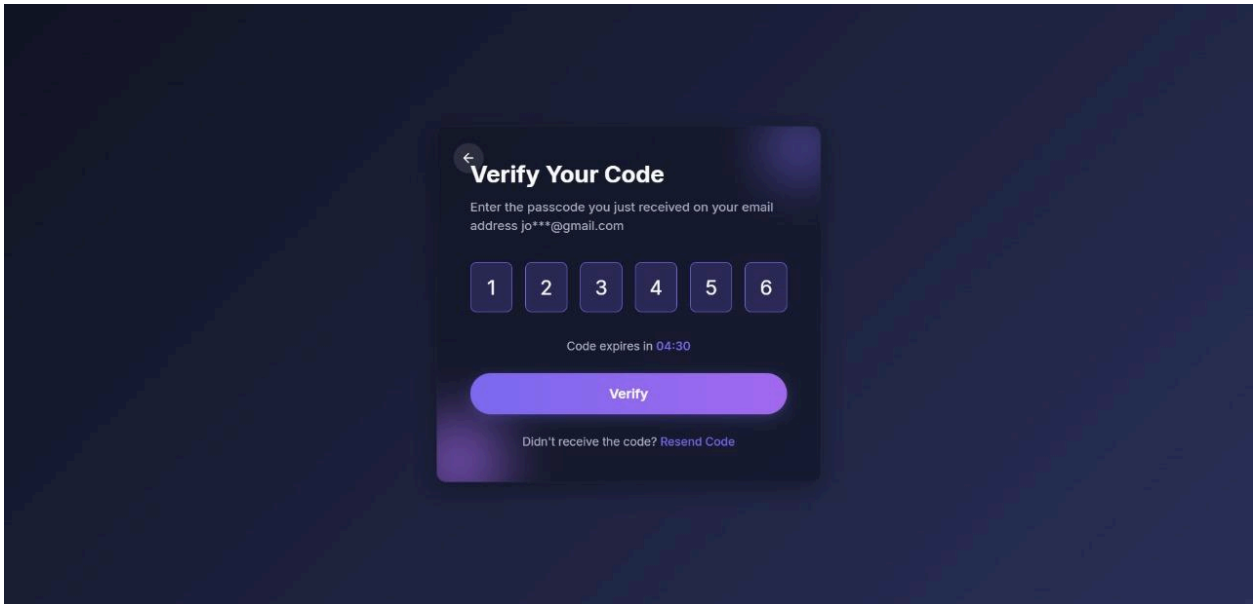

DoB
id
last_name
first_name
timestamp
created_on
users
phone
salt
pepper
has
has
password
pass
has
Email
email_id
user_id
FK
user_id
FK
pass_id
key
generates
d
face
face_data
gmail
outlook
user_id
FK
face_id
timestamp
email
email
sessionid
user_id
FK
N
logs
sessionlogs
login_time
N
OTP
expiration
user_id
FK
timestamp
public key
template
service_key
otp_id
otp_no
d
outlook_service
gmail_service

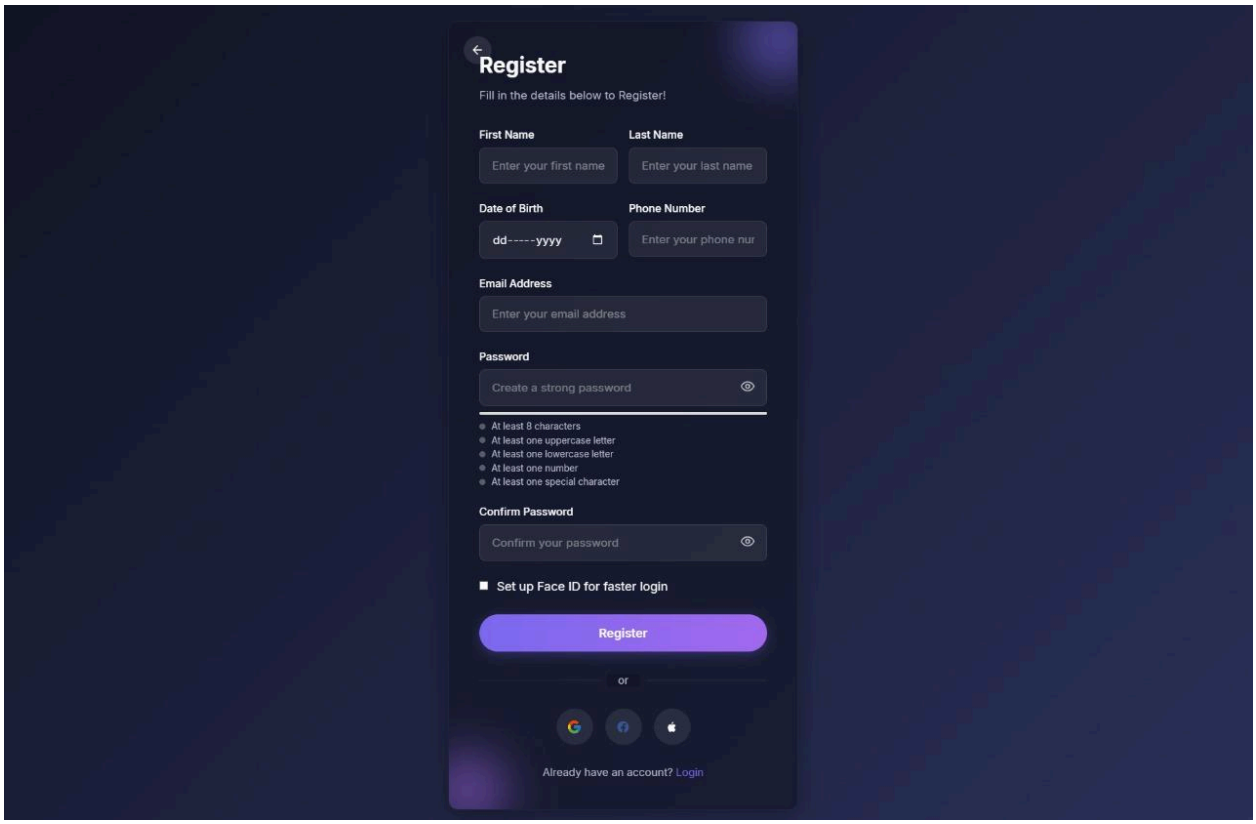
4. Database Schema

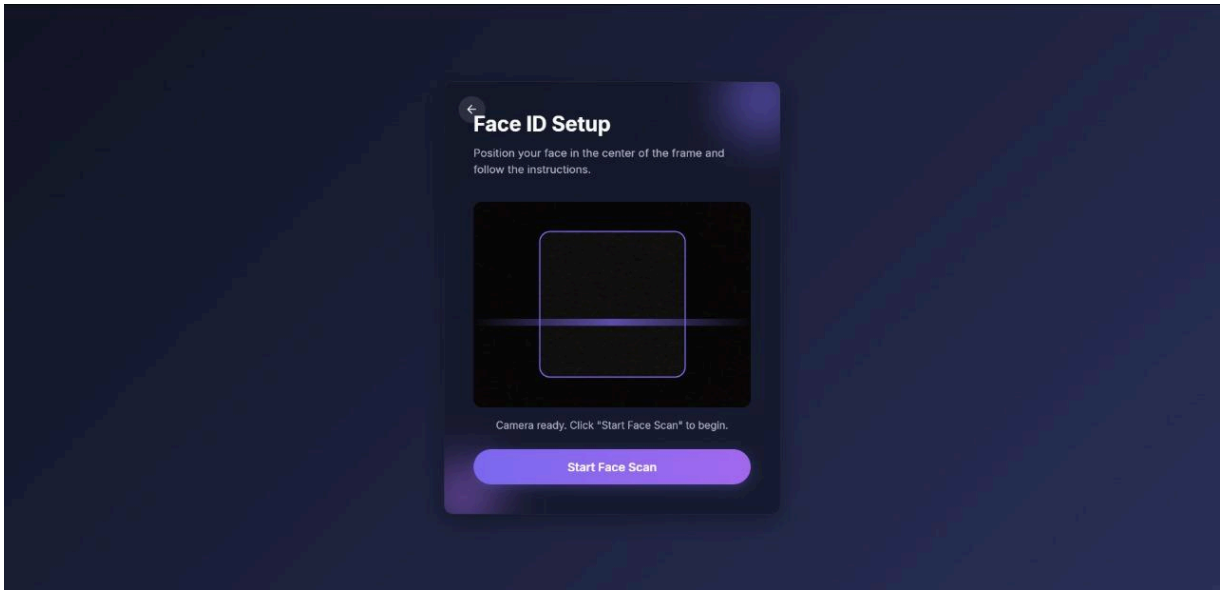
```sql
CREATE TABLE IF NOT EXISTS users (
    id SERIAL PRIMARY KEY,
    firstname VARCHAR(50) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    dob DATE NOT NULL,
    phone VARCHAR(20) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
CREATE INDEX IF NOT EXISTS idx_users_email ON users(email);
select * from users;
--  Face Authentication Table (Stores facial recognition data)
CREATE TABLE face_auth (
    id INT PRIMARY KEY REFERENCES users(id) ON DELETE CASCADE,
    face_data text NOT NULL, -- Storing face data
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
select * from face_auth;
--for storing passwords
CREATE TABLE passwords (
    pass_id BIGSERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL,
    salt VARCHAR(64) NOT NULL,
    pepper VARCHAR(64) NOT NULL,
    key VARCHAR(255) NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
select * from passwords;
```

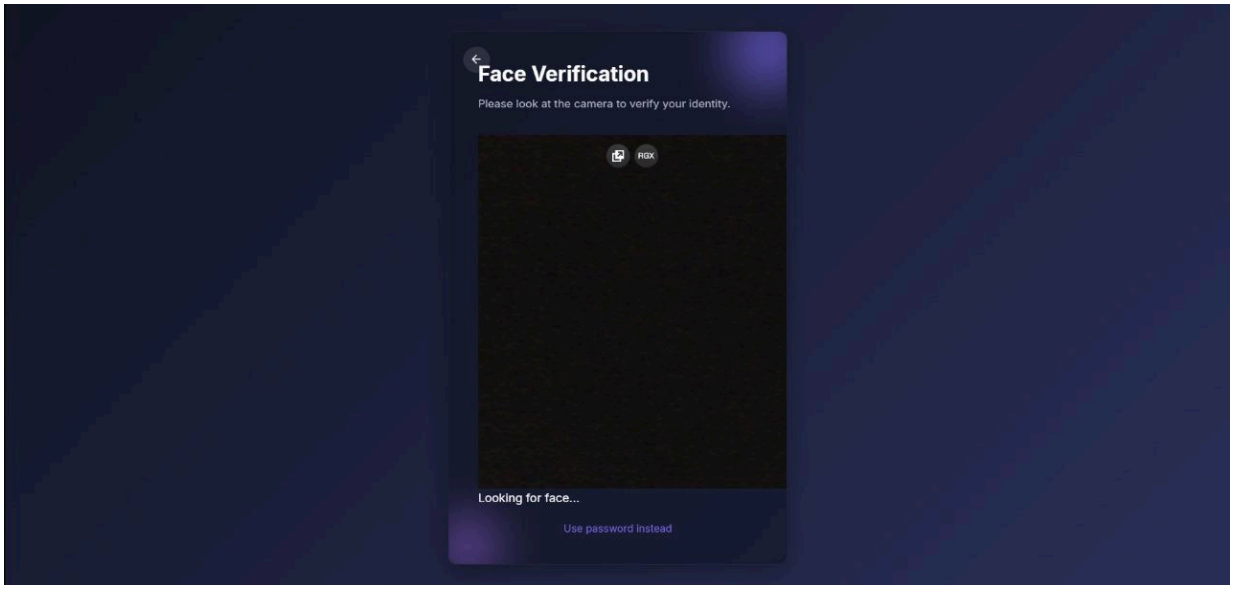
5. Screenshots of Working System

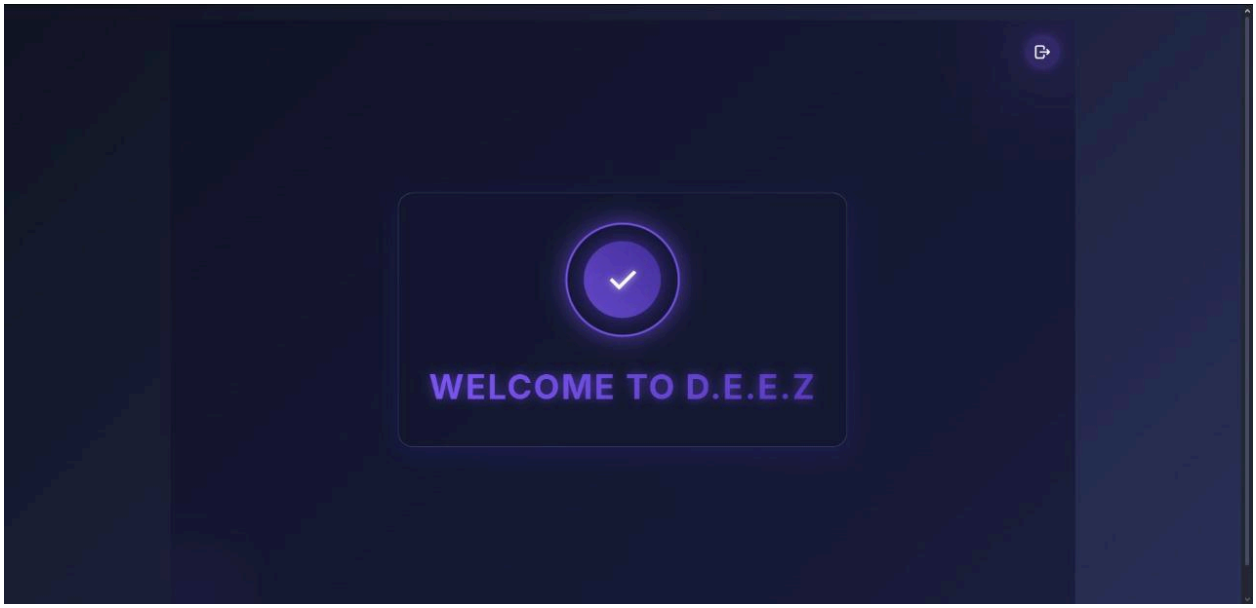



Verify Your Code
Enter the passcode you just received on your email address jo***@gmail.com
1
2
3
4
5
6
Code expires in 04:30
Verify
Didn't receive the code? Resend Code


Register
Fill in the details below to Register!
First Name
Enter your first name
Last Name
Enter your last name
Date of Birth
dd----yyyy
Phone Number
Enter your phone nur
Email Address
Enter your email address
Password
Create a strong password
At least 8 characters
At least one uppercase letter
At least one lowercase letter
At least one number
At least one special character
Confirm Password
Confirm your password
Set up Face ID for faster login
Register
or
Already have an account? Login


Face ID Setup
Position your face in the center of the frame and follow the instructions.
Camera ready. Click "Start Face Scan" to begin.
Start Face Scan


Face Verification
Please look at the camera to verify your identity.
RGX
Looking for face...
Use password instead



6. Functionalities Implemented

Features

- Registration: Validates inputs, sends OTP, offers face ID setup.
- OTP Verification: 6-digit code, 15-minute expiry timer, resend.
- Login: Email/password, face ID options.
- Face Recognition: Detects/stores face descriptors.
- Dashboard: Animated welcome, logout.
- Security: Password checks, Facial Recognition.
- Scalability: Partitioned tables, indexes.

Technical Details

- Frontend: HTML5, CSS3, ES6+, TypeScript, fetch, Tailwind CSS.
- Backend: PHP, PostgreSQL, JSON APIs.
- Build Tools: Vite, PostCSS, ESLint, Tailwind CSS, TypeScript.

Strengths

- Modern UI with animations.
- Robust authentication.
- Scalable database.
- Modular code.
- Advanced tooling.

7. GitHub Details

GitHub Repository:
https://github.com/MuhammadBinWaseemm/Secure_Face_Recognition_Auth_System

8. Challenges Faced

- Managing foreign key constraints during insertions
- Merging frontend-backend API responses
- Getting Facial Recognition Accuracy
- Storage of Facial Data

9. Future Enhancements

- Enhance face recognition accuracy.
- Add HTTPS, CSRF protection, encryption.
- Improve error handling, testing.

10. Conclusion

D.E.E.Z demonstrates a complete, secure, and scalable approach to modern user authentication by combining traditional credentials with biometrics.
The use of JSON for configuration, data exchange, and session management ensures modularity and flexibility across all components.
With minor enhancements, this system is ready for real-world deployment.
