**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ**

**ТЕХНОЛОГІЙ І ЗВ’ЯЗКУ**

**Звіт**

**з дисципліни Веб-технології та веб-дизайн**

**Лабораторна робота №12**

**на тему: «Підключення БД»**

Виконав: студент 3 курсу, групи ІПЗ-3.04 спеціальності

121 Інженерія програмного забезпечення

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Бухта М.М.

Перевірив\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Рябов Д.М.

**Одеса  2024**

**ЗАВДАННЯ**

Подключить базу данных к запрограммированному ранее интерфейсу

**ВИКОНАННЯ**

**main.cpp**

#include "config.hpp"

#include "DBQuery.hpp"

#include "Logger.hpp"

#include "ServerStarterController.hpp"

#include "ContextHandlerController.hpp"

#include "ServerStarterModel.hpp"

#include "ContextHandlerModel.hpp"

#include "ContextHandlerInterface.hpp"

#include "ClientHandlerInterface.hpp"

#include <cinttypes>

#include <chrono>

#include <ctime>

#include <iomanip>

#include <sstream>

#include <thread>

#include <memory>

#include <queue>

#include <sys/stat.h> // mkdir;

namespace db = server::db;

std::string current\_date\_time\_as\_string() {

// Get the current time using std::chrono

auto now = std::chrono::system\_clock::now();

// Convert the current time to a time\_t object

std::time\_t now\_c = std::chrono::system\_clock::to\_time\_t(now);

// Convert time\_t to a struct tm (broken down time)

std::tm tm\_struct = \*std::localtime(&now\_c);

// Format the date-time as a string

std::stringstream ss;

ss << std::put\_time(&tm\_struct, "%Y.%m.%d\_%H.%M.%S");

return ss.str();

}

std::string generate\_log\_file\_name(const std::string &first\_name\_part) {

std::stringstream file\_name\_stream;

file\_name\_stream << first\_name\_part << "\_" << current\_date\_time\_as\_string() << ".log";

return std::move(file\_name\_stream.str());

}

int main(int argc, char \*\*argv) {

const std::string LOGS\_FOLDER\_DIR\_NAME = "logs";

mkdir(LOGS\_FOLDER\_DIR\_NAME.c\_str(), 0755);

BLOG\_INIT(std::move(generate\_log\_file\_name(LOGS\_FOLDER\_DIR\_NAME + "/server\_logs.txt")), true);

BDECLARE\_TAG\_SCOPE("", \_\_FUNCTION\_\_);

std::queue<std::thread> thread\_pull;

auto context\_handler\_interface = std::make\_shared<server::context\_handler::view::ContextHandlerInterface>();

auto context\_handler\_model = std::make\_shared<server::context\_handler::models::ContextHandlerModel>();

auto context\_handler\_controller = std::make\_shared<server::context\_handler::controllers::ContextHandlerController>(context\_handler\_model, context\_handler\_interface);

auto server\_starter\_model = std::make\_shared<server::serverstarter::models::ServerStarterModel>();

auto client\_handler\_interface = std::make\_shared<server::client\_handler::view::ClientHandlerInterface>();

server::serverstarter::controllers::ServerStarterController server\_starter\_controller(server\_starter\_model,

context\_handler\_interface, client\_handler\_interface

);

thread\_pull.push(std::thread(&server::serverstarter::controllers::ServerStarterController::start, &server\_starter\_controller));

// dbquery.output\_all\_users();

while (!thread\_pull.empty()) {

if (thread\_pull.front().joinable()) {

thread\_pull.front().join();

}

thread\_pull.pop();

}

return 0;

}

**DBQuery.hpp**

#ifndef POLYCLINIC\_SERVER\_DB\_DBQUERY\_HPP

#define POLYCLINIC\_SERVER\_DB\_DBQUERY\_HPP

#include <pqxx/connection>

#include <pqxx/transaction>

#include <memory>

namespace server::db {

class DBQuery {

public:

DBQuery(void);

DBQuery(const DBQuery &other) = delete;

~DBQuery(void);

void operator=(const DBQuery &other) = delete;

void output\_all\_users(void);

bool check\_signin\_is\_valid(std::string email, std::string password);

private:

std::unique\_ptr<pqxx::connection> m\_db\_connection;

};

} // !server::db;

#endif // !POLYCLINIC\_SERVER\_DB\_DBQUERY\_HPP;

**DBQuery.cpp**

#include "DBQuery.hpp"

#include "config.hpp"

#include "Logger.hpp"

#include <pqxx/result>

#include <sstream>

#include <thread>

namespace config = server::common::config;

namespace server::db

{

DBQuery::DBQuery() {

BDECLARE\_TAG\_SCOPE("DBQuery", \_\_FUNCTION\_\_);

std::stringstream db\_connection\_command\_stream;

db\_connection\_command\_stream << "host=" << config::DB\_HOST << " port=" << config::DB\_PORT

<< " user=" << config::DB\_USERNAME << " password=" << config::DB\_PASSWORD

<< " dbname=" << config::DB\_NAME;

BLOG\_INFO("constructor called on thread #", std::this\_thread::get\_id());

BLOG\_INFO("Make connection to db. ", db\_connection\_command\_stream.str());

m\_db\_connection = std::make\_unique<pqxx::connection>(db\_connection\_command\_stream.str());

if (m\_db\_connection->is\_open()) {

BLOG\_DEBUG("Connection success");

} else {

BLOG\_FATAL("Connection failed");

abort();

}

};

DBQuery::~DBQuery(void) {

m\_db\_connection->disconnect();

}

void DBQuery::output\_all\_users(void) {

BDECLARE\_TAG\_SCOPE("DBQuery", \_\_FUNCTION\_\_);

BLOG\_INFO("called");

auto db\_transaction = std::make\_unique<pqxx::work>(\*m\_db\_connection);

pqxx::result res = db\_transaction->exec("SELECT \* FROM users");

db\_transaction->commit();

for (const auto &row : res) {

std::stringstream table\_info\_output;

for (const auto &field : row) {

table\_info\_output << field.c\_str() << '\t';

}

BLOG\_DEBUG(table\_info\_output.str());

}

}

bool DBQuery::check\_signin\_is\_valid(std::string email, std::string password) {

BDECLARE\_TAG\_SCOPE("DBQuery", \_\_FUNCTION\_\_);

std::stringstream db\_request;

auto db\_transaction = std::make\_unique<pqxx::work>(\*m\_db\_connection);

db\_request << "SELECT \* FROM check\_user\_login('"

<< email << "', '" << password << "')";

BLOG\_INFO("Current request: ", db\_request.str());

pqxx::result res = db\_transaction->exec(db\_request.str());

db\_transaction->commit();

return res.size() > 0 && res[0][0].as<bool>();

}

} // !server::db;

**config.hpp**

#ifndef POLYCLINIC\_SERVER\_COMMON\_CONFIG\_HPP

#define POLYCLINIC\_SERVER\_COMMON\_CONFIG\_HPP

#include <string>

#include <cinttypes>

namespace server::common::config {

const std::string LOCALHOST {"127.0.0.1"};

const std::string DB\_HOST {LOCALHOST};

const std::string DB\_USERNAME {"mbukhta"};

const std::string DB\_PASSWORD {"1qa@WS3ed"};

const std::string DB\_NAME {"polyclinic"};

const std::uint16\_t DB\_PORT {5432};

const std::uint16\_t SERVER\_PORT {8081};

const std::uint8\_t SERVER\_LISTEN\_SIZE = 10;

const char STR\_EOF = '\0';

const char EXTENSTION\_SEPARATOR = '.';

const std::uint16\_t BUFFER\_SIZE = 2048;

const std::string DEFAULT\_HTML\_FILE = "index.html";

} // !server::common::config;

#endif // !POLYCLINIC\_SERVER\_COMMON\_CONFIG\_HPP;

**ClientHandlerController.hpp**

#ifndef POLYCLINIC\_SERVER\_SERVER\_CLIENTHANDLER\_CONTROLLERS\_CLIENTHANDLERCONTROLLER\_HPP

#define POLYCLINIC\_SERVER\_SERVER\_CLIENTHANDLER\_CONTROLLERS\_CLIENTHANDLERCONTROLLER\_HPP

#include "IClientHandlerModel.hpp"

#include "IServerStarterModel.hpp"

#include "IContextHandlerInterface.hpp"

#include "IClientHandlerInterface.hpp"

#include "DBQuery.hpp"

#include "HttpHeaders.hpp"

#include "Signal"

#include <set>

#include <memory>

namespace server::client\_handler::controllers {

class ClientHandlerController {

public:

ClientHandlerController(std::weak\_ptr<serverstarter::models::IServerStarterModel> server\_model,

std::weak\_ptr<view::IClientHandlerInterface> client\_handler\_interface,

std::weak\_ptr<context\_handler::view::IContextHandlerInterface> context\_handler\_interface

);

~ClientHandlerController(void);

void start(void);

/\*

\* Slots starts;

\*/

public:

common::Slot<std::string, common::Socket> on\_page\_updated\_slot;

void on\_page\_updated(std::string address, common::Socket socket);

/\*

\* Slots ends;

\*/

private:

void init();

void catch\_new\_connection(void);

void disconnect(std::weak\_ptr<models::IClientHandlerModel> weak\_client);

void read\_data(std::weak\_ptr<models::IClientHandlerModel> weak\_client);

void send\_data(std::weak\_ptr<models::IClientHandlerModel> weak\_client);

void handle\_connect(const common::Socket& client\_socket);

void handle\_read(std::weak\_ptr<models::IClientHandlerModel> weak\_client, std::string&& read\_data, std::int32\_t bytes\_read);

void handle\_http\_request(std::weak\_ptr<models::IClientHandlerModel> weak\_client, const common::HttpHeaders &header);

void handle\_json\_post(common::Socket socket, std::string uri, std::string json\_pkg);

private:

std::shared\_ptr<context\_handler::view::IContextHandlerInterface> m\_context\_handler\_interface;

std::shared\_ptr<view::IClientHandlerInterface> m\_client\_handler\_interface;

std::shared\_ptr<db::DBQuery> m\_db\_query;

std::shared\_ptr<const serverstarter::models::IServerStarterModel> m\_SERVER\_STARTER\_MODEL;

std::set<std::shared\_ptr<models::IClientHandlerModel>> m\_client\_handler\_model\_container;

};

} // !server::client\_handler::controllers;

#endif // !POLYCLINIC\_SERVER\_SERVER\_CLIENTHANDLER\_CONTROLLERS\_CLIENTHANDLERCONTROLLER\_HPP;

**ClientHanderController.cpp**

#include "ClientHandlerController.hpp"

#include "ClientHandlerModel.hpp"

#include "config.hpp"

#include "EnumStringConvertor.hpp"

#include "HttpHeaders.hpp"

#include "Socket.hpp"

#include "Logger.hpp"

#include <json/json.h>

#include <thread>

#include <unordered\_map>

#include <sys/stat.h>

#include <unistd.h>

#include <sys/sendfile.h>

namespace server::client\_handler::controllers {

ClientHandlerController::ClientHandlerController(std::weak\_ptr<serverstarter::models::IServerStarterModel> server\_model,

std::weak\_ptr<view::IClientHandlerInterface> client\_handler\_interface,

std::weak\_ptr<context\_handler::view::IContextHandlerInterface> context\_handler\_interface

) : m\_SERVER\_STARTER\_MODEL{server\_model}, m\_client\_handler\_interface{client\_handler\_interface.lock()}

, m\_context\_handler\_interface{context\_handler\_interface.lock()}

, on\_page\_updated\_slot{std::bind(&ClientHandlerController::on\_page\_updated, this, std::placeholders::\_1, std::placeholders::\_2 )}

{

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

BLOG\_INFO("constructor called on thread #", std::this\_thread::get\_id());

m\_db\_query = std::make\_shared<db::DBQuery>();

}

ClientHandlerController::~ClientHandlerController(void) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

BLOG\_INFO("destructor called on thread #", std::this\_thread::get\_id());

for (auto &client : m\_client\_handler\_model\_container) {

client->socket().close();

if (!client->socket().is\_valid()) {

BLOG\_DEBUG("client = ", client->socket().to\_string(), " is closed");

} else {

BLOG\_WARNING("Impossible to close the client = ", client->socket().to\_string(), ". ", client->socket().latest\_error());

}

}

}

void ClientHandlerController::start(void) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

BLOG\_INFO("called");

init();

catch\_new\_connection();

}

void ClientHandlerController::init() {

common::connect(&m\_context\_handler\_interface->page\_updated, &on\_page\_updated\_slot);

}

void ClientHandlerController::on\_page\_updated(std::string address, common::Socket socket) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

BLOG\_INFO("Page address = ", address);

}

void ClientHandlerController::catch\_new\_connection(void) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

BLOG\_INFO("called");

auto server\_socket = m\_SERVER\_STARTER\_MODEL->socket();

while (server\_socket.is\_valid()) {

BLOG\_DEBUG("waiting for connection");

auto client\_socket = server\_socket.accept();

if (!client\_socket.is\_valid()) {

BLOG\_ERROR("impossible to connect to client: ", client\_socket.latest\_error());

continue;

}

handle\_connect(client\_socket);

}

}

void ClientHandlerController::disconnect(std::weak\_ptr<models::IClientHandlerModel> weak\_client) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

auto client = weak\_client.lock();

BLOG\_INFO("client: ", client->socket().to\_string());

m\_client\_handler\_interface->on\_client\_disconnected.emit(client->socket());

m\_client\_handler\_model\_container.erase(client);

if (client->socket().close()) {

BLOG\_ERROR("Cannot to close the client!");

}

}

void ClientHandlerController::read\_data(std::weak\_ptr<models::IClientHandlerModel> weak\_client) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

auto client = weak\_client.lock();

BLOG\_INFO("client: ", client->socket().to\_string());

while (client->socket().is\_valid()) {

BLOG\_DEBUG("Reading client: ", client->socket().to\_string());

std::string read\_buf;

auto bytes\_read = client->socket().read(read\_buf, common::config::BUFFER\_SIZE);

handle\_read(client, std::move(read\_buf), bytes\_read);

}

}

void ClientHandlerController::send\_data(std::weak\_ptr<models::IClientHandlerModel> weak\_client) {

}

void ClientHandlerController::handle\_connect(const common::Socket& client\_socket) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

auto client = std::make\_shared<models::ClientHandlerModel>();

client->set\_socket(std::move(client\_socket));

BLOG\_INFO("client: ", client->socket().to\_string());

m\_client\_handler\_model\_container.insert(client);

std::thread(&ClientHandlerController::read\_data, this, client).detach();

}

void ClientHandlerController::handle\_read(std::weak\_ptr<models::IClientHandlerModel> weak\_client, std::string&& read\_data, std::int32\_t bytes\_read) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

auto client = weak\_client.lock();

BLOG\_INFO("client: ", client->socket().to\_string(), "bytes\_read = ", bytes\_read, "; data = ", read\_data);

if (bytes\_read > 0) {

common::HttpHeaders http\_headers(read\_data);

handle\_http\_request(client, http\_headers);

} else if (bytes\_read == 0) {

BLOG\_INFO("Disconnect client", client->socket().to\_string());

disconnect(client);

// TODO: create signal-slot for disconnect client;

} else {

BLOG\_ERROR("Error in reading data: ", client->socket().latest\_error());

}

}

void ClientHandlerController::handle\_http\_request(std::weak\_ptr<models::IClientHandlerModel> weak\_client, const common::HttpHeaders &header) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

auto client = weak\_client.lock();

BLOG\_INFO("client: ", client->socket().to\_string(), "; http request type = ", common::EnumStringConvertor::init()->to\_string(header.method), "; http body = ", header.body);

switch (header.method) {

case common::HttpHeaders::HttpRequestType::GET:

m\_context\_handler\_interface->request\_to\_open\_uri(header.uri, client->socket());

break;

case common::HttpHeaders::HttpRequestType::POST:

BLOG\_WARNING("Tmp logic for the POST request");

std::thread(&ClientHandlerController::handle\_json\_post, this, client->socket(), header.uri, header.body).detach();

break;

default:

BLOG\_WARNING("HTTP request method \"", common::EnumStringConvertor::init()->to\_string(header.method), "\" is not handledsss");

break;

}

}

void ClientHandlerController::handle\_json\_post(common::Socket socket, std::string uri, std::string json\_pkg) {

BDECLARE\_TAG\_SCOPE("ClientHandlerController", \_\_FUNCTION\_\_);

BLOG\_INFO("Socket = ", socket.to\_string(), "; uri = ", uri, " json content = ", json\_pkg);

Json::Value json\_data;

Json::Value json\_response;

Json::CharReaderBuilder json\_reader;

std::istringstream json\_stream(json\_pkg);

std::string err;

common::HttpHeaders http\_header;

std::stringstream request;

if (!Json::parseFromStream(json\_reader, json\_stream, &json\_data, &err)) {

BLOG\_ERROR("Impossible to read json data: ", err);

return;

}

if (uri == "/api/login") {

BLOG\_DEBUG("email = ", json\_data["email"].asString(), "; password = ", json\_data["password"].asString());

bool is\_successful = m\_db\_query->check\_signin\_is\_valid(json\_data["email"].asString(), json\_data["password"].asString());

json\_response["is\_login\_success"] = is\_successful;

request << "HTTP/1.1 200 OK\r\n";

} else {

BLOG\_WARNING("uri = ", uri, "; is not handled!");

}

request << http\_header.extension\_to\_content\_type("json");

auto json\_response\_str = std::move(json\_response.toStyledString());

request << "Content-Length: " << json\_response\_str.size() << "\r\n";

request << "Connection: close\r\n\r\n";

request << json\_response\_str;

socket.send(request.str());

BLOG\_DEBUG("Send http response : ", request.str());

}

} // !server::client\_handler::controllers;

**ContextHandlerController.hpp**

#ifndef POLYCLINIC\_SERVER\_SERVER\_CONTEXTHANDLER\_CONTROLLERS\_CONTEXTHANDLERCONTROLLER\_HPP

#define POLYCLINIC\_SERVER\_SERVER\_CONTEXTHANDLER\_CONTROLLERS\_CONTEXTHANDLERCONTROLLER\_HPP

#include "IContextHandlerModel.hpp"

#include "IContextHandlerInterface.hpp"

#include <memory>

#include <string>

namespace server::context\_handler::controllers {

class ContextHandlerController {

public:

ContextHandlerController(std::weak\_ptr<models::IContextHandlerModel> model,

std::weak\_ptr<view::IContextHandlerInterface> view

);

void init();

void open\_default\_page\_by\_uri(const std::string& uri);

/\*

\* Slots starts;

\*/

public:

common::Slot<std::string, common::Socket> on\_open\_uri\_slot;

void on\_open\_uri(std::string address, common::Socket socket);

/\*

\* Slots ends;

\*/

private:

std::shared\_ptr<models::IContextHandlerModel> m\_context\_handler\_model;

std::shared\_ptr<view::IContextHandlerInterface> m\_context\_handler\_interface;

};

} // !server::context\_handler::controllers;

#endif // !POLYCLINIC\_SERVER\_SERVER\_CONTEXTHANDLER\_CONTROLLERS\_CONTEXTHANDLERCONTROLLER\_HPP;

**ContextHandlerController.cpp**

#include "ContextHandlerController.hpp"

#include "Logger.hpp"

#include "config.hpp"

#include "HttpHeaders.hpp"

#include "File.hpp"

#include <sys/stat.h>

#include <unistd.h>

#include <sys/sendfile.h>

#include <sstream>

#include <fcntl.h>

#include <cstring>

namespace server::context\_handler::controllers {

ContextHandlerController::ContextHandlerController(std::weak\_ptr<models::IContextHandlerModel> model,

std::weak\_ptr<view::IContextHandlerInterface> view)

: m\_context\_handler\_model{model.lock()}

, m\_context\_handler\_interface{view.lock()}

, on\_open\_uri\_slot{std::bind(&ContextHandlerController::on\_open\_uri, this, std::placeholders::\_1, std::placeholders::\_2)}

{

BDECLARE\_TAG\_SCOPE("ContextHandlerController", \_\_FUNCTION\_\_);

BLOG\_INFO("Constructor called");

init();

}

void ContextHandlerController::init() {

common::connect(&m\_context\_handler\_interface->open\_uri, &on\_open\_uri\_slot);

}

void ContextHandlerController::on\_open\_uri(std::string uri, common::Socket socket) {

BDECLARE\_TAG\_SCOPE("ContextHandlerController", \_\_FUNCTION\_\_);

BLOG\_INFO("uri = ", uri, "; socket = ", socket.to\_string());

m\_context\_handler\_model->set\_current\_uri(uri);

std::stringstream request;

request << "HTTP/1.1 200 OK\r\n";

std::string file\_path = "./frontend\_web";

std::int32\_t extension\_index\_pos = uri.find\_last\_of(common::config::EXTENSTION\_SEPARATOR);

std::string extension = "html";

if (extension\_index\_pos != std::string::npos && extension\_index\_pos < uri.size()) {

extension = uri.substr(extension\_index\_pos + 1);

}

common::HttpHeaders http\_header;

request << http\_header.extension\_to\_content\_type(extension);

file\_path += uri;

if (extension == "html" && extension\_index\_pos == std::string::npos) {

file\_path += common::config::DEFAULT\_HTML\_FILE;

}

BLOG\_DEBUG("request = ", request.str(), "; file\_path = ", file\_path);

common::File requested\_file;

if (requested\_file.open(file\_path, common::File::OpenTypeEnum::READ\_ONLY) < 0) {

BLOG\_ERROR("Cannot open file path: ", file\_path, ". Error: ", requested\_file.latest\_error());

std::string error\_response = "HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n<h1>404 Not Found</h1>";

socket.send(error\_response);

requested\_file.close();

return;

}

request << "Content-Length: " << requested\_file.size() << "\r\n";

request << "Connection: close\r\n\r\n";

std::string response = request.str();

if (socket.send(response) < 0) {

BLOG\_ERROR("Failed to write HTTP headers: ", strerror(errno));

requested\_file.close();

return;

}

auto sent\_bytes = socket.sendfile(requested\_file);

if (sent\_bytes == requested\_file.size()) {

BLOG\_DEBUG("Sent file: ", file\_path, " happened successfull. Was sent ", sent\_bytes, " bytes.");

} else {

BLOG\_ERROR("Error sending file. Bytes sent = ", sent\_bytes, "; total size = ", requested\_file.size(), ". Error ", socket.latest\_error());

}

requested\_file.close();

m\_context\_handler\_interface->page\_updated.emit(uri, socket);

}

} // namespace server::context\_handler::controllers

**bars/top\_bar.css**

.header {

background-color: #f8f8f8;

padding: 10px 20px;

}

.header-top {

display: flex;

justify-content: space-between;

align-items: center;

font-size: 12px;

}

.contact-info span {

margin-right: 20px;

}

.header-main {

display: flex;

justify-content: space-between;

align-items: center;

}

.logo {

width: 150px;

}

.nav ul {

list-style-type: none;

padding: 0;

}

.nav li {

display: inline;

margin-right: 20px;

}

.nav a {

text-decoration: none;

color: #000;

font-family: 'fa-solid-900';

}

.btn-appointment {

background-color: #ff3366;

color: #fff;

padding: 10px 20px;

text-decoration: none;

border-radius: 5px;

font-family: 'fa-solid-900';

}

**bars/top\_bar.html**

<div class="header-top">

<div class="contact-info">

<span>CALL US: +41 678-678-456 | +41 678-678-903</span>

<span>594 Sunset Ave. Manahawkin, NJ 08050</span>

<span>Email: jenniewilkerson@info.com</span>

</div>

</div>

<div class="header-main">

<img src="../src/logo.svg" alt="Логотип" class="logo">

<nav class="nav">

<ul>

<li><a href="../">HOME</a></li>

<li><a href="../profile\_page/">PROFILE</a></li>

<li><a href="#blog">BLOG</a></li>

<li><a href="#team">TEAM</a></li>

<li><a href="#faq">FAQ</a></li>

<li><a href="#press">PRESS</a></li>

<li><a href="#testimonials">TESTIMONIALS</a></li>

<li><a href="#contact">CONTACT</a></li>

</ul>

</nav>

<a href="#" class="btn-appointment">Book an Appointment</a>

</div>

**profile\_page/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Index</title>

</head>

<body>

<script src="../scripts/redirect.js"></script>

</body>

</html>

**profile\_page/profile.css**

body {

font-family: Arial, sans-serif;

background-color: #f8f8f8;

margin: 0;

padding: 0;

}

header {

width: 100%;

position: fixed;

top: 0;

z-index: 1000;

}

.container {

width: 80%;

background-color: #fff;

padding: 20px;

border-radius: 10px;

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

margin: 100px auto; /\* Добавим auto чтобы центрировать контейнер \*/

text-align: center; /\* Центрируем текст внутри контейнера \*/

}

.profile-header {

text-align: center;

margin-bottom: 20px;

}

.profile-header h1 {

margin: 0;

}

.profile-header h2 {

margin: 5px 0;

color: #777;

}

.avatar {

width: 100px;

height: 100px;

border-radius: 50%;

margin-bottom: 10px;

}

.profile-content {

display: flex;

justify-content: center;

align-items: flex-start; /\* Обеспечим выравнивание контента по началу \*/

gap: 50px; /\* Добавим промежуток между колонками \*/

}

.personal-info, .medical-info {

width: 45%;

text-align: left; /\* Выравниваем текст по левому краю внутри колонок \*/

}

.personal-info h3, .medical-info h3, .medical-info h4 {

margin-bottom: 10px;

color: #3498db;

}

.personal-info p, .medical-info p, .medical-info ul {

margin: 5px 0;

}

.medical-info ul {

list-style-type: none;

padding: 0;

}

.medical-info ul li {

background-color: #eef6fd;

padding: 10px;

margin-bottom: 5px;

border-radius: 5px;

}

.more-link {

color: #3498db;

text-decoration: none;

font-weight: bold;

margin-left: 5px;

}

.more-link:hover {

text-decoration: underline;

}

.btn-logout {

background-color: #ff3366;

color: #fff;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

margin-top: 10px;

font-family: 'fa-solid-900';

}

**profile\_page/profile.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Profile</title>

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

<link rel="stylesheet" href="../bars/top\_bar.css">

</head>

<body>

<header class="header" w3-include-html="../bars/top\_bar.html"></header>

<div class="container">

<div class="profile-header">

<img src="../src/default\_avatar.jpg" alt="User Avatar" class="avatar">

<h1>Profile</h1>

<h2>Welcome, [User's Name]</h2>

<button class="btn-logout" onclick="logout()">Logout</button>

</div>

<div class="profile-content">

<div class="personal-info">

<h3>Personal Information</h3>

<p><strong>Name:</strong> [User's Name]</p>

<p><strong>Email:</strong> [User's Email]</p>

<p><strong>Phone:</strong> [User's Phone Number]</p>

</div>

<div class="medical-info">

<h3>Medical Card</h3>

<p><strong>Allergens:</strong> [User's Allergens] <a href="#" class="more-link">More...</a></p>

<h4>Booked Doctors</h4>

<ul>

<li>Doctor: [Doctor's Name], Appointment: latest... <a href="#" class="more-link">More...</a></li>

</ul>

<h4>Doctor Appointments</h4>

<ul>

<li>Complaint: latest..., Date: latest... <a href="#" class="more-link">More...</a></li>

</ul>

<h4>Prescription Drugs</h4>

<ul>

<li>Drug: latest..., Description: latest... <a href="#" class="more-link">More...</a></li>

</ul>

</div>

</div>

</div>

<script src="../scripts/include.js"></script>

<script src="../scripts/auth.js"></script>

</body>

</html>

**profile\_page/registation.css**

body {

font-family: Arial, sans-serif;

background-color: #f8f8f8;

margin: 0;

padding: 0;

}

.content-wrapper {

display: flex;

justify-content: center;

align-items: center;

height: calc(100vh - 80px); /\* Учитывая высоту топбара \*/

padding-top: 60px; /\* Высота топбара с небольшим подъемом \*/

}

.container {

display: flex;

justify-content: center;

align-items: center;

width: 100%;

}

.form-container {

background-color: #fff;

padding: 30px;

border-radius: 10px;

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

max-width: 400px;

width: 100%;

text-align: center;

}

.form-container h2 {

margin-bottom: 20px;

}

.form-group {

margin-bottom: 15px;

text-align: left;

}

.form-group label {

display: block;

margin-bottom: 5px;

}

.form-group input {

width: 100%;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

}

.btn {

background-color: #3498db;

color: #fff;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

width: 100%;

margin-top: 10px;

}

p {

margin-top: 20px;

}

a {

color: #3498db;

text-decoration: none;

}

**profile\_page/registation.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Register</title>

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

<link rel="stylesheet" href="../bars/top\_bar.css">

</head>

<body>

<header class="header"></header>

<div class="content-wrapper">

<div class="container">

<div class="form-container">

<h2>Register</h2>

<form id="registerForm">

<div class="form-group">

<label for="fullname">Full Name</label>

<input type="text" id="fullname" required>

</div>

<div class="form-group">

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

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

</div>

<div class="form-group">

<label for="password">Password</label>

<input type="password" id="password" required>

</div>

<div class="form-group">

<label for="confirmpassword">Confirm Password</label>

<input type="password" id="confirmpassword" required>

</div>

<button type="submit" class="btn">Register</button>

<p>Already have an account? <a href="signin.html">Sign In here</a></p>

</form>

</div>

</div>

</div>

<script src="../scripts/include.js"></script>

<script src="../scripts/auth.js"></script>

</body>

</html>

**profile\_page/signin.css**

body {

font-family: Arial, sans-serif;

background-color: #f8f8f8;

margin: 0;

padding: 0;

}

.content-wrapper {

display: flex;

justify-content: center;

align-items: center;

height: calc(100vh - 80px); /\* Учитывая высоту топбара \*/

padding-top: 60px; /\* Высота топбара с небольшим подъемом \*/

}

.container {

display: flex;

justify-content: center;

align-items: center;

width: 100%;

}

.form-container {

background-color: #fff;

padding: 30px;

border-radius: 10px;

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

max-width: 400px;

width: 100%;

text-align: center;

}

.form-container h2 {

margin-bottom: 20px;

}

.form-group {

margin-bottom: 15px;

text-align: left;

}

.form-group label {

display: block;

margin-bottom: 5px;

}

.form-group input {

width: 100%;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

}

.btn {

background-color: #3498db;

color: #fff;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

width: 100%;

margin-top: 10px;

}

p {

margin-top: 20px;

}

a {

color: #3498db;

text-decoration: none;

}

**profile\_page/signin.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Sign In</title>

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

<link rel="stylesheet" href="../bars/top\_bar.css">

</head>

<body>

<header class="header"></header>

<div class="content-wrapper">

<div class="container">

<div class="form-container">

<h2>Sign In</h2>

<form id="signinForm">

<div class="form-group">

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

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

</div>

<div class="form-group">

<label for="password">Password</label>

<input type="password" id="password" required>

</div>

<button type="submit" class="btn">Sign In</button>

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

</form>

</div>

</div>

</div>

<script src="../scripts/include.js"></script>

<script src="../scripts/auth.js"></script>

</body>

</html>

**scripts/auth.js**

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

const registerForm = document.getElementById('registerForm');

const signinForm = document.getElementById('signinForm');

if (signinForm) {

signinForm.addEventListener('submit', async (event) => {

event.preventDefault();

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

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

try {

const response = await fetch('/api/login', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ email, password }),

credentials: 'include'

});

const data = await response.json();

if (response.ok && data.is\_login\_success) {

localStorage.setItem('isLoggedIn', 'true');

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

} else {

alert(data.message || 'Login failed');

}

} catch (error) {

console.error('Error:', error);

}

});

}

if (registerForm) {

registerForm.addEventListener('submit', async (event) => {

event.preventDefault();

const fullname = document.getElementById('fullname').value;

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

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

const confirmpassword = document.getElementById('confirmpassword').value;

if (password !== confirmpassword) {

alert('Passwords do not match');

return;

}

try {

const response = await fetch('/api/register', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ fullname, email, password }),

credentials: 'include'

});

const data = await response.json();

if (response.ok && data.is\_login\_success) {

localStorage.setItem('isLoggedIn', 'true');

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

} else {

alert(data.message || 'Registration failed');

}

} catch (error) {

console.error('Error:', error);

}

});

}

});

function logout() {

localStorage.setItem('isLoggedIn', 'false');

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

}

**scripts/include.js**

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

var includes = document.querySelectorAll('[w3-include-html]');

includes.forEach(function(include) {

var file = include.getAttribute('w3-include-html');

fetch(file)

.then(response => {

if (!response.ok) throw new Error('Network response was not ok');

return response.text();

})

.then(data => include.outerHTML = data)

.catch(error => console.error('There was a problem with the fetch operation:', error));

});

});

**scripts/redirect.js**

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

const isLoggedIn = localStorage.getItem('isLoggedIn');

if (isLoggedIn === 'true') {

window.location.href = '/profile\_page/profile.html';

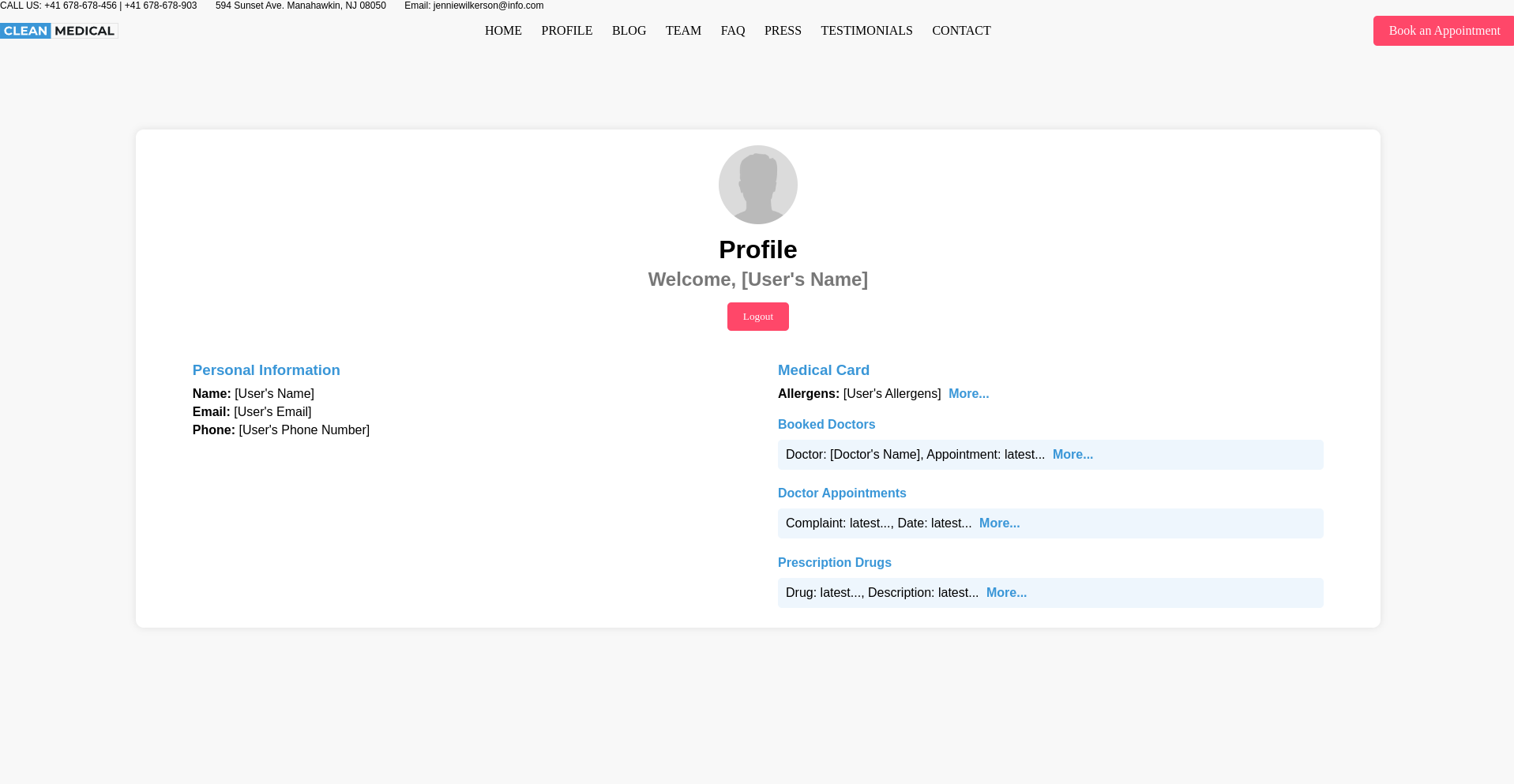
} else {

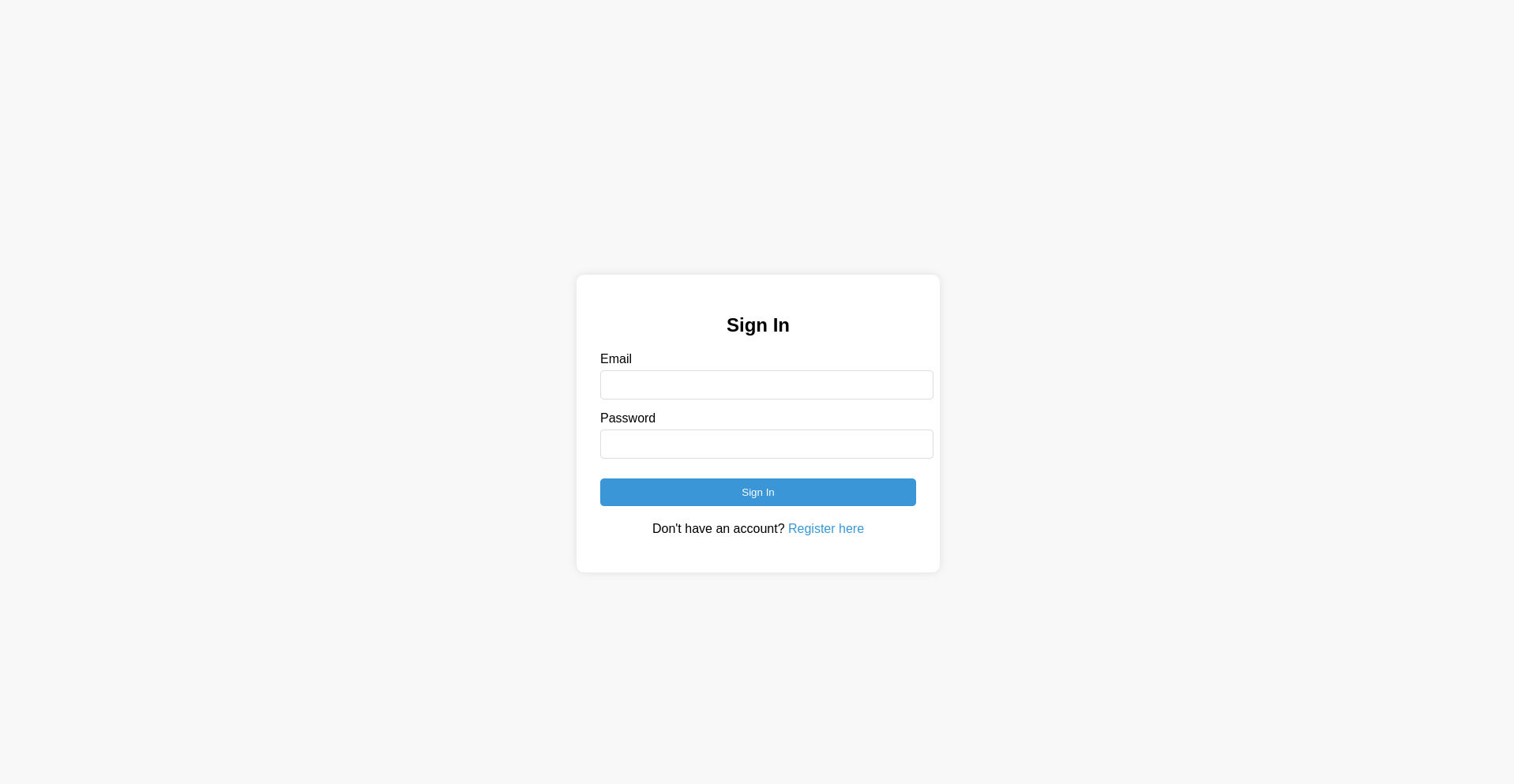
window.location.href = 'signin.html';

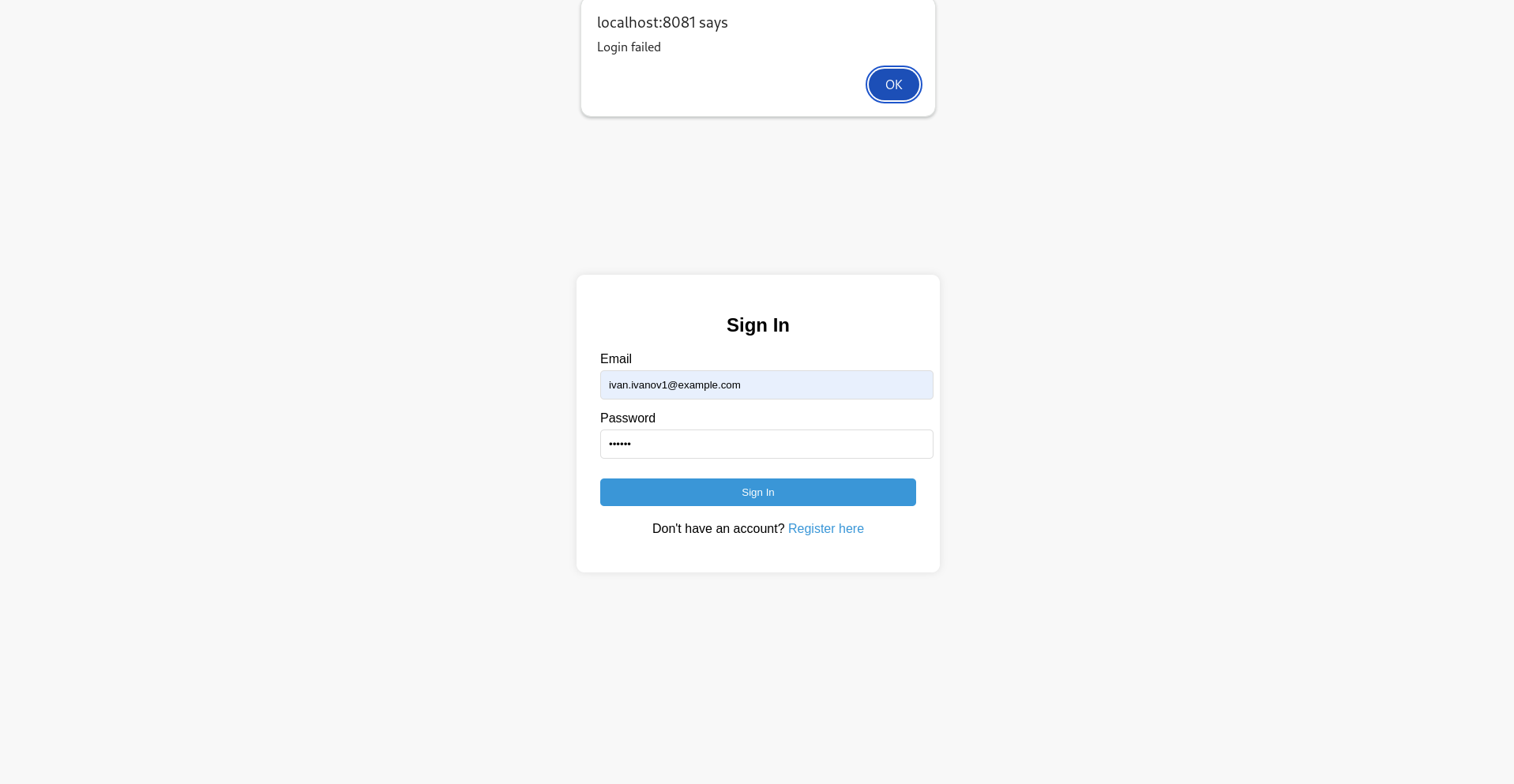
}

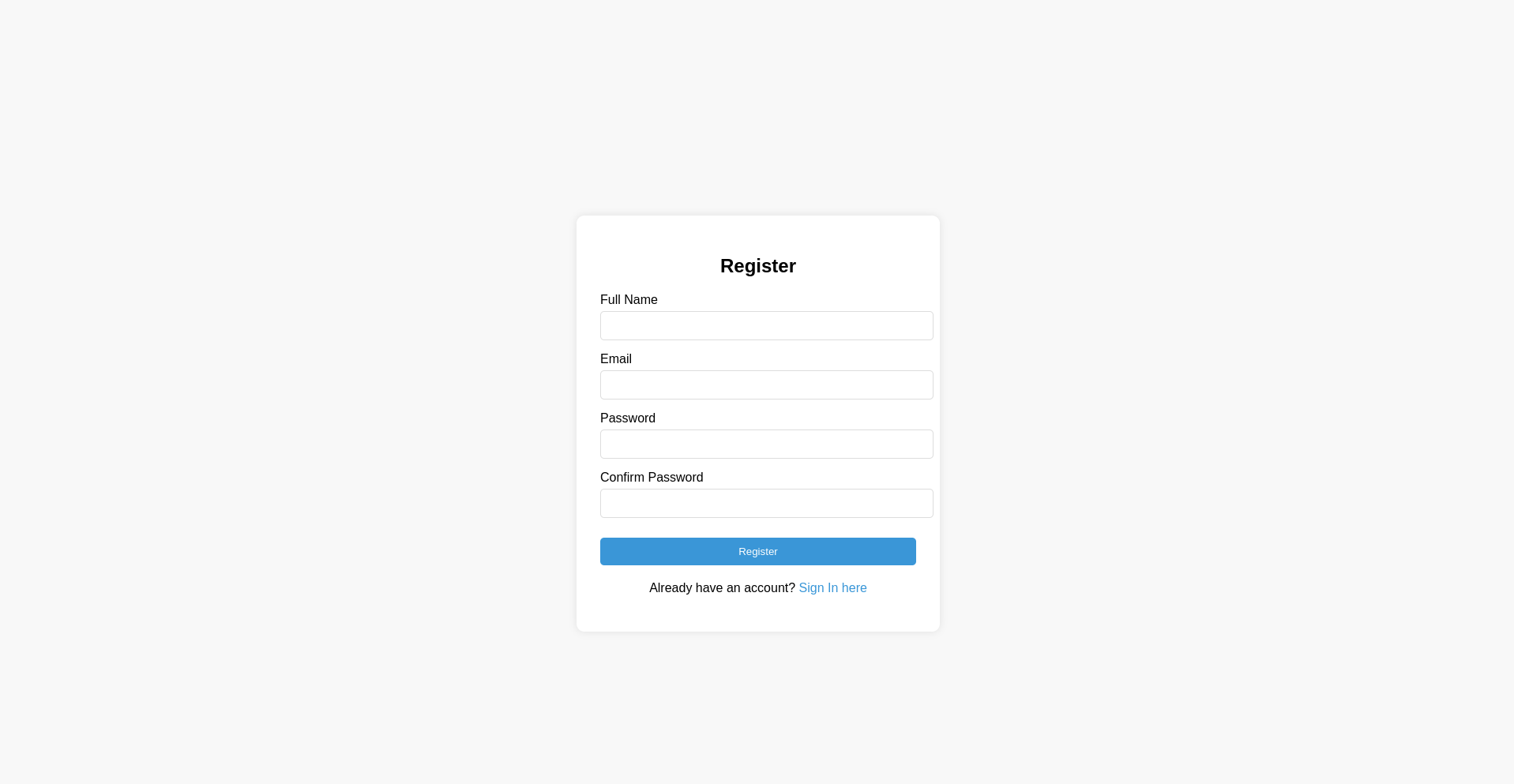
});

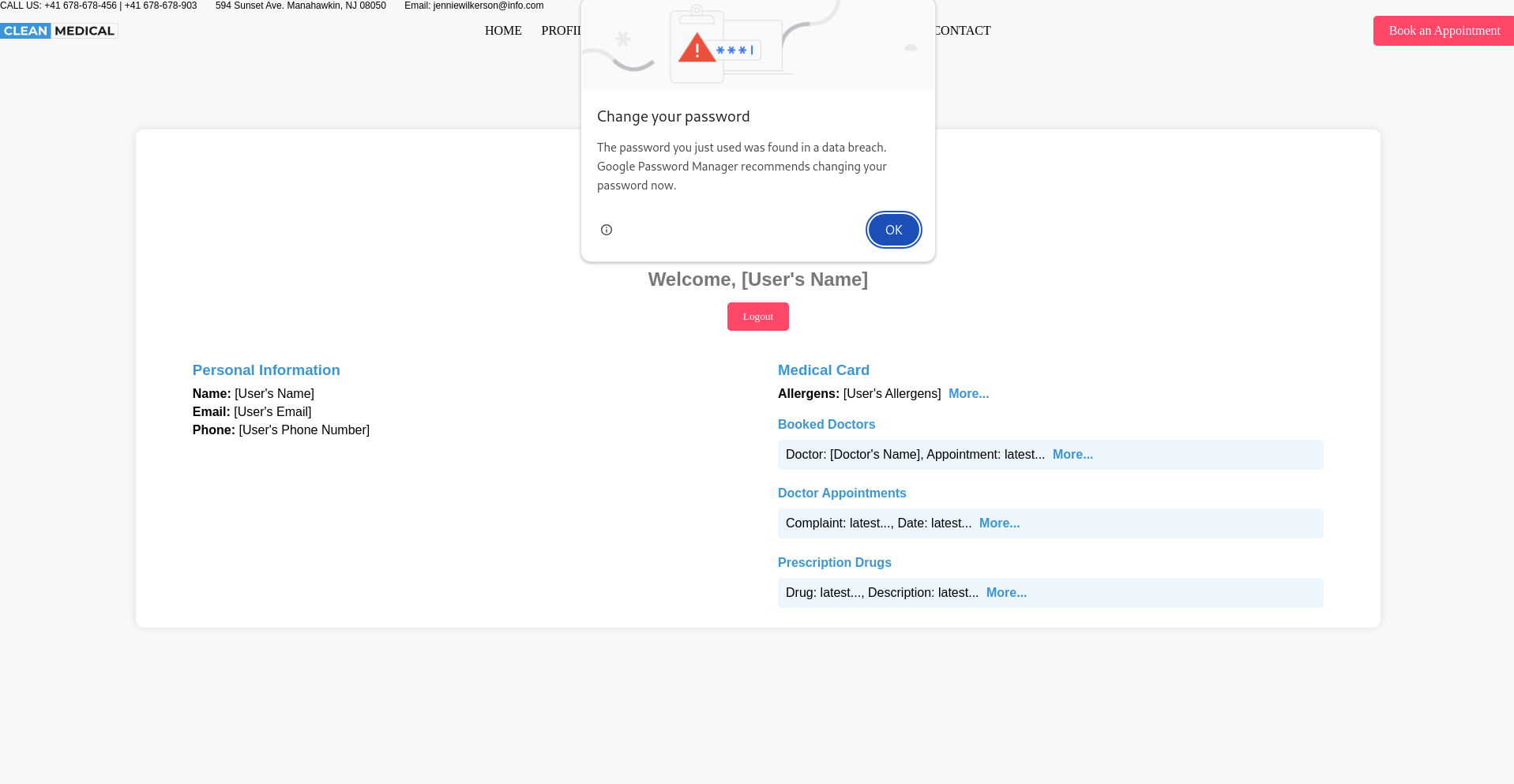
**РЕЗУЛЬТАТ ВИКОНАННЯ**

Рисунок 1 – сторінка користувача

Рисунок 2 – сторінка входа в профіль

Рисунок 3 - некоретно введені дані

Рисунок 4 – сторінка регістрації

Рисунок 5 – вход в профіль зі слабким паролем