Были составлены тест-кейсы к модулю, который я разрабатывал

Таблица тест-кейсов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заголовок | Описание | Предпосылка | Шаги к выполнению | Ожидаемый результат |
| Регистрация пользователя в базу | Попытка регистрации | Сотрудника [semeon.kuzn3tsov@yandex.ru](mailto:semeon.kuzn3tsov@yandex.ru) нет в базе данных | 1. Зайти на сайт 2. Ввести данные в формы 3. Нажать кнопку зарегестрироваться | Пользователь успешно добавлен в базу. |
| Авторизация пользователя в базу | Попытка входа с существующим пользователем | Сотруднику [semeon.kuzn3tsov@yandex.ru](mailto:semeon.kuzn3tsov@yandex.ru) необходимо зарегестрироваться в системе | * 1. После регистрации перейти в окно авторизации   2. Ввести данные в формы   3. Нажать кнопку авторизоваться | Пользователь успешно зашёл в базу |
| Авторизация несуществующего пользователя | Попытка входа с несуществующим email. | Пользовательunknown@example.comне зарегистрирован. | Вызвать LoginAsync("unknown@example.com", "AnyPass123") | Вывод ошибки UserNotFoundException |
| Неверный пароль | Попытка входа в существующий аккаунт, но ввести неверный пароль | Пользовательtest@example.comсуществует. | LoginAsync("test@example.com", "WrongPass123") | Выбрасывается AuthenticationException (илиInvalidCredentialsException) |
| Блокировка после нескольких попыток | Попытка защиты от брутфорса | Пользовательtest@example.comсуществует.  Лимит попыток = 5 | 1. 5 раз вызватьLoginAsync("test@example.com", "WrongPass"). 2. Попытаться войти с правильным паролем. | * После 5 неудачных попыток аккаунт временно блокируется. * 6-я попытка (даже с верным паролем) вызывает AccountLockedException. |

Также были разработаны unit-тесты для регистрации и авторизации:

1. RegisterAsync\_Success\_ReturnsSession() проверяет, что при успешной регистрации метод возвращает SupabaseSession

[TestMethod]

public async Task RegisterAsync\_Success\_ReturnsSession()

{

// Arrange

var expectedSession = new SupabaseSession();

\_authWrapperMock.Setup(x => x.SignUp(

"test@example.com",

"password123"

)).ReturnsAsync(expectedSession);

// Act

var result = await \_authService.RegisterAsync("test@example.com", "password123");

// Assert

Assert.AreEqual(expectedSession, result);

}

1. RegisterAsync\_RetryOnFailure\_ReturnsSession(), проверяет что при первой ошибке метод делает повторную попытку регистрации

[TestMethod]

public async Task RegisterAsync\_RetryOnFailure\_ReturnsSession()

{

// Arrange

var expectedSession = new SupabaseSession();

\_authWrapperMock.SetupSequence(x => x.SignUp(

"test@example.com",

"password123"

))

.ThrowsAsync(new Exception("First attempt failed"))

.ReturnsAsync(expectedSession);

// Act

var result = await \_authService.RegisterAsync("test@example.com", "password123");

// Assert

Assert.AreEqual(expectedSession, result);

\_authWrapperMock.Verify(x => x.SignUp(

"test@example.com",

"password123"

), Times.Exactly(2));

}

1. RegisterAsync\_NullEmail\_ThrowsArgumentNullException() проверяет, что если email == null, выбрасывается ArgumentNullException

[TestMethod]

public async Task RegisterAsync\_NullEmail\_ThrowsArgumentNullException()

{

// Act & Assert

await Assert.ThrowsExceptionAsync<ArgumentNullException>(async () =>

{

await \_authService.RegisterAsync(null, "password123");

});

}

1. RegisterAsync\_NullPassword\_ThrowsArgumentNullException() проверяет, что если password == null, выбрасывается ArgumentNullException

[TestMethod]

public async Task RegisterAsync\_NullPassword\_ThrowsArgumentNullException()

{

// Act & Assert

await Assert.ThrowsExceptionAsync<ArgumentNullException>(async () =>

{

await \_authService.RegisterAsync("user@example.com", null);

});

}

1. RegisterAsync\_CallsSignUp\_WithCorrectParameters() проверяет, что signUp вызывается с правильными аргументами

[TestMethod]

public async Task RegisterAsync\_CallsSignUp\_WithCorrectParameters()

{

// Arrange

string expectedEmail = "check@domain.com";

string expectedPassword = "super-secret";

\_authWrapperMock.Setup(x => x.SignUp(expectedEmail, expectedPassword))

.ReturnsAsync(new SupabaseSession());

// Act

await \_authService.RegisterAsync(expectedEmail, expectedPassword);

// Assert

\_authWrapperMock.Verify(x => x.SignUp(expectedEmail, expectedPassword), Times.Once);

}

|  |  |
| --- | --- |
| RegisterAsync\_Success\_ReturnsSession | Возвращается правильная сессия при успехе |
| RegisterAsync\_RetryOnFailure\_ReturnsSession | Повторяется попытка при ошибке |
| RegisterAsync\_NullEmail\_ThrowsArgumentNullException | Проверка null email |
| RegisterAsync\_NullPassword\_ThrowsArgumentNullException | Проверка null password |
| RegisterAsync\_CallsSignUp\_WithCorrectParameters | SignUp вызывается с нужными аргументами |