

מגישיים:

רפאל קואפיק 337614747
שי קרונפלד 322234782
משה אסקרוב 314085986

חלק א' - התרחישים לבדיקה - משמרות:

תרחיש 1:

שם התרחיש: צפיה במשמרות שהוקצטו לעובד.

תנאים מוקדמים:

1. המשמש מחובר למערכת כדי לעבוד שכבר אושר.
2. העובד משוייך לפחות לצוות אחד בחברה.
3. קיימת לפחות משמרת אחת שהוקצתה לעובד בשבוע הנוכחי ממנהל של הצוות שלו.
4. קיימת גישה לאינטרנט והאפליקציה פועילה.

שלבים לביצוע:

1. פתיחת האפליקציה כמנהל של עובד והתחברות עם שם משתמש וסיסמה.
2. ניווט למנסך "manage shifts".
3. לשירות משמרות בשבוע הקרוב ובשבוע שאחריו לעובד.
4. פתיחת האפליקציה והתחברות באותו עובד עם שם משתמש וסיסמה.
5. ניווט למנסך המשמרות שלו.
6. **בדיקה** שמצווג טווח תאריכים של השבוע הנוכחי.
7. **בדיקה** שמצווגת רשימת משמרות של העובד בלבד.
8. לחיצה על כפתור שבוע הבא.
9. **בדיקה** שמשמרות של שבוע הבא מוצגות.
10. לחיצה על שבוע קודם וחזרה לשבוע המקורי.
11. **בדיקה** אחרונה שהכל תקין וסיום התרחיש.

תוצאה צפוייה:

-מצווגות רק המשמרות של העובד בהתאם לשבוע הנבחר.
-ניווט בין שבועות מעדכן את הרשימה ואת טווח התאריכים.
-אין הציגה של משמרות שאין שייכות לעובד.

תרחיש 2:

שם התרחיש: יצרת בקשה חדשה להחלפת משמרות

תנאים מוקדמים:

1. המשמש מחובר למערכת כדי לעבוד.
2. העובד משוייך לפחות עם משמרות עתידיות.
3. קיימת לפחות משמרת אחת עתידית שהוקצתה לעובד.

שלבים לביצוע:

1. כניסה לאפליקציהעובד.
2. פתיחת מסך החלפת משמרות.
3. בחירת צוות מתוך הרשימה.
4. לחיצה על כפתור בקשה חדשה.
5. בחירת סוג הבקשה (ויתור / החלפה).
6. בחירת משמרות עתידית מרשימה המשמרות.
7. אישור ושליחת הבקשה.
8. **בדיקה** שהבקשה מופיעה לעובד בחלק של "בקשות שפותחת" בעמוד של "החלפת משמרות".
9. התחרבות לאחד המנהלים של הקבוצה שבה התקבלה ההצעה.
10. כניסה למסך של "אישור החלפות".
11. **בדיקה** שהבקשה להחלפה נמצאת בדף.

תוצאה צפוייה:

1. הבקשה מועברת בהצלחה.
2. הבקשה מופיעה ברשימה הבקשות של העובד.
3. סטוס הבקשה מוגדר כ-ממתין לאישור / פתוח בהתאם לוגיקה במערכת.
4. הבקשה מופיעה ברשימה הפתוחות אצל אחד המנהלים של הקבוצה.

תרחיש 3:

שם התרחיש: אישור/דחיה בקשה החלפת משמרות על ידי מנהל.

תנאים מוקדמים:

1. המשתמש מחובר למערכת כמנהל.
2. קיימת בקשה להחלפת משמרות במצב "ממתין לאישור".
3. הבקשה שייצת לצוות שהמנהל אחראי עליו.

שלבים לביצוע:

1. התחרבות כמנהל.
2. פתיחת מסך אישורי החלפות.
3. בחירת צוות מהרשימה.
4. צפיה ברשימה הבקשות המתיניות.
5. לחיצה על כפתור אישור/דחיה ליד אחת הבקשות.
6. **בדיקה** שהבקשה לא מופיעה יותר במסך בקשות פתוחות.
7. התחרבות למשתמש של העובד שייצר את הבקשה.
8. במידה והבקשה התקבלה. **בדיקה** שב"בקשות של" הפניה התעדכנה במצב אושר ובאמת התחלפו המשמרות/ המשמרות שויתרת עלייה כתע לשוייר לה מי שהחליף את העובד ולא העובד עצמו.
9. במידה והחלפה לא אושרה. **בדיקה** שהבקשה חזרה במצב של "פתוחה" ועובדים יכולים שוב להגיש את ההצעות שלהם לתחלה.

תוצאה צפוייה:

1. הבקשה מאושרת/מוסרבת בהצלחה.
2. הבקשה נעלמת מרשימה הבקשות המתיניות.
3. סטוס הבקשה מתעדכן בהתאם (מאושר).
4. בוצע השינוי המתאים בלוח המשמרות והתעדכנו החלפות.

חלק ב' - Unit tests

הטסטים יבדקו על תהליכי הצעת

טעו 1. הבדיקות האלו בודקות שהפונקציה `findOtherParticipantId` מחזירה נכון את זהה המשתתף השני בשיחה, ומטפלת כמובן במקרים של רשימה ריקה, ערכים חסרים, או כאשר אין משתתף נוסף מלבד המשתמש הנוכחי.

```
;package com.example.workconnect.adapters

;import com.example.workconnect.models.ChatConversation
;import com.example.workconnect.utils.ChatUtils
;import org.junit.Test
;*.import static org.junit.Assert
;*.import static org.mockito.Mockito

;import java.util.Arrays
;import java.util.List

} public class ChatConversationAdapterTest

;"private static final String CURRENT_USER_ID = "user1

Test@
} ()public void testGetOtherParticipantId_TwoParticipants_ReturnsOtherUserId
Setting up the conditions .1 //
; (ChatConversation mockConversation = mock(ChatConversation.class
; ("List<String> participantIds = Arrays.asList("user1", "user2"
; (when(mockConversation.getParticipantIds()).thenReturn(participantIds

Calling the function under test .2 //
String result =
ChatUtils.findOtherParticipantId(mockConversation.getParticipantIds(),
; (CURRENT_USER_ID

Assertions to verify the expected result .3 //
; (assertNotNull("Result should not be null", result
; (assertEquals("Should return the other participant ID", "user2", result
{

Test@
} ()public void testGetOtherParticipantId_ConversationNull_ReturnsNull
Setting up the conditions .1 //
; List<String> participantIds = null

Calling the function under test .2 //
```

```
String result = ChatUtils.findOtherParticipantId(participantIds,
; (CURRENT_USER_ID

Assertions to verify the expected result .3 //
; assertNull("When participantIds is null, should return null", result
{



Test@
} ()public void testGetOtherParticipantId_ParticipantIdsNull_ReturnsNull
Setting up the conditions .1 //
;List<String> participantIds = null

Calling the function under test .2 //
String result = ChatUtils.findOtherParticipantId(participantIds,
; (CURRENT_USER_ID

Assertions to verify the expected result .3 //
; assertNull("When participantIds is null, should return null", result
{



Test@
} ()public void testGetOtherParticipantId_NoOtherParticipant_ReturnsNull
Setting up the conditions .1 //
;"List<String> participantIds = Arrays.asList("user1

Calling the function under test .2 //
String result = ChatUtils.findOtherParticipantId(participantIds,
; (CURRENT_USER_ID

Assertions to verify the expected result .3 //
; assertNull("When no other participant exists, should return null", result
{



Test@
public void
} ()testGetOtherParticipantId_MultipleParticipants_ReturnsFirstOtherUserId
Setting up the conditions .1 //
;"List<String> participantIds = Arrays.asList("user1", "user2", "user3

Calling the function under test .2 //
String result = ChatUtils.findOtherParticipantId(participantIds,
; (CURRENT_USER_ID

Assertions to verify the expected result .3 //
; assertNotNull("Result should not be null", result
```

```

assertEquals("Should return the first other participant ID", "user2",
; (result
assertNotEquals("Should not return current user ID", CURRENT_USER_ID,
; (result
{
{
}

```

Results:



טוטו 2. הבדיקות האלו בודקות שהfonנקציה `insertDateSeparators` מוסיפה מפריד תאריך בצורה נכונה בין ה הודעות בצ'אט, כולל מקרים של הודעות מאותנו יום ומימים שונים, ובשימוש ב-Mockito.

```

;package com.example.workconnect.adapters

;import com.example.workconnect.models.ChatItem
;import com.example.workconnect.models.ChatMessage
;import com.example.workconnect.utils.ChatUtils
;import org.junit.Test
;*.import static org.junit.Assert
;*.import static org.mockito.Mockito

;import java.util.ArrayList
;import java.util.Calendar
;import java.util.Date
;import java.util.List

} public class ChatMessageAdapterTest

;"private static final String TEST_USER_ID = "testUserId

Test@
public void
} ()testInsertDateSeparators_NullMessages_ReturnsEmptyList
Setting up the conditions .1 //
;List<ChatMessage> messages = null

```

```
Calling the function under test .2 //
List<ChatItem> result =
; (ChatUtils.insertDateSeparators(messages

Assertions to verify the expected result .3 //
; (assertNotNull("Result should not be null", result
assertEquals("Null messages should return empty list", 0,
; ()result.size
{

Test@
public void
) testInsertDateSeparators_EmptyList_ReturnsEmptyList
Setting up the conditions .1 //
; ()<>List<ChatMessage> messages = new ArrayList

Calling the function under test .2 //
List<ChatItem> result =
; (ChatUtils.insertDateSeparators(messages

Assertions to verify the expected result .3 //
; (assertNotNull("Result should not be null", result
assertEquals("Empty list should return empty list", 0,
; ()result.size
{

Test@
public void
) testInsertDateSeparators_SingleMessage_ReturnsSeparatorAndMessage
} (
Setting up the conditions .1 //
; ()<>List<ChatMessage> messages = new ArrayList
ChatMessage message = createTestMessage("msg1", new
; ()Date
; (messages.add(message

Calling the function under test .2 //
List<ChatItem> result =
; (ChatUtils.insertDateSeparators(messages

Assertions to verify the expected result .3 //
; (assertNotNull("Result should not be null", result
assertEquals("Single message should return separator +
; ()message", 2, result.size
assertTrue("First item should be a date separator",
; ()result.get(0).isDateSeparator
assertTrue("Second item should be a message",
; ()result.get(1).isMessage
```

```
assertEquals("Message should match", message,
; ()result.get(1).getMessage
{

Test@
public void
testInsertDateSeparators_MultipleMessagesSameDay_NoAdditionalSepar
} ()ators
Setting up the conditions .1 //
; ()<>List<ChatMessage> messages = new ArrayList
; ()Date sameDay = new Date
; ((messages.add(createTestMessage("msg1", sameDay
; ((messages.add(createTestMessage("msg2", sameDay
; ((messages.add(createTestMessage("msg3", sameDay

Calling the function under test .2 //
List<ChatItem> result =
; (ChatUtils.insertDateSeparators(messages

Assertions to verify the expected result .3 //
; (assertNotNull("Result should not be null", result
assertEquals("Three messages same day should have 1
; ()separator + 3 messages", 4, result.size
assertTrue("First item should be a date separator",
; ()result.get(0).isDateSeparator
assertTrue("Second item should be a message",
; ()result.get(1).isMessage
assertTrue("Third item should be a message",
; ()result.get(2).isMessage
assertTrue("Fourth item should be a message",
; ()result.get(3).isMessage
{

Test@
public void
()testInsertDateSeparators_MessagesDifferentDays_InsertsSeparators
}
Setting up the conditions .1 //
; ()<>List<ChatMessage> messages = new ArrayList
; ()Calendar cal = Calendar.getInstance
; ()Date day1 = cal.getTime

; (cal.add(Calendar.DAY_OF_YEAR, 1
; ()Date day2 = cal.getTime

; (cal.add(Calendar.DAY_OF_YEAR, 1
; ()Date day3 = cal.getTime
```

```

; ((messages.add(createTestMessage("msg1", day1
; ((messages.add(createTestMessage("msg2", day2
; ((messages.add(createTestMessage("msg3", day3

Calling the function under test .2 //
List<ChatItem> result =
; (ChatUtils.insertDateSeparators(messages

Assertions to verify the expected result .3 //
; (assertNotNull("Result should not be null", result
assertEquals("Three messages different days should have 3
; ()separators + 3 messages", 6, result.size
assertTrue("Item 0 should be separator",
; ()result.get(0).isDateSeparator
assertTrue("Item 1 should be message",
; ()result.get(1).isMessage
assertTrue("Item 2 should be separator",
; ()result.get(2).isDateSeparator
assertTrue("Item 3 should be message",
; ()result.get(3).isMessage
assertTrue("Item 4 should be separator",
; ()result.get(4).isDateSeparator
assertTrue("Item 5 should be message",
; ()result.get(5).isMessage
{

Test@
public void
testInsertDateSeparators_MockedMessagesDifferentDays_InsertsSepa
} ()tors
Setting up the conditions .1 //
; ()<>List<ChatMessage> messages = new ArrayList

; ()Calendar cal = Calendar.getInstance
; ()Date day1 = cal.getTime
; (cal.add(Calendar.DAY_OF_YEAR, 1
; ()Date day2 = cal.getTime

; (ChatMessage mockMessage1 = mock(ChatMessage.class
; (when(mockMessage1.getSentAt()).thenReturn(day1
; ("when(mockMessage1.getId()).thenReturn("msg1

; (ChatMessage mockMessage2 = mock(ChatMessage.class
; (when(mockMessage2.getSentAt()).thenReturn(day2
; ("when(mockMessage2.getId()).thenReturn("msg2

; (messages.add(mockMessage1
; (messages.add(mockMessage2

```

```

Calling the function under test .2 //
List<ChatItem> result =
; (ChatUtils.insertDateSeparators(messages

Assertions to verify the expected result .3 //
; (assertNotNull("Result should not be null", result
assertEquals("Two messages different days should have 2
; ()separators + 2 messages", 4, result.size
assertTrue("Item 0 should be separator",
; ()result.get(0).isDateSeparator
assertTrue("Item 1 should be message",
; ()result.get(1).isMessage
assertTrue("Item 2 should be separator",
; ()result.get(2).isDateSeparator
assertTrue("Item 3 should be message",
; ()result.get(3).isMessage

; ()verify(mockMessage1, times(3)).getSentAt
; ()verify(mockMessage2, atLeastOnce()).getSentAt
{

} (private ChatMessage createTestMessage(String id, Date sentAt
; ()ChatMessage message = new ChatMessage
; (message.setId(id
; (message.setSenderId(TEST_USER_ID
; (message.setText("Test message " + id
; (message.setSentAt(sentAt
; (message.setStatus(ChatMessage.MessageStatus.SENT
; return message
{
{

```

Results:



טוט 3. הבדיקות הללו בודקות את הפונקציות ב-`ChatUtils`: השוואת תאריכים, מציאת משתף אחר בשיחה, והוספת מפ прид תאריך בין הודעות – כולל שימוש ב-`Mockito`.

```

;package com.example.workconnect.utils

;import com.example.workconnect.models.ChatItem
;import com.example.workconnect.models.ChatMessage
;import org.junit.Test
;*.import static org.junit.Assert

```

```
.*import static org.mockito.Mockito

;import java.util.ArrayList
;import java.util.Arrays
;import java.util.Calendar
;import java.util.Date
;import java.util.List

} public class ChatUtilsTest

Test@
} ()public void testIsDifferentDay_SameDay_ReturnsFalse
Setting up the conditions .1 //
;()Calendar cal = Calendar.getInstance
;()Date date1 = cal.getTime
;()Date date2 = cal.getTime

Calling the function under test .2 //
;boolean result = DateHelper.isDifferentDay(date1, date2

Assertions to verify the expected result .3 //
assertFalse("Two dates on the same day should return false",
;result
{

Test@
public void
} ()testFindOtherParticipantId_TwoParticipants_ReturnsOtherUserId
Setting up the conditions .1 //
List<String> participantIds = Arrays.asList("user1",
;"user2
;"String currentUserId = "user1

Calling the function under test .2 //
String result =
; (ChatUtils.findOtherParticipantId(participantIds, currentUserId

Assertions to verify the expected result .3 //
; assertNotNull("Result should not be null", result
assertEquals("Should return the other participant ID",
;"user2", result
{

Test@
public void
testInsertDateSeparators_MockedMessagesDifferentDays_InsertsSepa
} ()tors
Setting up the conditions .1 //
```

```

; ()<>List<ChatMessage> messages = new ArrayList

; ()Calendar cal = Calendar.getInstance
; ()Date day1 = cal.getTime
; (cal.add(Calendar.DAY_OF_YEAR, 1
; ()Date day2 = cal.getTime

; (ChatMessage mockMessage1 = mock(ChatMessage.class
; (when(mockMessage1.getSentAt()).thenReturn(day1
; (messages.add(mockMessage1

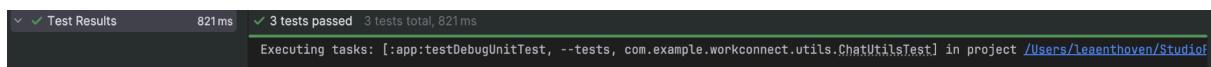
; (ChatMessage mockMessage2 = mock(ChatMessage.class
; (when(mockMessage2.getSentAt()).thenReturn(day2
; (messages.add(mockMessage2

Calling the function under test .2 //
List<ChatItem> result =
; (ChatUtils.insertDateSeparators(messages

Assertions to verify the expected result .3 //
; (assertNotNull("Result should not be null", result
assertEquals("Two messages on different days should have 2
; ()separators + 2 messages", 4, result.size
assertTrue("First item should be a date separator",
; ()result.get(0).isDateSeparator
assertTrue("Second item should be a message",
; ()result.get(1).isMessage
assertTrue("Third item should be a date separator",
; ()result.get(2).isDateSeparator
assertTrue("Fourth item should be a message",
; ()result.get(3).isMessage
{
{

```

Results:



חלק ג' - כתיבת טסטים על UI - Espresso :

הטסטים יבדקו על תהליכי החופשות.

טעות 1: עובד מנסה להציג בקשת חופשה ללא יתרת ימים מספקת.

```
package com.example.workconnect.ui.vacations;
import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.action.ViewActions.*;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.*;
import static org.hamcrest.Matchers.equalTo;
import androidx.test.espresso.contrib.PickerActions;

import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.rule.ActivityTestRule;

import com.example.workconnect.R;
import com.example.workconnect.ui.home.HomeActivity;
import com.google.firebase.auth.FirebaseAuth;

import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(AndroidJUnit4.class)
public class VacationEdgeCaseUiTest {

    private static final String EMPLOYEE_EMAIL = "MANAGE1@M.COM";
    private static final String EMPLOYEE_PASSWORD = "123456";

    @Rule
    public ActivityTestRule<NewVacationRequestActivity> activityRule =
```

```
new ActivityTestRule<>(NewVacationRequestActivity.class);

@Before
public void ensureLoggedIn() throws Exception {
    FirebaseAuth auth = FirebaseAuth.getInstance();
    if (auth.getCurrentUser() != null) return;

    final Object lock = new Object();
    final boolean[] done = {false};
    final Exception[] err = {null};

    auth.signInWithEmailAndPassword(EMPLOYEE_EMAIL, EMPLOYEE_PASSWORD)
        .addOnCompleteListener(task -> {
            synchronized (lock) {
                if (!task.isSuccessful()) {
                    err[0] = task.getException();
                }
                done[0] = true;
                lock.notifyAll();
            }
        });
}

long deadline = System.currentTimeMillis() + 10_000;
synchronized (lock) {
    while (!done[0] && System.currentTimeMillis() < deadline) {
        lock.wait(200);
    }
}

if (!done[0]) throw new AssertionError("Login timed out");
if (err[0] != null) throw err[0];
}

@Test
public void submitVacationRequest_notEnoughBalance_showsErrorToast() {

    // Start date
    onView(withId(R.id.et_start_date)).perform(click());

    onView(withClassName(equalTo(android.widget.DatePicker.class.getName())))
        .perform(PickerActions.setDate(2026, 2, 1));
    onView(withId(android.R.id.button1)).perform(click());

    // End date
    onView(withId(R.id.et_end_date)).perform(click());

    onView(withClassName(equalTo(android.widget.DatePicker.class.getName())))
        .perform(PickerActions.setDate(2026, 2, 15));
    onView(withId(android.R.id.button1)).perform(click());

    onView(withId(R.id.et_reason))
        .perform(replaceText("Vacation longer than balance"),
closeSoftKeyboard());
}
```

```

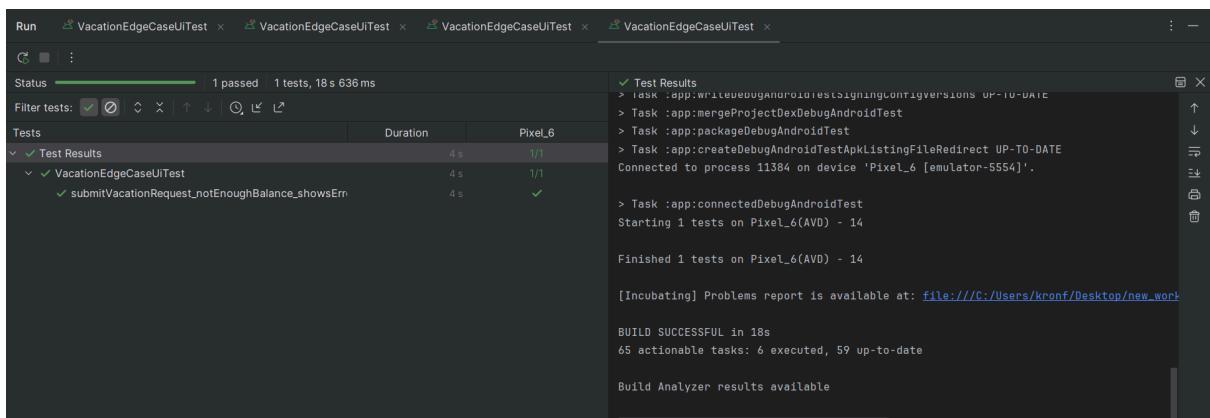
        onView(withId(R.id.btn_send_request)).perform(click());

        onView(withId(R.id.tv_title))
            .check(matches(isDisplayed()));

        onView(withId(R.id.et_reason))
            .check(matches(withText("Vacation longer than balance")));
    }
}

```

ניתן לראות שהטסט עבר בהצלחה:



טסט זה בודק מקרה קצה בתהילך ניהול החופשות. הטסט מדמיה עובד שמדובר טופס בקשה חופשה: בוחר תאריך התחלת, תאריך סיום ומזמן סיבה, ולאחר מכן לוחץ על כפתור "Send request". לאחר פעולה השילחה, הטסט מאמת את מצב ה- UI בקורס שהמסך נשאר פתוח (כלומר לא מתבצע מעבר/סגירה לאחר השילחה) ושודות הטופס נשארים כפי שהזנו. אימוטה זה מצביע על כך שהבקשה לא נשלחה בהצלחה עקב יתרת ימי חופשה לא מספקת.

טסט 2: מנהל מאשר בקשה חופשה של עובד

```

package com.example.workconnect.ui.vacations;

import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.isDisplayed;
import static androidx.test.espresso.matcher.ViewMatchers.withId;

import android.view.View;

import androidx.recyclerview.widget.RecyclerView;
import androidx.test.core.app.ActivityScenario;

```

```
import androidx.test.espresso.PerformException;
import androidx.test.espresso.UiController;
import androidx.test.espresso.ViewAction;
import androidx.test.espresso.contrib.RecyclerViewActions;
import androidx.test.ext.junit.runners.AndroidJUnit4;

import com.example.workconnect.R;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;

import org.hamcrest.Matcher;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;

import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;

import static org.hamcrest.Matchers.allOf;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertTrue;

@RunWith(AndroidJUnit4.class)
public class ManagerApproveVacationUiTest {

    private static final String MANAGER_EMAIL = "shay@mid.com";
    private static final String MANAGER_PASSWORD = "123456";

    private static final String COLLECTION = "vacation_requests";

    @Before
    public void ensureLoggedInAsManager() throws Exception {
        FirebaseAuth auth = FirebaseAuth.getInstance();
        if (auth.getCurrentUser() != null) return;

        final Object lock = new Object();
        final boolean[] done = {false};
        final Exception[] err = {null};

        auth.signInWithEmailAndPassword(MANAGER_EMAIL, MANAGER_PASSWORD)
            .addOnCompleteListener(task -> {
                synchronized (lock) {
                    if (!task.isSuccessful()) {
                        err[0] = task.getException();
                    }
                    done[0] = true;
                    lock.notifyAll();
                }
            });
    }

    long deadline = System.currentTimeMillis() + 10_000;
    synchronized (lock) {
        while (!done[0] && System.currentTimeMillis() < deadline) {
```

```

        lock.wait(200);
    }
}

if (!done[0]) throw new AssertionError("Login timed out");
if (err[0] != null) throw err[0];

assertNotNull(auth.getCurrentUser());
}

@Test
public void approveRequest_statusBecomesApproved_andItemRemoved() {

    ActivityScenario.launch(PendingVacationRequestsActivity.class);

    onView(withId(R.id.tv_title)).check(matches(isDisplayed()));

    onView(withId(R.id.rv_requests)).perform(waitForRecyclerViewItemCount(1,
8000));

    final int[] before = new int[1];

    onView(withId(R.id.rv_requests)).perform(getRecyclerViewItemCount(before));

    onView(withId(R.id.rv_requests))
        .perform(RecyclerViewActions.scrollToPosition(0));

    final String[] requestId = new String[1];
    onView(withId(R.id.rv_requests))
        .perform(getRequestIdFromItemAtPosition(0, requestId));

    onView(withId(R.id.rv_requests))
        .perform(RecyclerViewActions.actionOnItemAtPosition(
            0,
            clickChildViewWithId(R.id.btn_approve)
        ));

    onView(withId(R.id.rv_requests))
        .perform(waitForRecyclerViewExactItemCount(before[0] - 1,
10_000));

    waitForFirestoreStatus(requestId[0], "APPROVED", 10_000);
}

// ===== Helpers =====

private static ViewAction getRequestIdFromItemAtPosition(int position,
String[] out) {
    return new ViewAction() {
        @Override public Matcher<View> getConstraints() { return
allof(isDisplayed()); }
        @Override public String getDescription() { return "Get requestId
from itemView tag"; }

```

```
    @Override public void perform(UiController uiController, View
view) {
        RecyclerView rv = (RecyclerView) view;
        RecyclerView.ViewHolder vh =
rv.findViewHolderForAdapterPosition(position);
        if (vh == null) throw new AssertionError("ViewHolder is
null");
        Object tag = vh.itemView.getTag();
        if (tag == null) throw new AssertionError("itemView tag is
null");
        out[0] = tag.toString();
    }
};

private static void waitForFirestoreStatus(String requestId, String
expected, long timeoutMs) {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    long start = System.currentTimeMillis();

    while (System.currentTimeMillis() - start < timeoutMs) {
        CountDownLatch latch = new CountDownLatch(1);
        final String[] value = new String[1];

        db.collection(COLLECTION).document(requestId).get()
            .addOnSuccessListener(snap -> {
                value[0] = snap.getString("status");
                latch.countDown();
            });
    }

    try {
        latch.await(3, TimeUnit.SECONDS);
    } catch (InterruptedException ignored) {}

    if (expected.equals(value[0])) return;

    try { Thread.sleep(250); } catch (InterruptedException ignored) {}
}

throw new AssertionError("status did not become " + expected);
}

// --- Recycler helpers ---

private static ViewActionwaitForRecyclerViewMinItemCount(int minCount,
long timeoutMs) {
    return new ViewAction() {
        @Override public Matcher<View> getConstraints() { return
allof(isDisplayed()); }

        @Override public String getDescription() {
            return "Wait for RecyclerView itemCount >= " + minCount + "
within " + timeoutMs + "ms";
        }
    };
}
```

```
        }

    @Override public void perform(UiController uiController, View
view) {
    RecyclerView rv = (RecyclerView) view;
    long start = System.currentTimeMillis();

    while (true) {
        RecyclerView.Adapter<?> adapter = rv.getAdapter();
        int count = (adapter == null) ? 0 :
adapter.getItemCount();

        if (count >= minCount) return;

        if (System.currentTimeMillis() - start > timeoutMs) {
            throw new PerformException.Builder()
                .withActionDescription(getDescription())
                .withViewDescription("RecyclerView itemCount
stayed " + count)
                .build();
        }

        uiController.loopMainThreadForAtLeast(300);
    }
}

private static ViewAction getRecyclerViewItemCount(final int[] outCount) {
    return new ViewAction() {
        @Override public Matcher<View> getConstraints() { return
allOf(isDisplayed()); }
        @Override public String getDescription() { return "Get item
count"; }
        @Override public void perform(UiController uiController, View
view) {
            RecyclerView rv = (RecyclerView) view;
            outCount[0] = rv.getAdapter() == null ? 0 :
rv.getAdapter().getItemCount();
        }
    };
}

private static ViewAction clickChildViewWithId(int id) {
    return new ViewAction() {
        @Override public Matcher<View> getConstraints() { return
allOf(isDisplayed()); }
        @Override public String getDescription() { return "Click child"; }
        @Override public void perform(UiController uiController, View
view) {
            View v = view.findViewById(id);
            if (v == null) throw new AssertionError("No view");
        }
    };
}
```

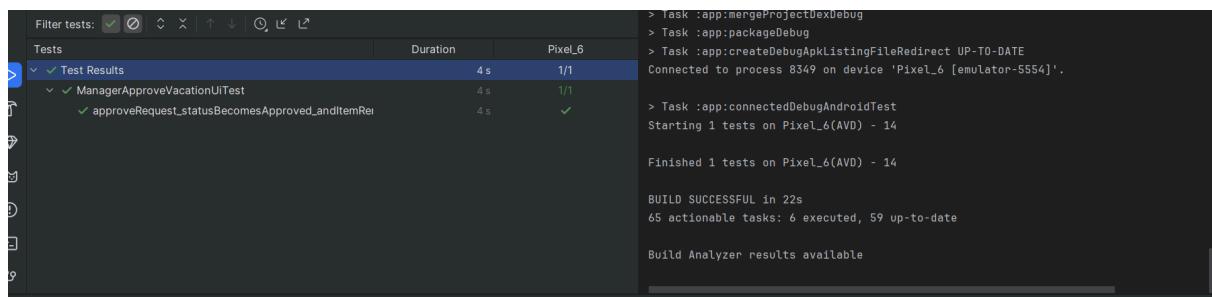
```

        v.performClick();
    }
}

private static ViewAction waitForRecyclerViewExactItemCount(int expected,
long timeoutMs) {
    return new ViewAction() {
        @Override public Matcher<View> getConstraints() { return
allOf(isDisplayed()); }
        @Override public String getDescription() { return "Wait for exact
itemCount"; }
        @Override public void perform(UiController uiController, View
view) {
            RecyclerView rv = (RecyclerView) view;
            long start = System.currentTimeMillis();
            while (true) {
                int count = rv.getAdapter() == null ? 0 :
rv.getAdapter().getItemCount();
                if (count == expected) return;
                if (System.currentTimeMillis() - start > timeoutMs)
                    throw new AssertionError("Timeout waiting for
itemCount=" + expected);
                uiController.loopMainThreadForAtLeast(250);
            }
        }
    };
}
}

```

ניתן לראות שהטסט עבר בהצלחה:



טסט זה בודק את תהליכי האישור הניהולי של בקשה חופשה. הטסט מדממת מנהל שנכנס למסך "Vacation Requests" (pending), ובוחר לאישר (Approve) את הבקשה הראשונה ברשימתה. לאחר אישור, הטסט מאשר את מצב ה- UI בדף שמספר הבקשות המתאימות ברשימה קטן (הבקשה נעלמת מסוף הבקשות המתאימות). בנוסף, הטסט מאמת מודול הננתונים (Firestore) שהסटטוס של אותה בקשה מתעדכן ל-"APPROVED", בהתאם לדרישות תהליכי העבודה במערכת.

תלוקע - code review

3. תשובות מרכזיות:

1. כרגע יש כפילות בין `getForViewer`, `sanitizeForViewer` ו-`getViewer`. עדיף לעשות פונקציה אחת שטפלת בכל המצים, ובכך אם בעתיד נרצה להוסיף מידע לפגישות שנוצרה להסתיר, זה יחסוך לזכור בכל פעם לעדכן את שני הפונקציות במקומות שונים באותו מקום אחד.
2. ב-`kickImpl` זורק שגיאות `Error` רגילות, בזמן ששאר הקוד משתמש ב-`zError`.
3. כרגע בכמה מקומות בקוד מתבצע
 - (`)ctx.db.query("meetings").collect`
ולאחר מכן `filter(...)`.
במקרים אלו הקוד מביא את כל הפגישות מהבסיס נתונים ולאחר מכן מסנן אותן. זה מבצע `FullScan` על כל הפגישות, שהוא יכול לגרום לאייתיות אם משתמשים הרבה בחיפוש שכמות הפגישות גדולה. כרגע למחרות שמתבצע אינדוקס כשמדבר ב-`zError` `listOwnedByUser`, זה לא קורא כשמחפשים פגישות לפי נושא (`topic`).
אם לפגישות אמרו להיות נושא אחד, אז אפשר לעשות אינדוקס כמו שעושים ב-`listOwnedByUser`.
אם לפגישה יכולה להיות כמה נושאים, אפשר בעתיד לבצע גרמול על מאגר המידע (קשר `Many-to-Many`) ובכך לאפשר אינדוקס לפי נושא הפגישה על ה-`collection` החדש של `meetingsXtopics`.

דוגמה לשיפור 3:

שיפור שבוצע בהתקבוס המשוב שלו: שגיאות `Error` הוחלפו ל-`zError`.

קוד ריבוי של שי קרונפלד לאוריה אוירבר סיגאוי - ת.ז 214984932 :

לbelow code review לקובץ users :

קריאה קוד: רוב השמות של הפונקציות ושל המשתנים טובים. אפשר לשפר:

- o updateBasic - שם קצר יותר. לא ברורה לבדוק מטרת הפונקציה מהשם.
- o נ - שם משתנה טוב ל- local קטן, אבל בקובץ אחרוך זה נהיה פחות קריא. עדיף user
- o במקומם נ.

הערות מרכזיות:

1. **"API Public" חושפת מידע שלא אמרו להיות ציבורי** - הפונקציה getPublicByIds מחזירה גם email וgm phone. לפי השם והכוונה ("Public"), זה בדרך כלל **במידע רגיש** שלא רוצים לחושף לכל מי שמחזיק userIds (אם זה בתוך אפליקציה, זה עדין "public shape").
הצעה: להחזיר רק שם, ביוגרפיה, אזכור זמן, topics, avatar. ואם צריך פרטן קשור אז ליצור query נפרדת עם הרשות או בדיקת חברות בקובוצה/ התאמת.

2. **שכפול לוגיקה של "מיפוי משתמש לתצוגה"** - יש כפילות בצורה בה את בונה "user response" גם ב- updateBasic וגם ב- getPublicByIds:

- חישוב avatarStorageId : getUrl ? fallback : avatarUrl
- החזרת שדות עם ?? null

הצעה: ליצור helper אחד שמייצר "UserDTO" עקי. זה ישפר קרייאות, תחזוקה ואחדות בין .endpoints

3. **דרימת שינוי אימייל: חסר קשריות נגד ניסיונות חוזרים / ספאם** - יש דברים טובים שבדקתם כמו: used, hashToken, TTL, usedRateLimit, cooldownRateLimit, cooldownDuration. מהicket "taken", מהicket בקשה קודמת.
אבל עדין אפשר לשפר ב:

- Rate limit
- ניסיון אימות אינסופי: אין attempts / lockout (אפשר לנסות קודמים עד שפוגעים).
- בדיקת אימייל בסיסית מדי (")@(")includes

הצעות:

- לשמר ב-user.emailChanges שדה attempts ולהגביל (נניח 5).
- להוסיף down - cool: לא לאפשר requestEmailChange שוב אם נשלח קוד בדקה الأخيرة.
- אימות אימייל מינימלי יותר טוב (regex פשוט) או שימוש בספרייה הצד קלינט (אם אם השרת זה Convex).

4. **לאחד ולארגן את לוגיקת שינוי האימייל לזרימה מודולרית יותר**

כרגע הלוגיקה של שינוי אימייל מפוזרת:

- ייצור בקשה (עם hash/ token)
- שליחת מייל
- אימות קוד
- עדכון משתמש

שיפור מבנה: להוציא חלקיים לפונקציות פנימיות:

- normalizeEmail(email)
- createEmailChangeRequest(...)
- validateEmailChangeRequest(...)

זה יקטין קוד כבד בתוך handler ויאפשר בדיקה והבנה קלה יותר.

שיפור שבוצע בהתקבוס המשוב שלו: פירוק פונקציות פנימיות מההערה הרבעית.

קוד רביו של רפאל קוואפיק לדניאל נזרנקו(auth.ts - תז 322719501)

Code Review Summary — convex/auth.ts

Code Readability: Variable names, function names, and documentation

Positive notes

- Function names are clear and intention-revealing (`generateCode`, `hashToken`, `sendVerificationEmail`, `signUp`, `signIn`, `verifyCode`).
- Constants like `EMAIL_VERIFICATION_WINDOW_MS` make timing rules easy to understand.
- Step-by-step comments help follow the authentication flow.

Change Request 1 — Improve naming and add lightweight documentation

- Rename vague variables to improve clarity:
 - `magic` → `magicLink`
 - `_pw` → `_passwordHash` (or avoid destructuring it at all if unused)
- Add short JSDoc comments on exported handlers (`signUp`, `signIn`, `verifyCode`) describing:
 - purpose
 - inputs
 - expected behavior and errors
- Optional: add explicit return types for exported handlers where it improves clarity.

Improving Structure: Principles like DRY (Don't Repeat Yourself) and modularity

Positive notes

- Logic is grouped by feature (sign-up, sign-in, verify code) and reads in a coherent order.
- Timing rules are already centralized using constants.

Change Request 2 — Reduce duplication with small utility helpers

- Extract repeated email normalization into a single helper:

```
function normalizeEmail(email: string): string {  
  return email.trim().toLowerCase();  
}
```

And change this:

```
const normalizedEmail = email.trim().toLowerCase();
```

With this:

```
const normalizedEmail = normalizeEmail(email);
```

- The same database query for retrieving a user by email is duplicated in `_upsertPendingUser`, `signIn`, and `verifyCode`.

Current repeated code:

```
const user = await ctx.db  
.query("users")  
.withIndex("by_email", (q) => q.eq("email", normalizedEmail))  
.unique();
```

Proposed helper:

```
async function getUserByEmail(ctx: any, email: string) {  
  return ctx.db  
.query("users")  
.withIndex("by_email", (q) => q.eq("email", email))  
.unique();  
}
```

Replace usage with:

```
const user = await getUserByEmail(ctx, normalizedEmail);
```

Why this matters

- Single source of truth for common logic.
- Easier future changes and maintenance.
- Better testability of small helpers.
- Reduced duplication across `_upsertPendingUser`, `signUp`, `signIn`, and `verifyCode`.

Bug Detection and Efficiency: Finding issues and optimizing logic

Positive notes

- Password and verification checks are ordered in a secure way.
- Verification codes are stored hashed using SHA-256, which is good security practice.
- Cooldown and expiration logic is clear and well structured.

Change Request 3 — Improve robustness of the email service

- Issue 1: No timeout for external API calls

The `fetch` call in `sendVerificationEmail` has no timeout. If the Resend API hangs, the request may block indefinitely.

Example from current code:

```
const response = await fetch("https://api.resend.com/emails", { ... });
```



Proposed improvement:

Use `AbortController` to enforce a timeout (e.g., 10 seconds).

- Issue 2: No retry logic for temporary failures

If the email API fails due to a temporary network error or an HTTP 5xx response, the system immediately throws an error without retrying.

Example from current code:

```
if (!response.ok) {
  throw new ConvexError("Failed to send verification email");
}
```



Proposed improvement: Add retry logic (2 attempts) for:

- network failures or timeouts
- HTTP 5xx responses

Never retry on 4xx client errors.

בעזרת ה `code review`, אלו השינויים שנעשו:

before

```
const existingUser = await ctx.db
  .query("users")
  .withIndex("by_email", (q) => q.eq("email", normalizedEmail))
  .unique();
```



after

```
export
ctx: QueryCtx,
email: string
) {
const normalized = normalizeEmail(email)
return await ctx.db
.query("users")
.withIndex("by_email", (q) => q.eq("email", normalized))
.unique();
}
const existingUser = getUserByEmail(ctx, normalizedEmail)
```



```

async function sendVerificationEmail(to: string, code: string) {
  const apiKey = process.env.RESEND_API_KEY;
  if (!apiKey) {
    console.error("RESEND_API_KEY is not set in Convex env");
    throw new ConvexError("Email service not configured");
  }

  const response = await fetch("https://api.resend.com/emails", {
    method: "POST",
    headers: {
      Authorization: `Bearer ${apiKey}`,
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      // For dev you can use Resend's onboarding address.
      // Later: change to something like "NoReply <noreply@yourdomain.com>",
      from: "LearnBuddies <noreply@danielnaz.com>",
      to: [to],
      subject: "Your verification code",
      html: `
        <p>Hi,</p>
        <p>Your verification code is:</p>
        <p style="font-size: 24px; font-weight: 700; letter-spacing: 0.3em;">
          ${code}
        </p>
        <p>This code expires in 3 minutes.</p>
      `,
    }),
  });

  if (!response.ok) {
    const text = await response.text();
    console.error("Resend error:", response.status, text);
    throw new ConvexError("Failed to send verification email");
  }
}

```

after:

```

sync function se    ndVerificationEmail(to: string, code: string) {
  const apiKey = process.env.RESEND_API_KEY;
  if (!apiKey) {
    console.error("RESEND_API_KEY is not set in Convex env");
    throw new ConvexError("Email service not configured");
  }

  const timeoutMs = 10_000;
  const controller = new AbortController();
  const timeoutId = setTimeout(() => controller.abort(), timeoutMs);

  try {
    const response = await fetch("https://api.resend.com/emails", {
      method: "POST",
      headers: {
        Authorization: `Bearer ${apiKey}`,
        "Content-Type": "application/json",
      },
      signal: controller.signal,
      body: JSON.stringify({
        from: "LearnBuddies <noreply@danielnaz.com>",
        to: [to],
        subject: "Your verification code",
        html: `
          <p>Hi,</p>
          <p>Your verification code is:</p>
          <p style="font-size: 24px; font-weight: 700; letter-spacing: 0.3em;">
            ${code}
          </p>
          <p>This code expires in 3 minutes.</p>
        `,
      }),
    });
  }

  if (!response.ok) {
    const text = await response.text().catch(() => "");
    console.error("Resend error:", response.status, text);
    throw new ConvexError("Failed to send verification email");
  }
} catch (err: any) {
  if (err?.name === "AbortError") {
    console.error(`Resend timeout after ${timeoutMs}ms`);
    throw new ConvexError("Email request timed out");
  }
  throw err;
} finally {
  clearTimeout(timeoutId);
}
}

```

before:

```

const normalizedEmail = email.trim().toLowerCase();

```

after:

```

function normalizeEmail(email: string) {
  return email.trim().toLowerCase();
}

const normalizedEmail = normalizeEmail(email)

```

