Laporan Tugas Kualitas Perangkat Lunak

TUGAS JUNIT

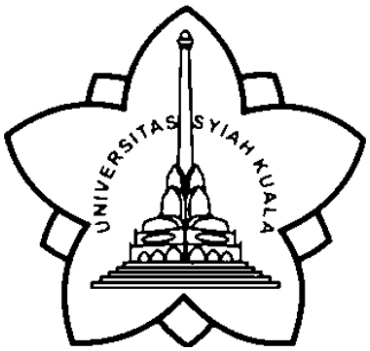
disusun untuk memenuhi

tugas mata kuliah Kualitas Perangkat Lunak

oleh:

AULIA VIKA RAHMAN

2208107010001



JURUSAN INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

DARUSSALAM, BANDA ACEH

2025

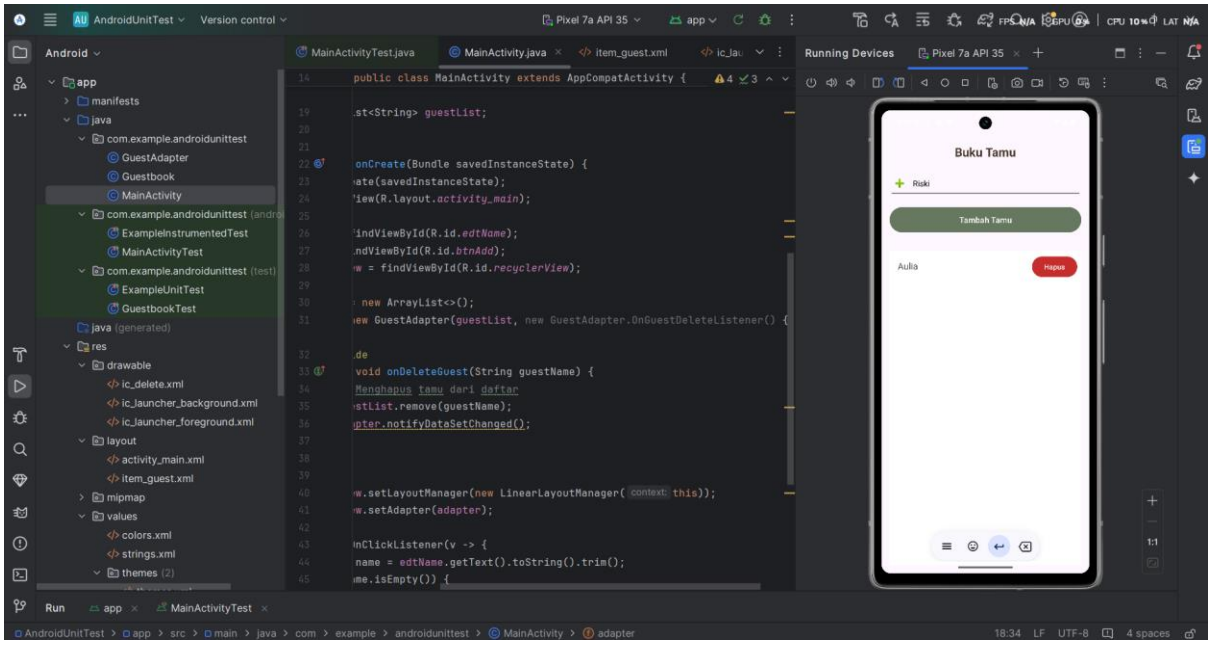
LINK GITHUB : https://github.com/auliavika/JUNIT.git

Deskripsi Proyek

Pada tugas ini, saya membuat sebuah proyek Android sederhana yang berfungsi sebagai aplikasi **Guestbook (Buku Tamu)**. Aplikasi ini memungkinkan pengguna untuk menambahkan dan menghapus tamu melalui antarmuka pengguna yang disediakan.

Aplikasi ini memiliki dua fitur utama:

1. **Menambahkan Tamu**: Pengguna dapat menambahkan nama tamu melalui kolom input dan menekan tombol "Tambah Tamu". Nama tamu yang ditambahkan akan muncul dalam daftar tamu.
2. **Menghapus Tamu**: Pengguna dapat menghapus tamu yang sudah ditambahkan. Setiap item dalam daftar tamu memiliki tombol "Hapus" yang memungkinkan pengguna untuk menghapus tamu dari daftar.



Gambar 1. Tampilan UI dan Struktur Folder Code Aplikasi

Aplikasi ini mengimplementasikan penggunaan **RecyclerView** untuk menampilkan daftar tamu, serta menggunakan **Button** untuk menambahkan dan menghapus tamu. Untuk pengujian aplikasi, saya menggunakan dua jenis pengujian:

- **Unit Test (Local Test)**: Pengujian dilakukan pada logika aplikasi tanpa berinteraksi dengan antarmuka pengguna.
- **Instrumented Test**: Pengujian dilakukan untuk memverifikasi interaksi aplikasi dengan antarmuka pengguna (UI) di perangkat Android.

Test Case yang Dibuat

Dalam proyek ini, saya telah membuat membuat empat local test dan empat instrumental test untuk memastikan bahwa aplikasi berfungsi dengan baik sesuai dengan kebutuhan yang diinginkan.

1. Local Test
   Local test atau unit test adalah jenis pengujian yang dilakukan di level unit terkecil dari kode program, biasanya berupa metode atau fungsi. Unit test bertujuan untuk menguji fungsionalitas dari bagian-bagian kecil aplikasi, seperti metode atau kelas, secara independen dari bagian lain dalam aplikasi.

   1.1. tambahTamu_namaValid_tamuBerhasilDitambahkan()

```
@Test
public void tambahTamu_namaValid_tamuBerhasilDitambahkan() {
    /*
     * Test ini memverifikasi bahwa jika nama tamu yang valid ditambahkan,
     * metode addGuest() akan mengembalikan nilai true (berhasil menambahkan tamu).
     */
    assertTrue(guestbook.addGuest( name: "Aulia"));
}
```

Gambar 2. Test Case 1, tambah Tamu_namaValid_tamuBerhasilDitambahkan

Tujuan: Menguji apakah tamu yang ditambahkan dengan nama valid akan berhasil ditambahkan ke dalam guestbook.
Hasil yang Diharapkan: Tamunya berhasil ditambahkan.

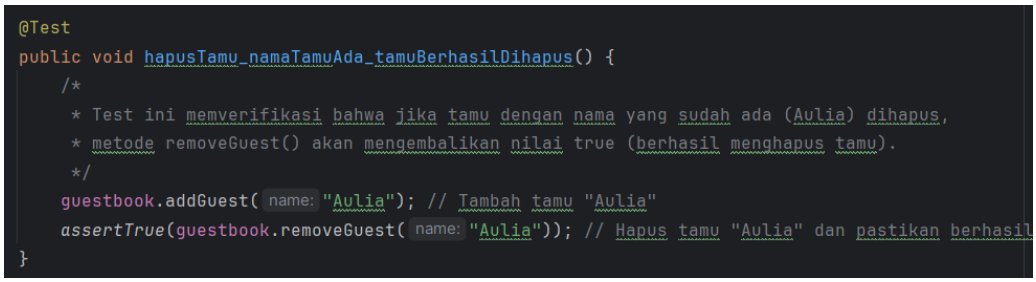
   1.2. tambahTamu_namaKosong_tamuGagalDitambahkan()

```
@Test
public void tambahTamu_namaValid_tamuBerhasilDitambahkan() {
    /*
     * Test ini memverifikasi bahwa jika nama tamu yang valid ditambahkan,
     * metode addGuest() akan mengembalikan nilai true (berhasil menambahkan tamu).
     */
    assertTrue(guestbook.addGuest( name: "Aulia"));
}
```

Gambar 3. Test Case 2, tambahTamu_namaKosong_tamuGagalDitambahkan()

Tujuan: Menguji apakah tamu dengan nama kosong tidak dapat ditambahkan ke dalam guestbook.
Hasil yang Diharapkan: Tamunya tidak dapat ditambahkan.

1.3. hapusTamu_namaTamuAda_tamuBerhasilDihapus()



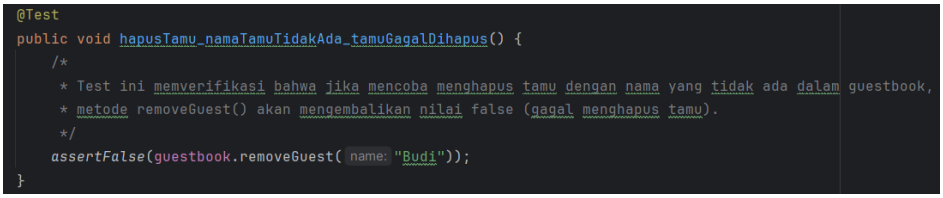
```
@Test
public void hapusTamu_namaTamuAda_tamuBerhasilDihapus() {
    /*
     * Test ini memverifikasi bahwa jika tamu dengan nama yang sudah ada (Aulia) dihapus,
     * metode removeGuest() akan mengembalikan nilai true (berhasil menghapus tamu).
     */
    guestbook.addGuest( name: "Aulia"); // Tambah tamu "Aulia"
    assertTrue(guestbook.removeGuest( name: "Aulia")); // Hapus tamu "Aulia" dan pastikan berhasil
}
```

Gambar 4. Test Case 3, hapusTamu_namaTamuAda_tamuBerhasilDihapus

Tujuan: Menguji apakah tamu yang ada di guestbook dapat dihapus dengan benar.

Hasil yang Diharapkan: Tamunya berhasil dihapus

1.4.hapusTamu_namaTamuTidakAda_tamuGagalDihapus()



```
@Test
public void hapusTamu_namaTamuTidakAda_tamuGagalDihapus() {
    /*
     * Test ini memverifikasi bahwa jika mencoba menghapus tamu dengan nama yang tidak ada dalam guestbook,
     * metode removeGuest() akan mengembalikan nilai false (gagal menghapus tamu).
     */
    assertFalse(guestbook.removeGuest( name: "Budi"));
}
```

Gambar 5. Test Case 4, hapusTamu_namaTamuTidakAda_tamuGagalDihapus()

Tujuan: Menguji apakah tamu yang tidak ada dalam guestbook gagal dihapus.
Hasil yang Diharapkan: Tamunya gagal dihapus.

2. Instrumented Test

Instrumented test adalah jenis pengujian yang dilakukan pada aplikasi Android dengan menjalankan aplikasi tersebut di perangkat fisik atau emulator. Tes ini menguji interaksi antara aplikasi dan sistem Android (seperti UI dan komponen aplikasi), termasuk pengujian alur interaksi pengguna seperti menekan tombol, memasukkan teks, dan memeriksa apakah UI merespons sesuai yang diinginkan.

2.1. tambahTamu_tamuTampilDiRecyclerView()

```
@Test
public void tambahTamu_tamuTampilDiRecyclerView() {
    /*
     * Test ini memverifikasi bahwa setelah tamu baru (Aulia) ditambahkan,
     * tamu tersebut muncul dalam RecyclerView.
     */
    // Masukkan nama tamu
    onView(withId(R.id.edtName)).perform(replaceText( stringToBeSet: "Aulia"));

    // Klik tombol Tambah
    onView(withId(R.id.btnAdd)).perform(click());

    // Pastikan nama "Aulia" muncul di RecyclerView
    onView(withText("Aulia")).check(matches(withText("Aulia")));
}
```

Gambar 6. Test Case 1, tambahTamu_tamuTampilDiRecyclerView()

Tujuan: Menguji apakah tamu yang baru ditambahkan akan muncul di UI, yaitu RecyclerView.
Hasil yang Diharapkan: Nama tamu yang ditambahkan muncul di RecyclerView.

2.2. hapusTamu_namaTamuHilangDariRecyclerView()

```
@Test
public void hapusTamu_namaTamuHilangDariRecyclerView() {
    /*
     * Test ini memverifikasi bahwa jika tamu yang ada (Aulia) dihapus,
     * tamu tersebut benar-benar hilang dari RecyclerView.
     */
    // Tambah tamu "Aulia"
    onView(withId(R.id.edtName)).perform(replaceText( stringToBeSet: "Aulia"));
    onView(withId(R.id.btnAdd)).perform(click());

    // Pastikan tamu "Aulia" muncul di RecyclerView
    onView(withText("Aulia")).check(matches(withText("Aulia")));

    // Klik tombol Hapus untuk "Aulia"
    onView(withText("Aulia")).perform(scrollTo(), click());

    // Klik tombol Hapus yang ada di dalam item guest
    onView(withId(R.id.btnDelete)).perform(click());

    // Pastikan nama "Aulia" sudah tidak ada di RecyclerView
    onView(withText("Aulia")).check(doesNotExist());
}
```

Gambar 7. Test Case 2, hapusTamu_namaTamuHilangDariRecyclerView()

Tujuan: Menguji apakah tamu yang dihapus akan hilang dari RecyclerView.
Hasil yang Diharapkan: Nama tamu yang dihapus hilang dari RecyclerView.

2.3.tambahTamu_namaKosong_tamuTidakDitambahkan()

```
@Test
public void tambahTamu_namaKosong_tamuTidakDitambahkan() {
    /*
     * Test ini memverifikasi bahwa jika nama tamu kosong dimasukkan,
     * tamu tidak akan ditambahkan ke RecyclerView.
     */
    // Masukkan nama kosong
    onView(withId(R.id.edtName)).perform(replaceText( stringToBeSet: ""));

    // Klik tombol Tambah
    onView(withId(R.id.btnAdd)).perform(click());

    // Pastikan RecyclerView masih kosong
    onView(withId(R.id.recyclerView)).check(matches(hasChildCount(0)));
}
```

Gambar 8, Test Case 3, tambahTamu_namaKosong_tamuTidakDitambahkan()

Tujuan: Menguji apakah tamu dengan nama kosong tidak dapat ditambahkan ke RecyclerView.

Hasil yang Diharapkan: RecyclerView tetap kosong.

2.4. tambahDanHapusTamu_urutannyaAcak()

```
@Test
public void tambahDanHapusTamu_urutannyaAcak() {
    /*
     * Test ini memverifikasi bahwa tamu ditambahkan dan dihapus dalam urutan yang benar,
     * meskipun penghapusan dilakukan dalam urutan yang tidak berurutan.
     */
    // Tambah tamu "Aulia"
    onView(withId(R.id.edtName)).perform(replaceText( stringToBeSet: "Aulia"));
    onView(withId(R.id.btnAdd)).perform(click());

    // Tambah tamu "Budi"
    onView(withId(R.id.edtName)).perform(replaceText( stringToBeSet: "Budi"));
    onView(withId(R.id.btnAdd)).perform(click());

    // Tambah tamu "Riski"
    onView(withId(R.id.edtName)).perform(replaceText( stringToBeSet: "Riski"));
    onView(withId(R.id.btnAdd)).perform(click());

    // Pastikan tamu "Aulia", "Budi", dan "Riski" muncul di RecyclerView
    onView(withText("Aulia")).check(matches(isDisplayed()));
    onView(withText("Budi")).check(matches(isDisplayed()));
    onView(withText("Riski")).check(matches(isDisplayed()));

    // Hapus tamu "Budi" (posisi 1)
    onView(withId(R.id.recyclerView))
            .perform(RecyclerViewActions.actionOnItemAtPosition( position: 1, clickChildViewWithId(R.id.btnDelete)));
```

```
    // Pastikan "Budi" sudah tidak ada di RecyclerView
    onView(withText("Budi")).check(doesNotExist());

    // Hapus tamu "Riski" (posisi 1 setelah "Budi" dihapus)
    onView(withId(R.id.recyclerView))
            .perform(RecyclerViewActions.actionOnItemAtPosition( position: 1, clickChildViewWithId(R.id.btnDelete)));

    // Pastikan "Riski" sudah tidak ada di RecyclerView
    onView(withText("Riski")).check(doesNotExist());

    // Hapus tamu "Aulia" (posisi 0 setelah "Budi" dan "Riski" dihapus)
    onView(withId(R.id.recyclerView))
            .perform(RecyclerViewActions.actionOnItemAtPosition( position: 0, clickChildViewWithId(R.id.btnDelete)));

    // Pastikan "Aulia" sudah tidak ada di RecyclerView
    onView(withText("Aulia")).check(doesNotExist());
}
```

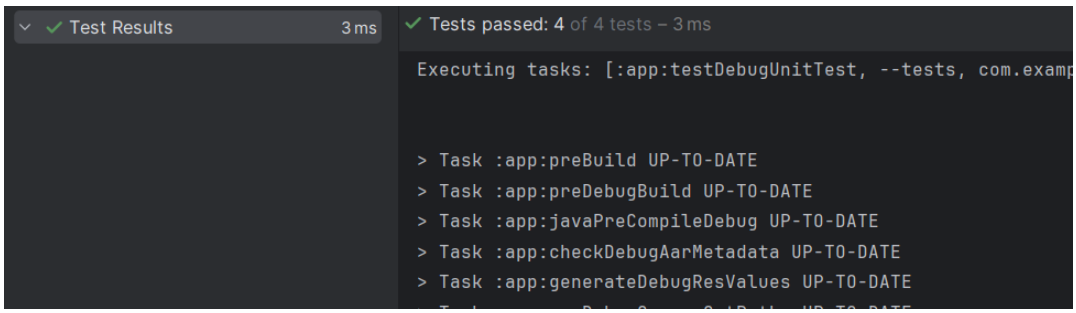
Gambar 9, Test Case 4, tambahDanHapusTamu_urutannyaAcak()

Tujuan: Menguji apakah tamu ditambahkan dan dihapus dalam urutan yang benar meskipun penghapusan dilakukan dalam urutan yang tidak berurutan.

Hasil yang Diharapkan: Tamu yang dihapus hilang dari RecyclerView meskipun urutan penghapusannya tidak berurutan.

Hasil

1. Local Test



Gambar 10. Hasil Local Test

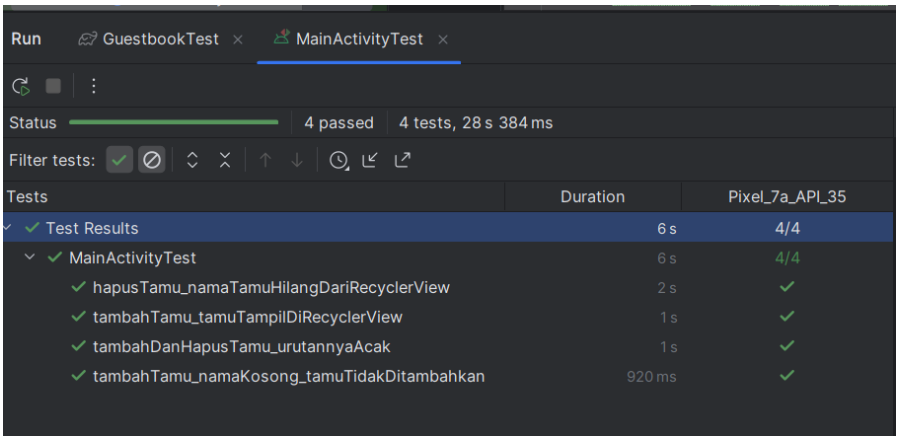
- **Test: tambahTamu_namaValid_tamuBerhasilDitambahkan**
  Hasil: Pengujian berhasil. Tamu dengan nama valid ditambahkan ke dalam guestbook. Metode addGuest() mengembalikan nilai true, yang sesuai dengan harapan.

- **Test: tambahTamu_namaKosong_tamuGagalDitambahkan**
  Hasil: Pengujian berhasil. Ketika nama kosong dimasukkan, metode addGuest() mengembalikan nilai false, yang berarti tamu gagal ditambahkan.

- **Test: hapusTamu_namaTamuAda_tamuBerhasilDihapus**
  Hasil: Pengujian berhasil. Tamu yang ada dalam guestbook berhasil dihapus dengan menggunakan metode removeGuest() dan mengembalikan nilai true.

- **Test: hapusTamu_namaTamuTidakAda_tamuGagalDihapus**
  Hasil: Pengujian berhasil. Ketika mencoba menghapus tamu yang tidak ada dalam guestbook, metode removeGuest() mengembalikan nilai false, yang menunjukkan bahwa penghapusan gagal.

2. Instrumented Test



Gambar 11. Hasil Instrumented Test

- **Test: tambahTamu_tamuTampilDiRecyclerView**
  Hasil: Pengujian berhasil. Setelah menambahkan tamu dengan nama "Aulia", tamu tersebut muncul dalam RecyclerView.

- **Test: hapusTamu_namaTamuHilangDariRecyclerView**
  Hasil: Pengujian berhasil. Setelah menambahkan tamu "Aulia" dan menghapusnya, tamu tersebut hilang dari RecyclerView sesuai yang diharapkan.

- **Test: tambahTamu_namaKosong_tamuTidakDitambahkan**
  Hasil: Pengujian berhasil. Ketika nama kosong dimasukkan, tamu tidak ditambahkan ke dalam RecyclerView, dan daftar tamu tetap kosong.

- **Test: tambahDanHapusTamu_urutannyaAcak**
  Hasil: Pengujian berhasil. Tamu ditambahkan dan dihapus dalam urutan yang tidak berurutan, namun aplikasi masih mengelola urutan tamu dengan benar. Tamu yang dihapus hilang dari RecyclerView sesuai urutan yang benar setelah penghapusan.

Kesimpulan

Semua pengujian, baik unit test maupun instrumented test, berhasil dijalankan dengan hasil yang sesuai harapan. Aplikasi berfungsi dengan baik dalam menambahkan dan menghapus tamu, serta menampilkan dan mengelola daftar tamu dalam RecyclerView. Dengan demikian, fungsionalitas aplikasi telah terverifikasi dan berjalan sesuai dengan spesifikasi yang diinginkan.