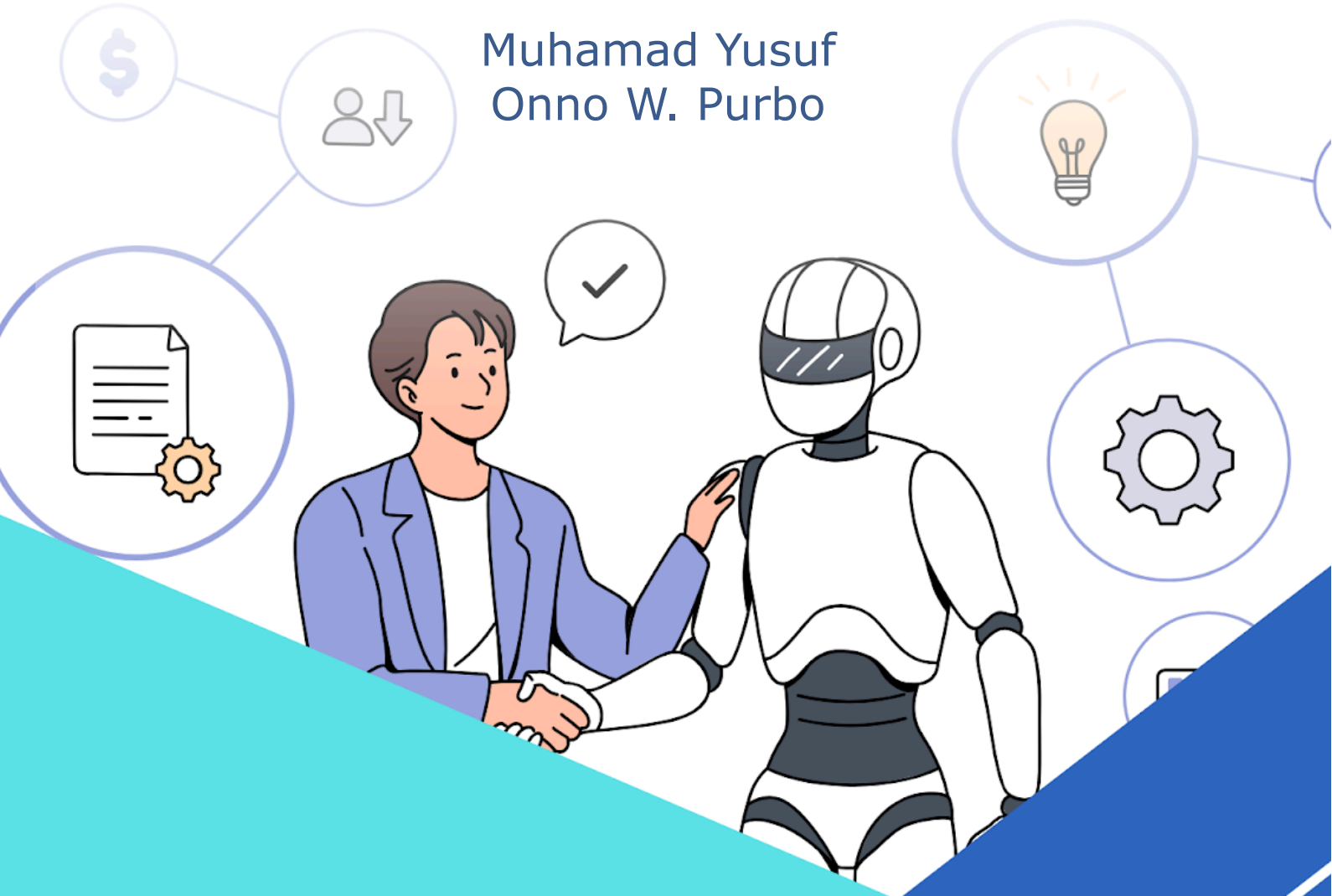
ARTIFICIAL INTELLIGENCE

Ayo Coding! Serunya Bikin Aplikasi Android di SMP

UNTUK SMP KELAS VII SEMESTER 2

Muhamad Yusuf
Onno W. Purbo



Buku AI SMP Kelas 7 Semester 2

Ayo Coding! Serunya Bikin Aplikasi Android di SMP

Jumlah Pertemuan: 14

Muhamad Yusuf
Onno W. Purbo

Institut Teknologi Tangerang Selatan (ITTS)
2025

Daftar Isi

Daftar Isi **3**
Lisensi & Catatan Karya **6**
Disclaimer **6**
Kata Pengantar **7**
BAB 1: Pengenalan Lingkungan App Inventor 2 8
Tujuan Pembelajaran 8
Peta Konsep 8
Apersepsi 8
Penjelasan Konsep (Teori) 9
Pengenalan App Inventor 2: Apa itu App Inventor 2? 9
Komponen Utama App Inventor 2 9
Proses Dasar Pengembangan Aplikasi 11
Contoh Kasus atau Ilustrasi 11
Contoh Soal dan Pembahasan 17
Fakta Menarik / Fun Fact 19
Tips Belajar atau Tips Cepat Menghafal 20
Aktivitas Siswa 21
Rangkuman 23
Latihan Soal 23
Tugas Proyek 26
BAB 2: Membuat Aplikasi "Hello World" Interaktif 29
Tujuan Pembelajaran 29
Peta Konsep 29
Apersepsi 29
Penjelasan Konsep (Teori) 30
Event Handling (Menangani Kejadian) 30
Blok when Button1.Click do 30
Blok set Label1.Text to 30
String (Teks) 31
Membuat Aplikasi "Hello World" Dasar 31
Contoh Kasus atau Ilustrasi 32
Contoh Soal dan Pembahasan 38
Fakta Menarik / Fun Facts 41
Tips Belajar atau Tips Cepat Menghafal 41
Aktivitas Siswa 42
Rangkuman 43
Latihan Soal 44
Tugas Proyek 46
BAB 3: Logika Kondisi (If-Then-Else) dan Variabel Dasar 50
Tujuan Pembelajaran 50
Peta Konsep 50
Apersepsi 50

Penjelasan Konsep (Teori)..... 51
Apa Itu Logika Kondisi?.....51
Operator Perbandingan.....52
Apa Itu Variabel?..... 52
Komponen TextBox..... 53
Konversi Tipe Data Secara Otomatis (Implisit).....53
Kesimpulan.....53
Contoh Kasus atau Ilustrasi..... 53
Contoh Soal dan Pembahasan..... 56
Fakta Menarik / Fun Facts.....58
Tips Belajar atau Tips Cepat Menghafal..... 58
Aktivitas Siswa..... 59
Rangkuman.....62
Latihan Soal..... 62
Tugas Proyek..... 65
BAB 4: Interaksi Antar Layar dan Suara..... 68
Tujuan Pembelajaran..... 68
Peta Konsep.....68
Apersepsi..... 68
Penjelasan Konsep (Teori)..... 68
Apa Itu Aplikasi Multi-Layar?.....68
Contoh Kasus atau Ilustrasi..... 71
Contoh Soal dan Pembahasan..... 72
Fakta Menarik / Fun Facts.....75
Tips Belajar atau Tips Cepat Menghafal..... 75
Aktivitas Siswa..... 75
Rangkuman.....78
Latihan Soal..... 78
Tugas Proyek..... 81
BAB 5: Game Sederhana: "Catch the Dot" atau "Whack-a-Mole"..... 84
Tujuan Pembelajaran..... 84
Peta Konsep.....84
Apersepsi..... 84
Penjelasan Konsep (Teori)..... 84
Apa Saja yang Dibutuhkan untuk Membuat Game Interaktif?..... 85
Contoh Kasus atau Ilustrasi..... 88
Contoh Soal dan Pembahasan..... 90
Fakta Menarik / Fun Facts.....92
Tips Belajar atau Tips Cepat Menghafal..... 92
Aktivitas Siswa..... 92
Rangkuman.....95
Latihan Soal..... 95
Tugas Proyek..... 98
BAB 6: Refleksi: Belajar Logika = Belajar Berpikir..... 101

Tujuan Pembelajaran..... 101
Peta Konsep.....101
Apersepsi..... 101
Penjelasan Konsep (Teori)..... 101
Hubungan App Inventor 2 dengan Computational Thinking.....102
Pentingnya Debugging..... 102
Aplikasi dalam Kehidupan Sehari-hari..... 103
Tantangan untuk Kalian.....103
Contoh Kasus atau Ilustrasi..... 103
Contoh Soal dan Pembahasan..... 106
Fakta Menarik / Fun Facts.....109
Tips Belajar atau Tips Cepat Menghafal..... 109
Aktivitas Siswa.....110
Rangkuman..... 112
Latihan Soal.....112
Tugas Proyek.....115
BAB 7: Mengenal Domain .id..... 118
Apa Itu Domain?.....119
Apa Itu .id?..... 119
Keunggulan domain .id.....119
Siapa Itu PANDI?..... 120
Kenapa Kamu Harus Tahu Ini?..... 120
Lebih dalam tentang PANDI.....120
Tata Kelola Internet..... 122
Institut Teknologi Tangerang Selatan (ITTS) dan Penulis..... 123

Lisensi & Catatan Karya

Karya ini dilisensikan di bawah lisensi **Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)**.
Artinya, **siapapun bebas untuk menggunakan, menyalin, membagikan, dan mengadaptasi** materi ini, **dengan syarat**:

- **Memberikan atribusi yang sesuai** (menyebut sumber asli),
- **Menyertakan lisensi yang sama** jika dimodifikasi atau dikembangkan lebih lanjut.

Lihat detail lisensi di sini https://creativecommons.org/licenses/by-nd/4.0/legalcode.id

Desain Cover: Irwan Siswanto

Disclaimer

Materi pembelajaran ini dibuat dengan **dana swadaya masyarakat Indonesia** dan **kontribusi sukarela dari para dosen di Institut Teknologi Tangerang Selatan (ITTS)**. Karya ini didistribusikan secara bebas untuk mendukung pendidikan digital dan kecerdasan buatan di kalangan pelajar Indonesia.

Kami **memohon maaf** jika terdapat kekurangan dalam isi maupun penyajian materi ini. Kritik dan saran sangat kami harapkan untuk terus menyempurnakan karya serupa di masa depan.

Mohon doa agar amal ibadah mereka yang terlibat di kegiatan ini diterima dan dilipatgandakan.

Kata Pengantar

Buku ini disusun sebagai bentuk partisipasi sederhana dari sivitas akademika Institut Teknologi Tangerang Selatan (ITTS) dalam mendukung kemajuan pendidikan di Indonesia. Di tengah pesatnya perkembangan teknologi, kami terdorong untuk menghadirkan materi pembelajaran yang relevan dan dapat diakses secara luas oleh masyarakat, khususnya generasi muda.

Dengan semangat berbagi dan memberdayakan, kami berharap buku ini dapat menjadi langkah awal yang bermakna dalam memperkenalkan konsep dasar kecerdasan buatan (AI) kepada pelajar tingkat SMP dan SMA. Materi disusun secara sistematis dan aplikatif, mencakup antara lain:

- Dasar-dasar *Computational Thinking*
- Pembuatan aplikasi Android menggunakan App Inventor 2
- Pemrograman visual dengan Scratch
- Pengenalan AI melalui Google Teachable Machine
- Eksplorasi AI generatif seperti ChatGPT, Gemini, dan Grok
- Pembelajaran AI berbasis data dengan Orange Data Mining, disertai contoh yang sederhana

Penyusunan dan penerbitan buku ini sepenuhnya dibiayai secara mandiri berkat dukungan berbagai pihak yang memiliki kepedulian terhadap kemajuan ilmu pengetahuan di tanah air. Untuk itu, kami mengucapkan terima kasih yang tulus atas segala bantuan dan dorongan yang telah diberikan.

Kami menyadari bahwa karya ini masih jauh dari sempurna. Karena itu, segala bentuk kritik, saran, maupun masukan yang membangun sangat kami hargai dan nantikan. Silakan sampaikan melalui kontak resmi ITTS atau langsung kepada tim penulis. Pembaruan dan perbaikan akan terus kami lakukan demi meningkatkan kualitas isi dan manfaat buku ini.

Semoga buku ini dapat menjadi pijakan awal bagi para pelajar dalam memahami dunia kecerdasan buatan, serta turut menginspirasi langkah kecil menuju masa depan Indonesia yang lebih cerah.

Dengan rendah hati, kami memohon doa dan dukungan dari para pembaca agar upaya sederhana ini dapat membawa manfaat yang luas dan menjadi amal jariyah bagi semua penulis yang terlibat.

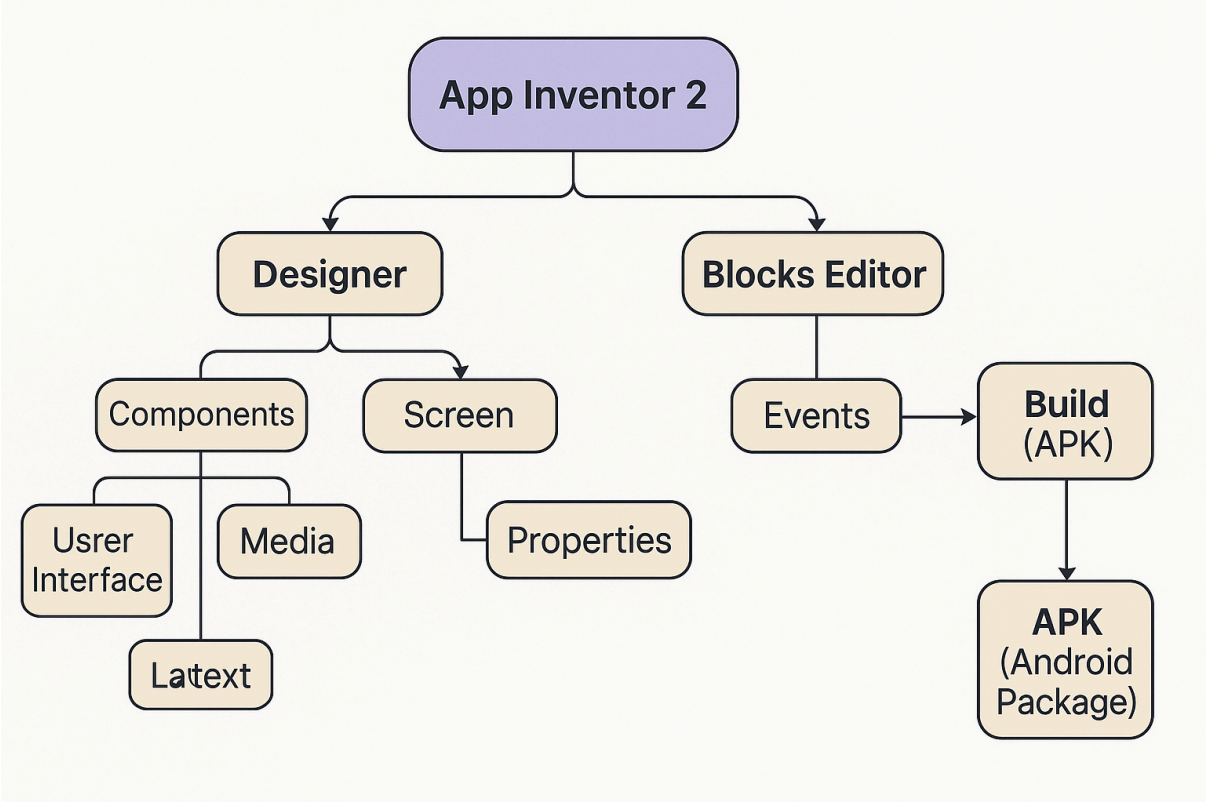
Tangerang Selatan, Agustus 2025
Penulis

BAB 1: Pengenalan Lingkungan App Inventor 2

Tujuan Pembelajaran

Peserta didik mampu mengenal dan menggunakan antarmuka App Inventor 2 untuk membuat aplikasi sederhana.

Peta Konsep



Apersepsi

Pernah tidak kamu bayangin bisa bikin aplikasi Android sendiri?
Bukan cuma main game atau install aplikasi orang lain, tapi jadi pencipta aplikasi yang bisa kamu pakai sendiri, bahkan dibagikan ke teman-temanmu?

Nah, sekarang kamu bisa!
Dengan MIT App Inventor 2, kamu akan belajar membuat aplikasi Android dengan cara yang simpel dan seru, tanpa harus bisa coding rumit.
Kita cukup menyusun balok warna-warni seperti main puzzle.
Siap tantangan seru jadi App Creator hari ini? Ayo kita mulai!

Penjelasan Konsep (Teori)

Pengenalan App Inventor 2: Apa itu App Inventor 2?

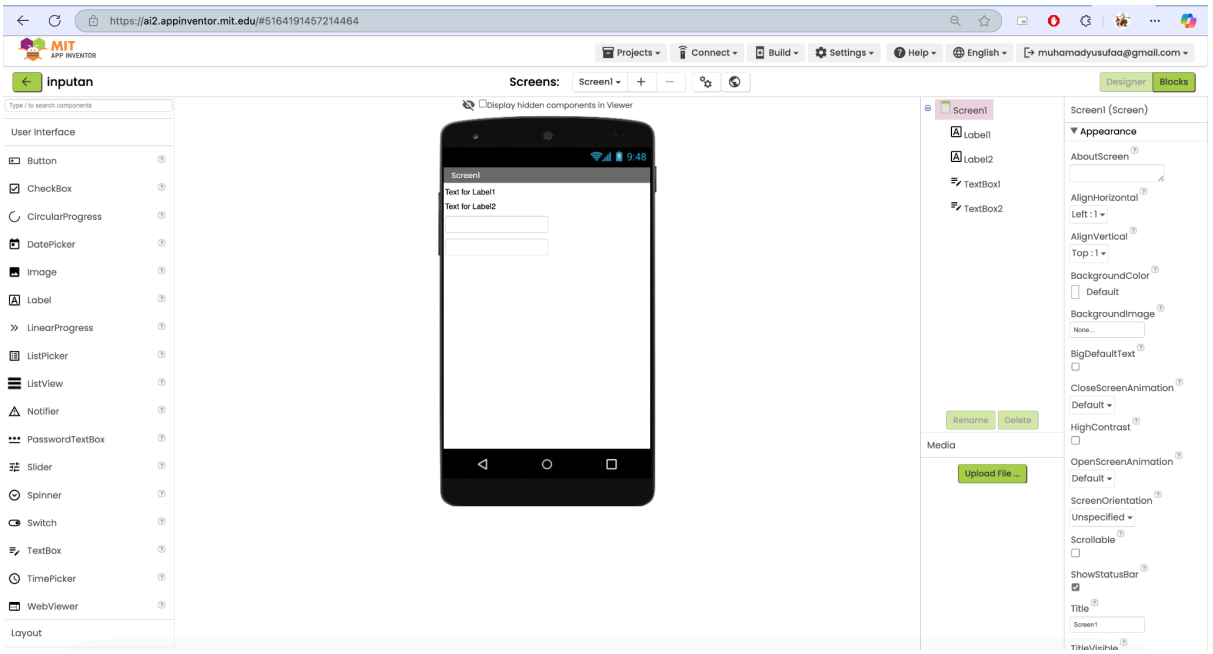
Pernahkah kamu berpikir untuk membuat aplikasi sendiri untuk HP Android?

Dengan MIT App Inventor 2, kamu bisa! App Inventor adalah tool keren dari MIT yang memungkinkan kita membuat aplikasi sungguhan tanpa perlu mengetik kode yang rumit. Kamu bisa berubah dari yang tadinya hanya pengguna aplikasi, menjadi seorang pencipta aplikasi.

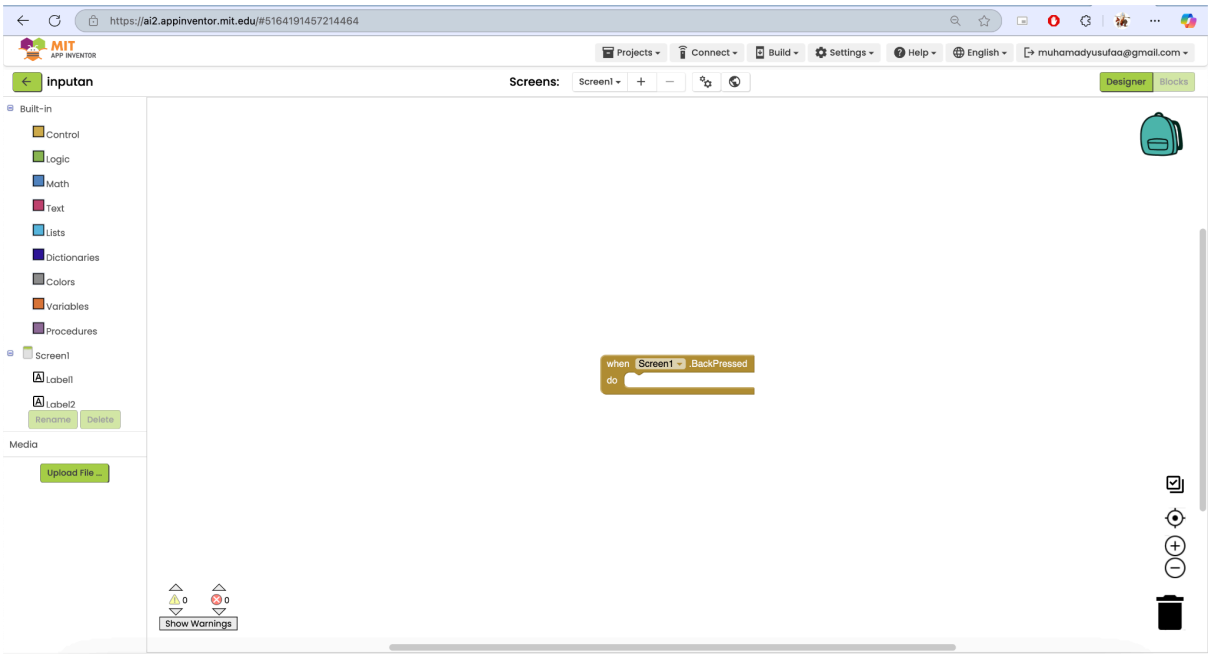
Caranya sangat mudah. Daripada mengetik kode, kamu hanya perlu menyusun balok-balok berwarna seperti sedang **bermain puzzle**. Setiap balok adalah sebuah perintah. Dengan cara ini, kamu bisa fokus pada ide kreatifmu untuk membuat aplikasi, bukan pusing dengan tulisan kode yang sulit. Belajar membuat aplikasi jadi terasa seperti bermain, dan kamu bahkan bisa membuat aplikasi pertamamu dalam waktu singkat!

Komponen Utama App Inventor 2

Untuk membuat aplikasi, kita akan bekerja di dua tempat utama: **Designer View** dan **Blocks Editor**. Keduanya punya tugas berbeda tapi saling berhubungan.



- **Designer View (Halaman Desain):** Anggap saja ini adalah tempat kamu **menghias** tampilan aplikasi. Di sinilah kamu menentukan semua yang akan dilihat oleh pengguna. Di dalam halaman ini, ada beberapa bagian penting:
    - **Components Palette (Palet Komponen):** Ini seperti kotak mainanmu. Isinya ada banyak sekali "barang" yang bisa kamu pasang di aplikasi, misalnya Tombol (Button), Tulisan (Label), dan Gambar (Image).
    - **Viewer (Layar Tampilan):** Ini adalah tiruan layar HP-mu. Kamu akan menyeret "barang" dari Palet Komponen kesini untuk menata tampilannya.
    - **Components List (Daftar Komponen):** Bagian ini menunjukkan daftar semua "barang" yang sudah kamu pakai di aplikasimu. Kamu bisa memilih, mengganti nama, atau menghapusnya dari sini.
    - **Properties Panel (Panel Pengaturan):** Kalau kamu memilih salah satu "barang" (misalnya sebuah tombol), di sini kamu bisa mengubah pengaturannya. Mau ganti warna, ukuran, atau tulisannya? Semua bisa diatur di sini.
    - **Media:** Ini adalah tempat untuk meng-upload file gambar atau suara dari komputer kamu untuk dipakai di dalam aplikasi.



- **Blocks Editor (Halaman Blok Perintah):** Jika di Designer kamu menghias "wajah" aplikasi, maka di Blocks Editor kamu akan mengatur "otak"-nya. Di sinilah kamu **memberi perintah**. Kamu akan menentukan apa yang terjadi kalau sebuah tombol ditekan. Caranya adalah dengan menyambungkan balok-balok perintah seperti menyusun puzzle. Setiap "barang" yang kamu pasang di Designer punya balok perintahnya sendiri di sini.

Proses Dasar Pengembangan Aplikasi

Membuat aplikasi dengan App Inventor pada dasarnya hanya mengulang tiga langkah mudah ini :

1. **Desain Tampilan (di Designer):** Pertama, kamu rancang dulu bagaimana tampilan aplikasi nanti. Ambil komponen dari *Palette*, tata di *Viewer*, lalu atur warnanya di *Properties*.
2. **Atur Logika (di Blocks Editor):** Setelah tampilannya bagus, pindah ke *Blocks Editor*. Di sini kamu susun balok-balok perintahnya. Misalnya: "Kalau Tombol1 ditekan, maka mainkan Suara1".
3. **Uji Coba Aplikasi (Testing):** Bagian paling seru! Kamu bisa langsung mencoba aplikasi di HP Android. Cukup hubungkan HP-mu menggunakan aplikasi **MIT AI2 Companion**. Semua perubahan yang kamu buat akan langsung terlihat di HP. Jadi, kalau ada yang salah, kamu bisa langsung memperbaikinya.

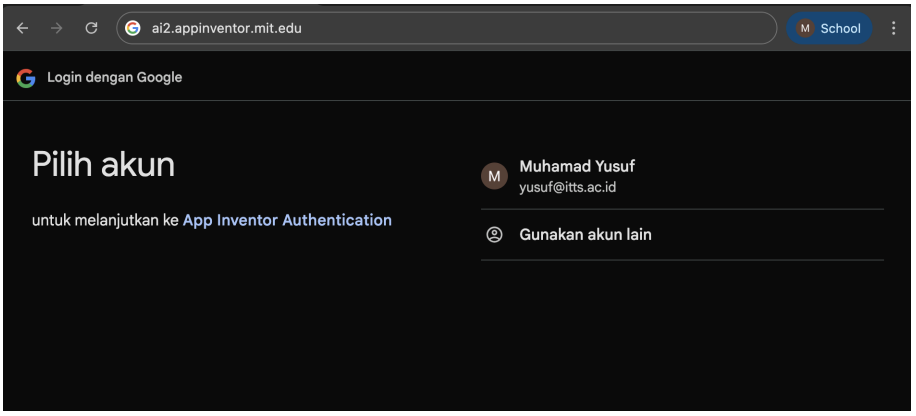
Contoh Kasus atau Ilustrasi

Mari kita coba langsung membuat aplikasi pertama kita yang sangat sederhana. Ikuti langkah-langkah berikut ini untuk berkenalan langsung dengan App Inventor.

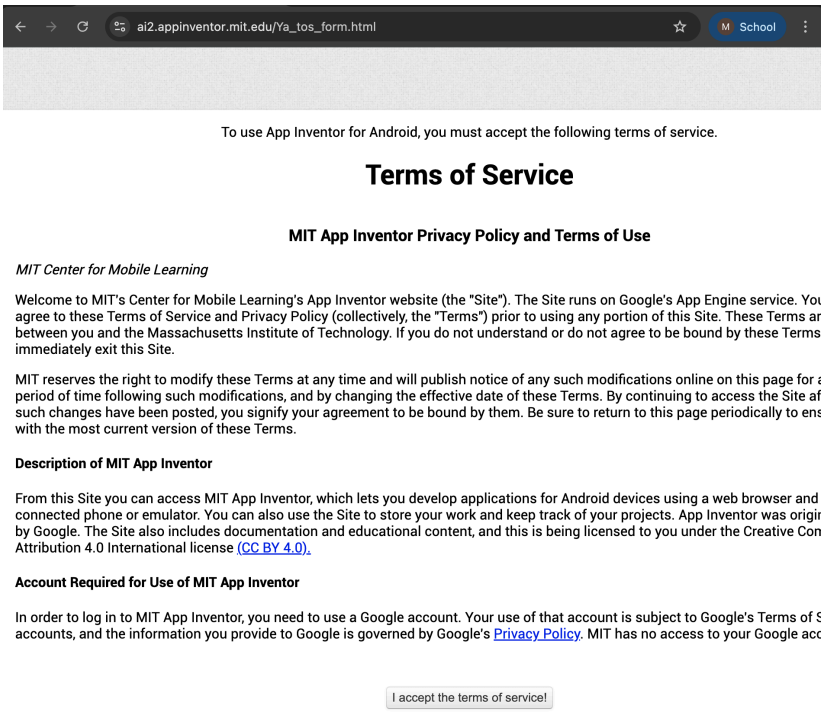
Kasus 1: Membuka App Inventor 2

Langkah pertama adalah membuka "pintu" menuju studio pembuatan aplikasi kita.

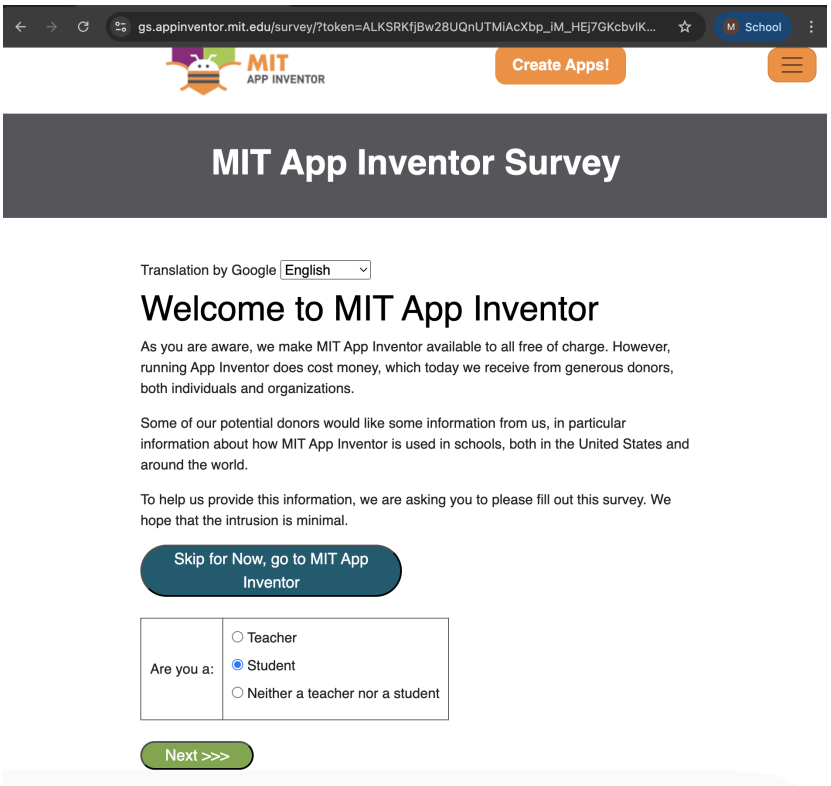
- Buka browser di komputer (seperti Google Chrome, Mozilla Firefox, atau lainnya).
- Ketik alamat situs App Inventor di bagian atas browser: **ai2.appinventor.mit.edu** lalu tekan Enter.
- Kamu akan diminta untuk masuk menggunakan akun Google. Jika sudah login, kamu mungkin bisa langsung masuk. Jika belum, masukkan email dan password Google-mu.



- Saat pertama kali masuk, mungkin akan muncul permintaan izin. Klik saja **"Allow"** (Izinkan). Setelah itu, setujui persyaratan layanan (*Terms of Service*) yang muncul. Selamat.



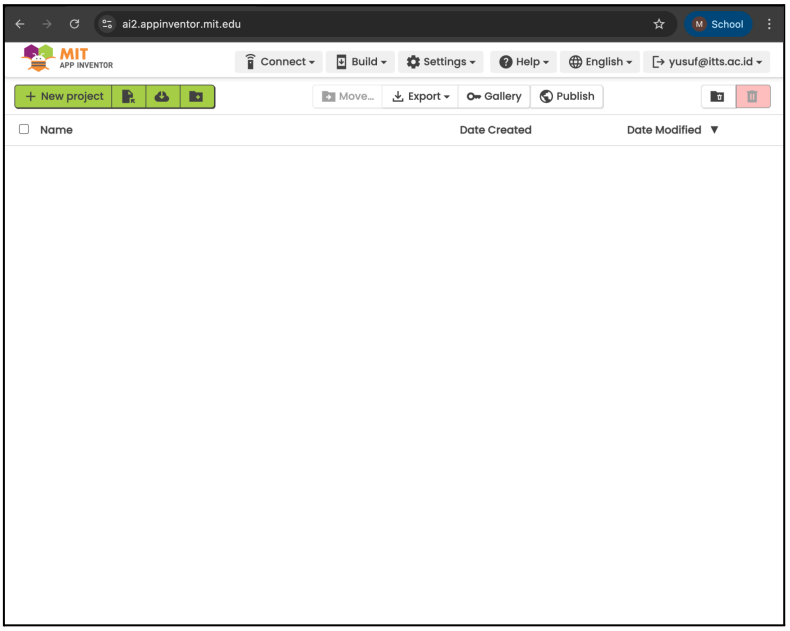
- Setelah itu kamu akan diminta mengisi survey. Klik saja "**Student".** Selamat, kamu sudah berhasil masuk ke App Inventor!



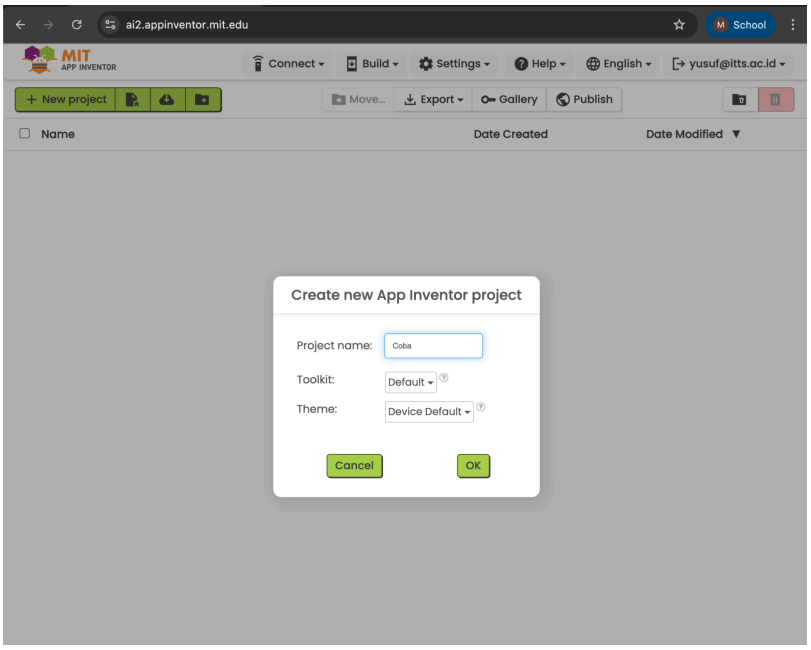
Kasus 2: Membuat Proyek Baru

Setiap aplikasi yang kita buat disebut sebagai "proyek". Yuk, kita buat proyek pertama kita!

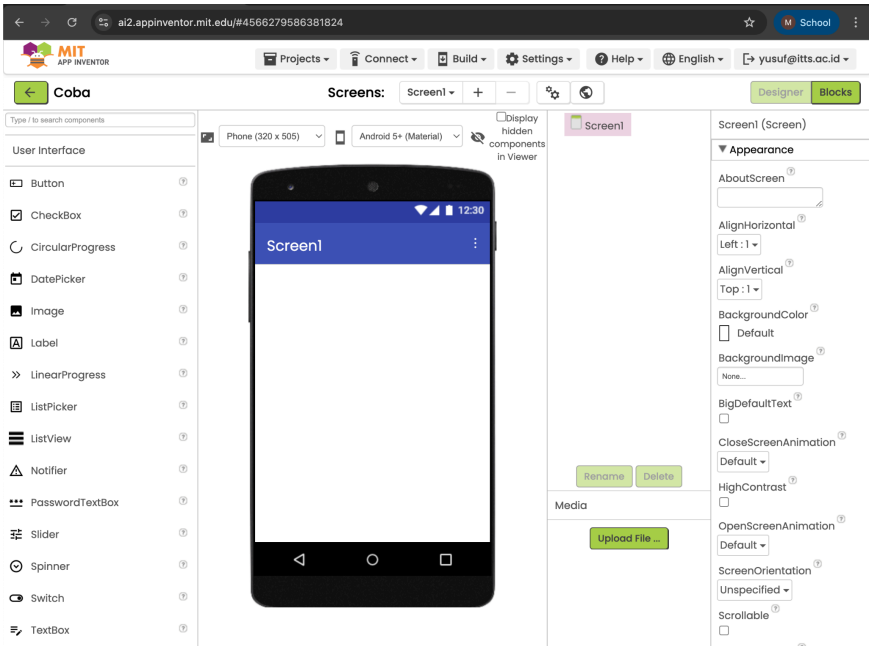
- Di halaman utama App Inventor, cari dan klik tombol **"Start new project"** (Mulai proyek baru) yang ada di bagian kiri atas.



- Akan muncul sebuah kotak kecil yang memintamu memberi nama untuk proyek ini. Ketik nama proyekmu, misalnya **"Coba"**.
- **Penting:** Nama proyek tidak boleh menggunakan spasi. Jika ingin memisahkan kata, gunakan garis bawah (_) atau sambungkan saja katanya.



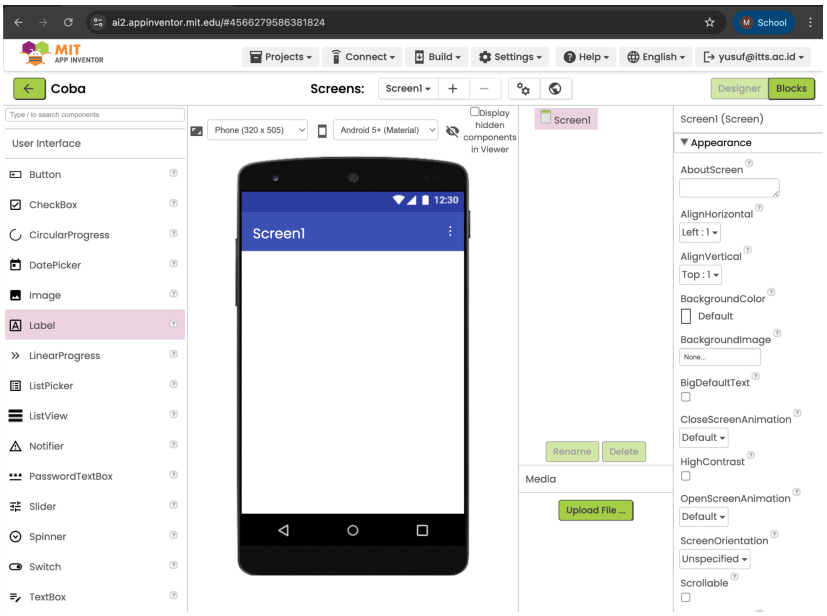
- Klik **"OK"**. Secara otomatis, kamu akan dibawa ke halaman kerja proyek barumu.



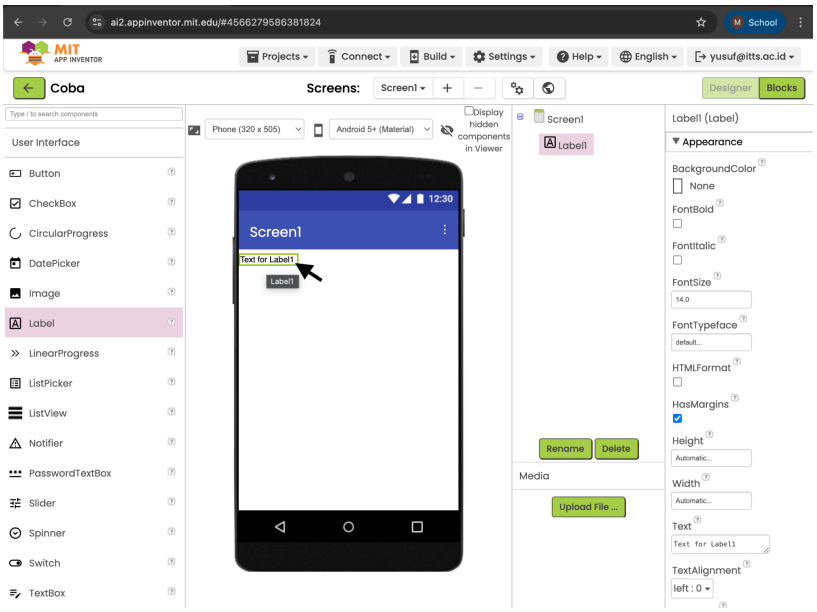
Kasus 3: Menambahkan Komponen Label dan Button

Sekarang kita akan mulai "menghias" aplikasi kita. Kita akan menambahkan sebuah tulisan (Label) dan sebuah tombol (Button).

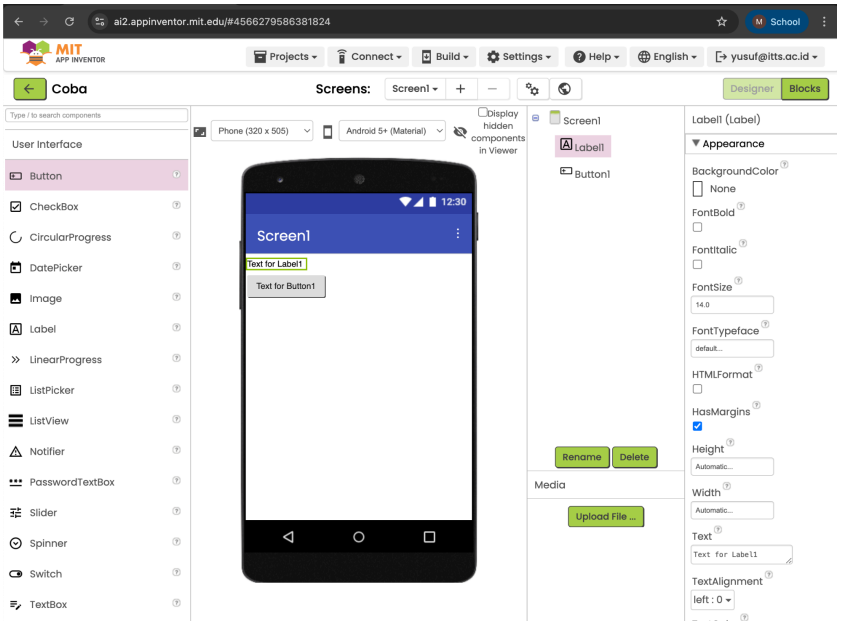
- Lihat ke panel **Palette** di sebelah kiri. Di bawah kategori *User Interface*, kamu akan menemukan banyak komponen.



- Klik pada komponen **Label**, tahan, lalu seret (drag) ke tengah layar di bagian **Viewer**. Lepaskan klik-mu. Sekarang kamu punya sebuah Label di aplikasimu!



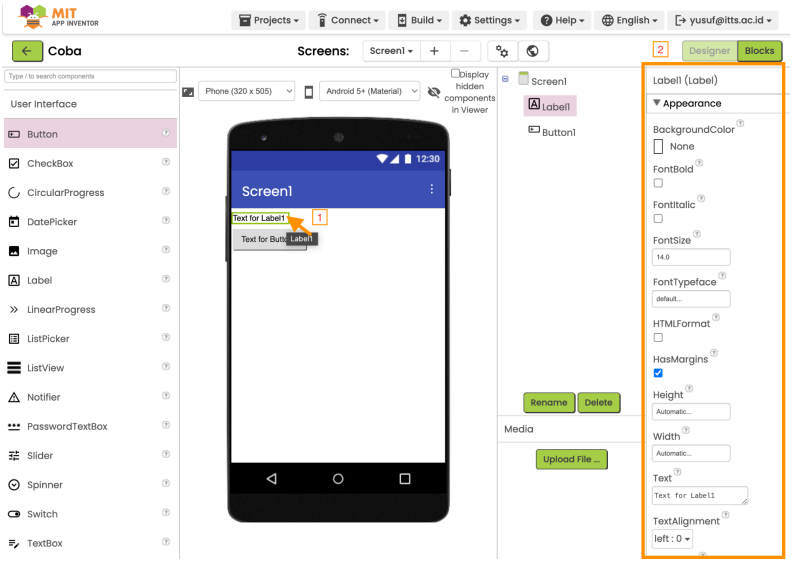
- Lakukan hal yang sama untuk komponen **Button**. Klik, tahan, dan seret ke bawah Label yang sudah kamu buat tadi.



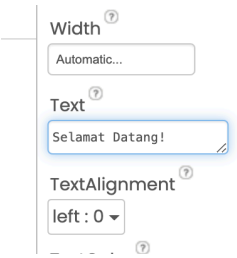
Kasus 4: Mengubah Properti Teks

Tulisan "Text for Label1" dan "Text for Button1" terlihat kurang menarik. Mari kita ubah tulisannya.

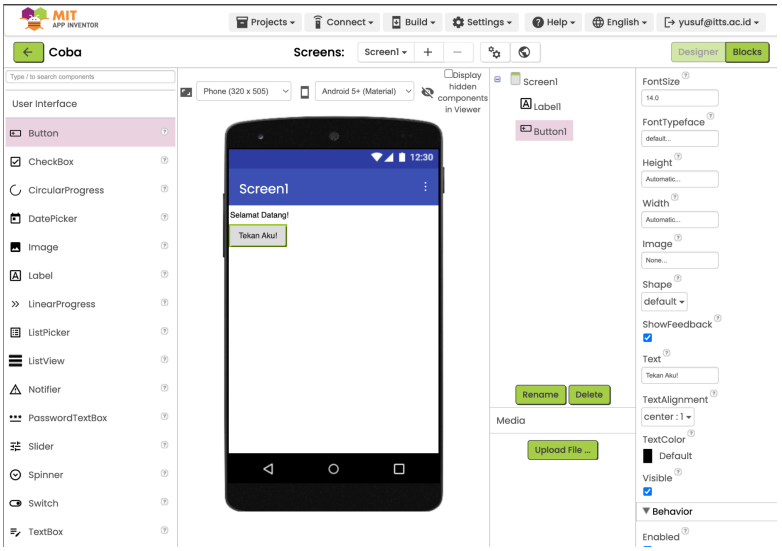
- Di area **Viewer** atau di **Components List**, klik pada komponen **Label1** untuk memilihnya.
- Sekarang, lihat ke **Properties Panel** di sebelah kanan. Geser ke bawah sampai kamu menemukan properti bernama **"Text"**.



- Hapus tulisan "Text for Label1" dan ganti dengan "Selamat Datang!". Tekan Enter. Lihat, tulisan di layar aplikasimu langsung berubah!



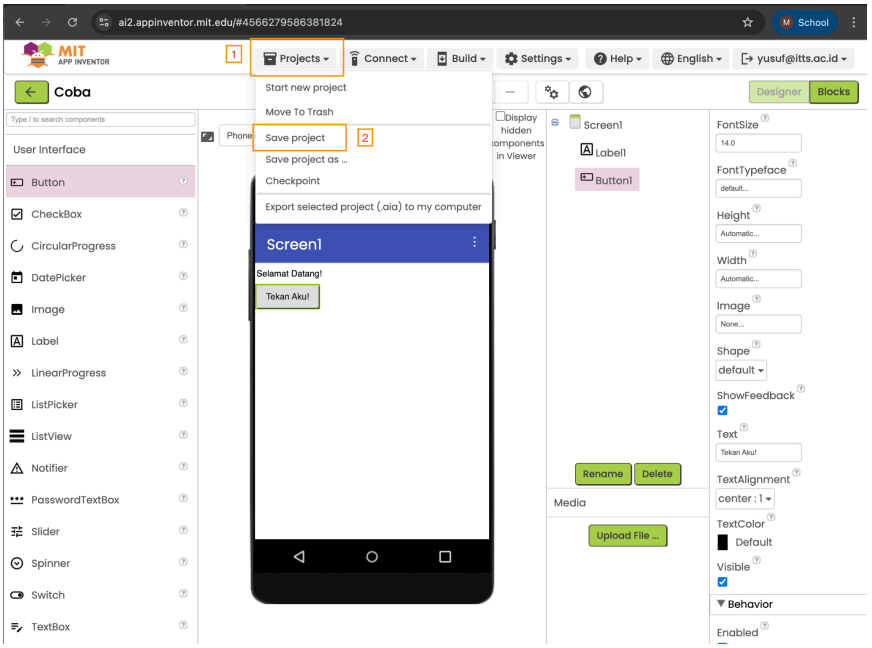
- Sekarang, lakukan hal yang sama untuk **Button1**. Klik Button1, cari properti **"Text"** di panel Properties, lalu ganti tulisannya menjadi "Tekan Aku!".



Kasus 5: Menyimpan Proyek

Jangan lupa untuk selalu menyimpan hasil kerjamu.

- Lihat ke menu bar di bagian atas layar.
- Klik menu **"Projects"**.
- Pilih opsi **"Save project"**. Proyekmu sekarang sudah aman tersimpan di akun App Inventor-mu.



Selamat! Kamu baru saja menyelesaikan langkah-langkah paling dasar dalam membuat sebuah aplikasi. Kamu sudah bisa membuka App Inventor, membuat proyek, dan mendesain tampilan sederhana.

Contoh Soal dan Pembahasan

Soal 1: Apa Fungsi "Blocks Editor"?

Pertanyaan:
Bayangkan kamu sudah mendesain tampilan aplikasi dengan tombol dan gambar. Tapi saat tombol ditekan, tidak terjadi apa-apa. Wah, ada yang kurang nih! Menurutmu, bagian manakah dari App Inventor yang harus kamu gunakan agar tombol tersebut bisa memberikan aksi, misalnya memainkan suara?

A. Components Palette
B. Blocks Editor
C. Properties Panel
D. Media Library

Pembahasan:
Jawaban yang benar adalah **B. Blocks Editor**.

Blocks Editor adalah tempat kamu menyusun logika aplikasi. Di sinilah kamu memberi "perintah" agar saat tombol ditekan, ada aksi yang dilakukan, misalnya memainkan suara atau menampilkan tulisan.

Soal 2: Petualangan di Designer View

Pertanyaan:
Kamu ingin membuat tampilan aplikasi yang memuat tulisan sambutan dan sebuah tombol. Komponen apa saja yang perlu kamu ambil dari Designer View?

A. Label dan Image
B. Label dan Button
C. TextBox dan Sound
D. Button dan Canvas

Pembahasan:
Jawaban yang benar adalah **B. Label dan Button**.
Label digunakan untuk menampilkan teks sambutan seperti "Selamat Datang!", sedangkan Button dipakai agar pengguna bisa menekan sesuatu, misalnya untuk memulai atau memicu aksi.

Soal 3: Misteri Komponen yang Hilang

Pertanyaan:
Di Viewer kamu sudah menyeret Button dan Label. Tapi kok nama tombolnya masih "Text for Button1"? Kamu ingin menggantinya jadi "Mulai!". Di manakah kamu bisa mengganti tulisan tersebut?

A. Blocks Editor
B. Viewer
C. Components List
D. Properties Panel

Pembahasan:
Jawaban yang benar adalah **D. Properties Panel**.
Di Properties Panel, kamu bisa mengganti isi teks, warna, ukuran, dan tampilan lainnya dari komponen. Untuk mengganti "Text for Button1" menjadi "Mulai!", kamu tinggal edit bagian "Text".

Soal 4: Tantangan Nama Proyek

Pertanyaan:
Saat membuat proyek baru, kamu ingin menamainya “Proyek Pertama Saya”. Tapi muncul pesan error! Wah, kenapa ya?

A. Karena terlalu panjang
B. Karena tidak boleh pakai huruf kapital
C. Karena tidak boleh pakai spasi
D. Karena belum login

Pembahasan:
Jawaban yang benar adalah **C. Karena tidak boleh pakai spasi**.
App Inventor tidak memperbolehkan nama proyek menggunakan spasi. Solusinya? Gunakan garis bawah (_) seperti: `Proyek_Pertama_Saya` atau sambungkan katanya: `ProyekPertamaSaya`.

Soal 5: Langkah Menuju Aplikasi Pertamamu!

Pertanyaan:
Urutkan langkah-langkah dasar membuat aplikasi di App Inventor berikut ini:

1. Menyusun logika aplikasi di Blocks Editor
2. Menyimpan proyek
3. Mendesain tampilan di Designer View
4. Menjalankan aplikasi di HP dengan AI2 Companion

A. 3 - 1 - 2 - 4
B. 3 - 1 - 4 - 2
C. 2 - 3 - 1 - 4
D. 1 - 3 - 2 - 4

Pembahasan:
Jawaban yang benar adalah **A. 3 - 1 - 2 - 4**.
Langkah membuat aplikasi dimulai dengan:

- Mendesain tampilan (3)
- Menyusun logika perintah (1)
- Menyimpan hasil kerja (2)
- Mencoba aplikasinya di HP (4)

Fakta Menarik / Fun Fact

- **Dari Raksasa Teknologi ke Universitas Hebat!** Tahukah kamu? App Inventor pada awalnya dikembangkan oleh Google, lho! Setelah itu, proyek ini diserahkan kepada Massachusetts Institute of Technology (MIT), salah satu universitas teknologi terbaik di dunia, untuk dikelola dan dikembangkan lebih lanjut. Ini menunjukkan betapa hebatnya platform ini, karena dibuat oleh raksasa teknologi dan sekarang dirawat oleh pusat pendidikan terkemuka.
- **Jutaan Aplikasi di Seluruh Dunia!** Kamu tidak sendirian! App Inventor digunakan oleh jutaan orang dari hampir 200 negara untuk membuat aplikasi mereka sendiri. Total sudah ada lebih dari 115 juta aplikasi yang dibuat menggunakan App Inventor. Bahkan, banyak dari aplikasi tersebut yang cukup bagus untuk diterbitkan secara resmi di Google Play Store, sama seperti aplikasi-aplikasi terkenal yang kamu gunakan sehari-hari!

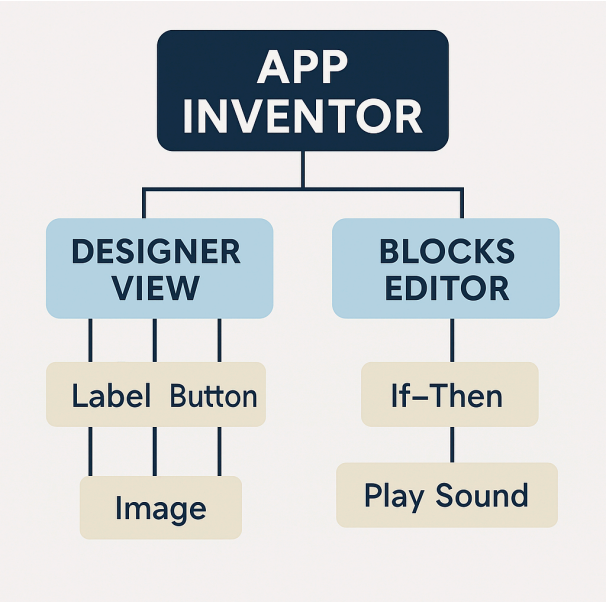
Tips Belajar atau Tips Cepat Menghafal

Agar tidak bingung saat membuat aplikasi, ingat dua hal utama ini:

- **Ingat: "Desain di Designer, atur perilaku di Blocks."** Ini adalah aturan paling dasar di App Inventor. Pikirkan seperti membangun robot:
    - **Designer** adalah tempat kamu merakit "tubuh" robot. Kamu merancang tampilannya, menaruh tombol, gambar, dan tulisan agar terlihat keren.
    - **Blocks** adalah tempat kamu memprogram "otak" robot. Kamu menyusun balok-balok perintah untuk menentukan apa yang akan dilakukan robot itu saat tombolnya ditekan.
- **Ingat: "Setiap komponen punya *properties* yang bisa diubah."** Setiap komponen atau "barang" yang kamu tambahkan ke layar, seperti tombol atau tulisan, memiliki daftar pengaturan yang disebut *Properties*. Di sinilah kamu bisa berkreasi! Kamu bisa mengubah hampir semua hal, mulai dari warna, ukuran huruf, hingga teks yang ditampilkan, untuk membuat aplikasi terlihat unik dan sesuai keinginanmu.

Membuat aplikasi mungkin terdengar sulit, tapi kamu bisa memulainya dengan cara yang seru dan mudah diingat! Yuk, coba trik cepat berikut ini:

Gunakan Mind Mapping!
Mind mapping itu seperti peta harta karun otakmu. Coba buat seperti ini:



Buat Cerita Seru
Bayangkan kamu membuat robot lucu bernama "RoboApp".

- Di *Designer View*, kamu merakit badannya: matanya adalah *Label*, tangannya *Button*.
- Di *Blocks Editor*, kamu mengatur otaknya: "Kalau tangan kanan ditekan, ucapkan ‘Hai!’."

Dengan cerita ini, kamu akan lebih mudah ingat bahwa Designer untuk tampilan, Blocks untuk logika.

Aktivitas Siswa

Ayo kita langsung mencoba dan berpetualang di dalam App Inventor! Aktivitas ini akan membantumu mengenal lebih dekat semua tool yang ada. Jangan takut untuk mencoba-coba, ya!

Tantangan Seru App Inventor: Jelajah Dunia Aplikasi!

Misi 1: Seret-Seret Komponen!

Anggap kamu sedang mendesain game atau aplikasi unikmu sendiri. Yuk mulai main!

Ambil dari "Kotak Mainan" (Palette):

- Dari **User Interface**:
  - 🔳 Image
  - 📝 TextBox
  - 🎚️ Slider
- Dari **Layout**:
  - 📦 HorizontalArrangement
- Dari **Media**:
  - 🔊 Sound *(komponen tak terlihat!)*

Tantangan: Seret dan letakkan **minimal 5 komponen** ke layar Viewer. Coba kombinasikan yang lucu!

Misi 2: Warna-Warni Pelangi!

Saatnya jadi desainer aplikasi.

Ganti Warna Latar:

- Klik **Screen1**
- Di panel **Properties**, ubah `BackgroundColor` ke warna pelangi favoritmu!

Ubah Ukuran Tulisan:

- Klik Label atau Button
- Ubah `FontSize` jadi misalnya **20**

Atur Ukuran Komponen:

- Klik Button
- Ubah `Width` & `Height` ke **"Fill parent..."**
- Lihat komponenmu membesar!

Misi 3: Blok Ajaib!

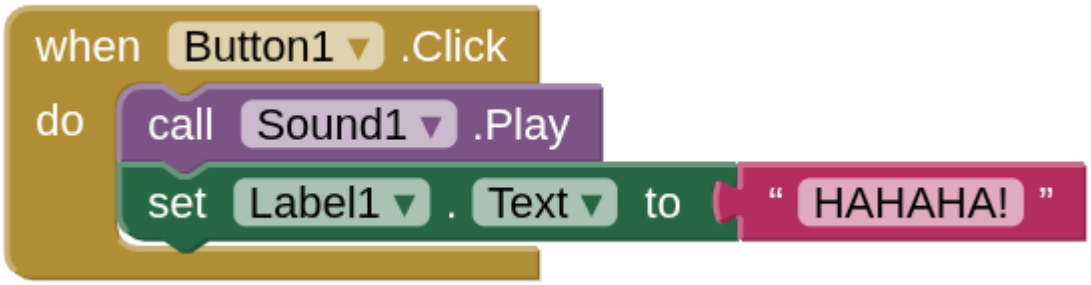
Buka **Blocks Editor** dan bikin keajaiban!

🖱️ Tambahkan sebuah **Button**, lalu program agar:

- Saat ditekan ➜ muncul **suara lucu**
  atau
- Saat ditekan ➜ **teks berubah jadi aneh!**

Gunakan blok seperti:

```
when Button1.Click do
  call Sound1.Play
  set Label1.Text to "HAHAHA!"
```



Misi 4: Uji Nyali!

📱 Hubungkan HP-mu ke App Inventor (pakai kode QR di companion app).
Lihat hasil karyamu **langsung muncul di layar**. Ajaib, kan?

Misi 5: Simpan & Kembangkan!

Menyimpan Proyek:

- Klik menu **Projects > Save project**

Membuka Proyek Lama:

- Menu **Projects > My projects**
- Klik nama proyekmu (contoh: “Coba”)

Kamu Luar Biasa!

Sekarang kamu sudah bisa:

- Menggunakan berbagai komponen
- Mengatur tampilan aplikasi
- Menulis blok logika sederhana
- Melihat hasil langsung di HP

- Menyimpan dan membuka proyek

Langkah berikutnya? Bikin aplikasi paling aneh dan paling keren yang bisa kamu bayangkan!

Rangkuman

Pernahkah kamu membayangkan membuat aplikasi sendiri, seperti yang kamu pakai di HP sehari-hari? Nah, lewat *MIT App Inventor 2*, kamu bisa belajar membuat aplikasi Android **tanpa perlu mengetik kode rumit**! Cukup dengan **menyusun balok perintah** seperti puzzle, kamu sudah bisa memberi "perintah" pada aplikasimu. Seru, kan? Kamu akan bekerja di dua tempat utama: **Designer View**, tempat kamu mendesain tampilan aplikasi, dan **Blocks Editor**, tempat kamu mengatur logika aplikasimu. Dengan alat ini, kamu bisa menjadi **pencipta aplikasi** yang kreatif hanya dalam beberapa langkah mudah.

Di *Designer View*, kamu akan menemukan berbagai "alat" seperti **Label, Button, dan Image** yang bisa kamu seret ke layar *Viewer* untuk mempercantik aplikasi. Setelah itu, kamu bisa mengatur properti setiap komponen lewat **Properties Panel**, mengganti teks, warna, ukuran, bahkan mengunggah gambar atau suara lewat menu **Media**. Di bagian **Blocks Editor**, kamu akan menyusun perintah—misalnya, "kalau tombol ditekan, putar suara." Menyusun blok-blok ini seperti menyusun ide-ide cerdasmu jadi kenyataan!

Proses membuat aplikasi bisa dibagi menjadi tiga langkah seru: **desain tampilan**, **atur logika**, dan **uji coba langsung di HP-mu**. Kamu akan mulai dari membuka situs *App Inventor*, membuat *project* baru, lalu menambahkan dan mengatur komponen seperti Label dan Button. Setelah itu, kamu bisa mengganti teksnya menjadi lebih menarik seperti “**Selamat Datang!**” dan “**Tekan Aku!**”. Jangan lupa untuk **menyimpan proyekmu**! Dengan langkah-langkah ini, kamu sudah selangkah lebih dekat untuk menjadi seorang *app developer* muda. Yuk, lanjutkan petualanganmu dan buat aplikasi yang unik dari idemu sendiri!

Latihan Soal

A. Benar atau Salah (5 Soal)

(Tulis "Benar" atau "Salah" saja sebagai jawaban)

1. App Inventor 2 adalah platform untuk membuat aplikasi Android dengan menulis kode program.
   Jawaban: Salah

2. Designer View digunakan untuk mengatur logika aplikasi.
   Jawaban: Salah

3. Viewer digunakan untuk melihat pratinjau desain aplikasi.
   Jawaban: Benar

4. Button adalah salah satu komponen di Components Palette.
   Jawaban: Benar

5. Kita bisa mengunggah suara ke dalam Media di App Inventor 2.
   Jawaban: Benar

B. Soal Pilihan Ganda (10 soal)

(Pilih A, B, C, atau D)

6. App Inventor 2 dikembangkan oleh:
   A. Microsoft
   B. Google dan MIT
   C. Apple
   D. Facebook
   Jawaban: B

7. Fungsi utama dari Blocks Editor adalah:
   A. Mengubah teks pada Label
   B. Menampilkan aplikasi di HP
   C. Mengatur logika aplikasi
   D. Menyimpan file proyek
   Jawaban: C

8. Yang bukan termasuk komponen User Interface di App Inventor adalah:
   A. Label
   B. Button
   C. Mouse
   D. Image
   Jawaban: C

9. Untuk mengubah ukuran tombol, kita bisa mengatur di bagian:
   A. Viewer
   B. Components List
   C. Properties Panel
   D. Media
   Jawaban: C

10. Jika ingin menambahkan suara ke aplikasi, kita harus mengunggah file ke:
    A. Properties
    B. Components List
    C. Media

D. Viewer
Jawaban: C

11. Screen1 adalah nama default untuk:
    A. Proyek baru
    B. Tampilan utama aplikasi
    C. Komponen suara
    D. Komponen logika
    Jawaban: B

12. Menu untuk menyimpan proyek berada di:
    A. Media
    B. Project
    C. Help
    D. File
    Jawaban: B

13. Komponen yang menampilkan teks statis adalah:
    A. Label
    B. Button
    C. Screen
    D. Image
    Jawaban: A

14. Warna latar belakang dapat diubah dari:
    A. Components List
    B. Viewer
    C. Blocks Editor
    D. Properties Panel
    Jawaban: D

15. Saat membuat aplikasi, urutan yang benar adalah:
    A. Blok dulu, baru desain
    B. Upload ke Play Store dulu
    C. Desain tampilan, atur logika, uji aplikasi
    D. Buat logo dulu
    Jawaban: C

C. Soal Isian Singkat (5 soal)

16. Tempat untuk menyusun logika aplikasi di App Inventor 2 disebut _____.
    Jawaban: Blocks Editor

17. Untuk melihat desain aplikasi secara langsung, kita menggunakan _____.
    Jawaban: Viewer

18. Komponen yang digunakan untuk menampilkan gambar disebut _____.
    Jawaban: Image

19. Menu untuk memulai proyek baru adalah _____.
    Jawaban: Start new project

20. Daftar semua komponen yang sudah kita tambahkan akan muncul di _____.
    Jawaban: Components List

D. Soal Explorasi

Soal:

Kamu diminta membuat aplikasi sederhana berisi tombol dan label. Deskripsikan langkah-langkah yang kamu lakukan mulai dari membuka App Inventor 2, membuat proyek, sampai menjalankan aplikasi di emulator atau ponsel.

Soal:

Bayangkan kamu ingin membuat aplikasi yang saat tombol ditekan, tulisan di layar berubah menjadi “Halo Dunia!”. Tuliskan komponen apa saja yang kamu perlukan dan bagaimana kamu akan mengatur logikanya di Blocks Editor.

Soal:

Kamu sedang menjelaskan kepada temanmu yang belum pernah menggunakan App Inventor 2. Buatlah penjelasan singkat dalam satu paragraf tentang perbedaan antara Designer View dan Blocks Editor serta mengapa keduanya penting dalam membuat aplikasi.

Tugas Proyek

Aktivitas 1: Membuka Gerbang Menuju Dunia Pembuat Aplikasi

Tujuan: Siswa dapat membuka dan masuk ke platform MIT App Inventor 2 menggunakan akun Google.

Langkah-langkah yang disarankan:

- Buka browser favoritmu (Chrome, Firefox, dsb.).
- Ketik alamat: **ai2.appinventor.mit.edu**, lalu tekan **Enter**.
- Masuk menggunakan akun Google masing-masing.
- Izinkan akses dan isi survei singkat dengan memilih opsi “Student”.
- Selamat! Kamu sudah siap menjelajahi dunia penciptaan aplikasi.

Tantangan: Coba bantu temanmu yang belum berhasil login. Jadi "mentor mini" hari ini!

Aktivitas 2: Beri Nama Karya Pertamamu!

Tujuan: Siswa mampu membuat proyek baru di App Inventor dan memahami pentingnya penamaan proyek.

Langkah-langkah yang disarankan:

- Di halaman awal, klik **Start new project**.
- Berikan nama unik dan kreatif untuk aplikasimu, misalnya: `HaloDunia`, `Aplikasi_Ku`, atau `SapaTeman`.
- Ingat: Hindari spasi! Gunakan garis bawah (_) jika perlu.
- Tekan **OK** dan kamu akan diarahkan ke ruang kerja proyekmu.

Tantangan: Ciptakan nama proyek yang menggambarkan ide aplikasi hanya dengan satu kata!

Aktivitas 3: Mendesain Wajah Aplikasimu

Tujuan: Siswa mengenal dan menggunakan komponen dasar seperti Label dan Button di tampilan aplikasi.

Langkah-langkah yang disarankan:

- Temukan **komponen Label** dan **Button** di sisi kiri (Palette).
- Seret dan lepaskan Label ke layar tengah (Viewer).
- Tambahkan juga Button di bawah Label.
- Tata posisi komponen agar terlihat rapi dan menarik.

Tantangan: Buat layout yang simetris dan enak dilihat. Bayangkan kamu sedang mendesain aplikasi sungguhan untuk publik!

Aktivitas 4: Menyulap Teks Jadi Lebih Menarik

Tujuan: Siswa memahami cara mengatur properti komponen, terutama bagian teks.

Langkah-langkah yang disarankan:

- Klik **Label1** di Viewer atau di daftar komponen.
- Di panel kanan (Properties), cari bagian **Text**.
- Ubah menjadi kalimat sapaan seperti "Halo, Dunia!".
- Lakukan hal yang sama pada Button1, misalnya ubah jadi “Klik Aku!” atau “Mulai!”.

Tantangan: Ubah juga ukuran, warna, dan jenis huruf Label agar tampilannya lebih kreatif!

Aktivitas 5: Jadikan Aplikasimu Hidup

Tujuan: Siswa memahami konsep logika dasar dengan menyusun blok perintah di Blocks Editor.

Langkah-langkah yang disarankan:

- Klik tab **Blocks** untuk masuk ke editor logika.

- Pilih komponen Button1 di daftar blok.
- Tambahkan perintah sederhana seperti: *“Saat Button1 diklik, tampilkan pesan.”*
- Gunakan balok seperti `when Button1.Click do` dan sambungkan dengan perintah `call Notifier.ShowMessageDialog`.

Tantangan: Tambahkan aksi unik—misalnya, ubah warna latar atau memainkan suara saat tombol ditekan!

Aktivitas 6: Uji Coba dan Simpan Karyamu

Tujuan: Siswa mampu menguji aplikasi buatan sendiri dan menyimpan proyek secara aman.

Langkah-langkah yang disarankan:

- Hubungkan HP Android menggunakan aplikasi **MIT AI2 Companion**.
- Scan QR Code yang muncul di App Inventor.
- Amati tampilan dan perintah aplikasimu langsung di HP!
- Jika sudah sesuai, klik **Projects > Save project** untuk menyimpan pekerjaanmu.

Tantangan: Temukan satu hal yang ingin kamu perbaiki dari aplikasi dan coba ubah sendiri!

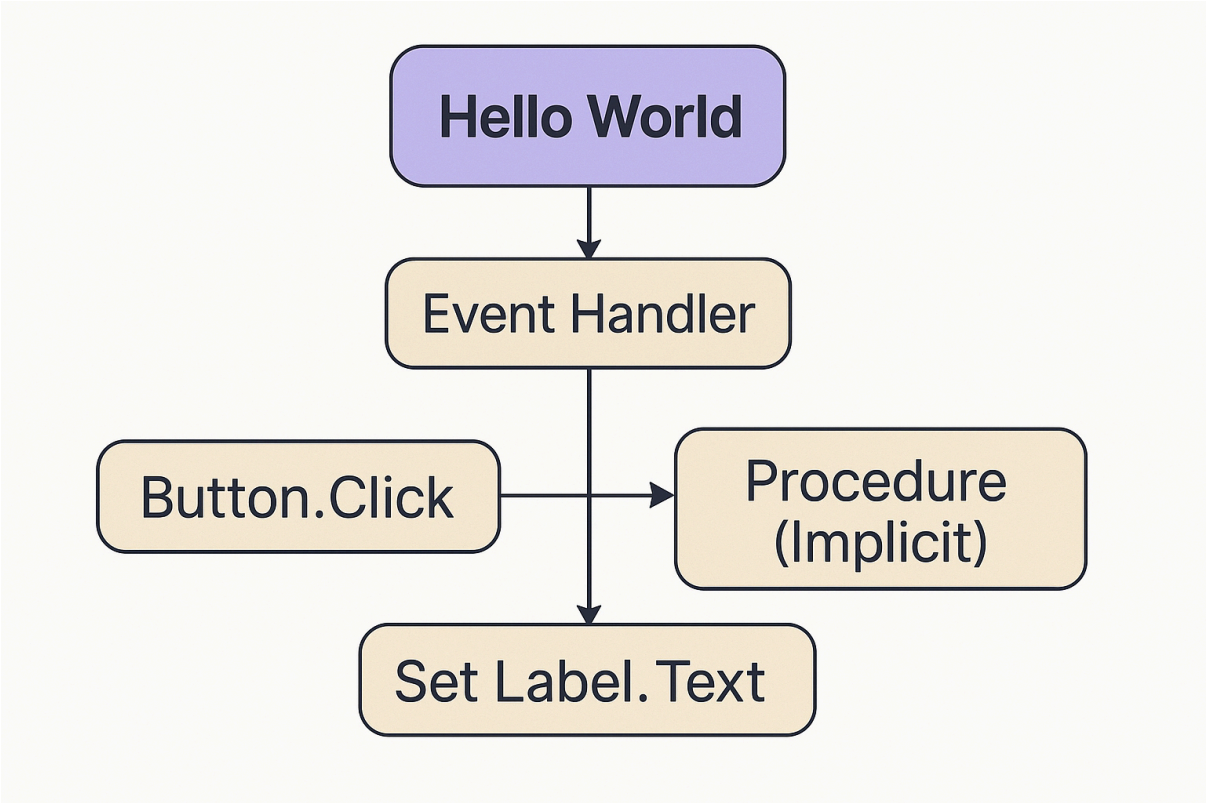
BAB 2: Membuat Aplikasi "Hello World" Interaktif

Tujuan Pembelajaran

Pada akhir pembelajaran ini, kalian diharapkan mampu:

- Membuat aplikasi sederhana yang dapat merespons sentuhan pengguna.
- Menggunakan blok perintah untuk mengubah tampilan teks saat tombol ditekan.
- Menggabungkan komponen tampilan dan logika (event handling) untuk membuat aplikasi interaktif.

Peta Konsep



Apersepsi

Pernahkah kalian menekan tombol "Suka" atau "Komentar" di media sosial?

- Apa yang terjadi saat kalian menyentuh tombol itu?
- Bagaimana aplikasi tahu harus menampilkan sesuatu?

Nah, hari ini kalian akan belajar bagaimana cara membuat aplikasi **yang bisa melakukan hal seperti itu!** Dengan satu sentuhan, aplikasi bisa langsung merespons — keren, kan?

Penjelasan Konsep (Teori)

Setelah kita bisa mendesain tampilan, sekarang saatnya kita memberi "otak" pada aplikasi kita di **Blocks Editor**. Kita akan belajar bagaimana membuat aplikasi merespons tindakan pengguna.

Event Handling (Menangani Kejadian)

Bayangkan sebuah bel di rumah. Apa yang terjadi **ketika** seseorang menekan bel? Kamu akan **melakukan** sesuatu, misalnya membuka pintu.

Dalam aplikasi, tindakan pengguna seperti menekan tombol disebut **"event"** atau **"kejadian"**. Aplikasi tidak akan melakukan apa-apa sampai kita memberinya perintah tentang apa yang harus dilakukan ketika sebuah kejadian terjadi. Proses memberi perintah inilah yang disebut *Event Handling*. Kita memberitahu aplikasi: "**Ketika** kejadian ini terjadi, **lakukan** perintah itu."

Blok `when Button1.Click do`

Ini adalah blok perintah yang paling sering kita gunakan. Blok ini berwarna emas dan bentuknya seperti mulut yang siap "memakan" perintah-perintah lain di dalamnya.

- **`when Button1.Click`**: Artinya, "**Ketika** Tombol1 di-klik". Ini adalah *event* atau pemicunya.
- **`do`**: Artinya, "**lakukan**".



Jadi, blok ini secara sederhana berarti: "Ketika pengguna mengklik Tombol1, lakukan semua perintah yang ada di dalam mulut blok ini." Perintah apa pun yang kita masukkan ke dalamnya hanya akan dijalankan setelah tombol itu diklik.

Blok `set Label1.Text to`

Bagaimana cara kita mengubah tulisan pada sebuah Label saat aplikasi berjalan? Kita menggunakan blok "setter". *Setter* bertugas untuk **mengatur** atau **mengubah** properti sebuah komponen.

- **`set Label1.Text`**: Artinya, "Atur properti **Teks** milik **Label1**".

- `to`: Artinya, "menjadi".



Blok ini memberi kita kekuatan untuk mengubah tulisan pada Label secara dinamis. Misalnya, tulisan yang awalnya "Selamat Datang" bisa kita ubah menjadi "Kamu berhasil!".

String (Teks)

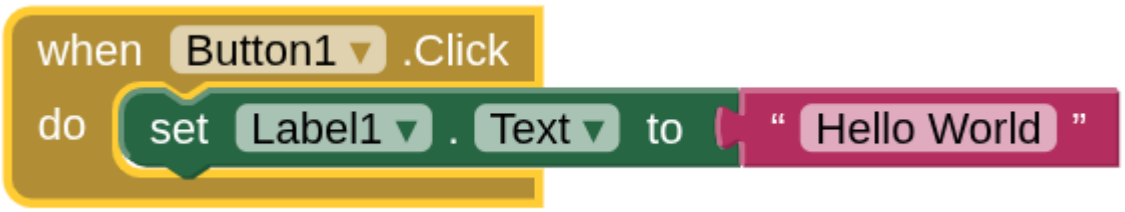
Dalam dunia pemrograman, semua bentuk tulisan (kata, kalimat, atau bahkan satu huruf) disebut sebagai **"String"**. Di App Inventor, kita menggunakan blok teks berwarna merah muda (pink) untuk membuat sebuah string.



Anggap saja blok ini seperti kertas catatan kecil. Kamu bisa menulis apapun di dalamnya. Kemudian, kertas catatan ini bisa kamu tempelkan ke blok perintah lain, misalnya ke blok `set Label1.Text to` untuk memberitahu aplikasi tulisan apa yang harus ditampilkan.

Membuat Aplikasi "Hello World" Dasar

Sekarang, mari kita gabungkan semua konsep di atas untuk membuat aplikasi paling dasar di dunia pemrograman: "Hello World". Tujuannya adalah membuat tulisan "Hello World!" muncul di layar ketika sebuah tombol ditekan.



Logikanya sangat sederhana:

1. Kita butuh pemicu, yaitu **ketika tombol diklik**. Kita gunakan blok `when Button1.Click do`.
2. Kita ingin **melakukan** sesuatu, yaitu mengubah tulisan pada Label.
3. Untuk mengubah tulisan, kita gunakan blok `set Label1.Text to`.

4. Tulisan apa yang ingin kita tampilkan? "Hello World!". Kita ambil blok teks berwarna pink, ketik "Hello World!" di dalamnya, lalu pasangkan ke blok `set Label1.Text to`.

Jika digabungkan, perintah lengkapnya menjadi: "**Ketika Tombol1 diklik, atur properti Teks pada Label1 menjadi 'Hello World!'**". Dengan begitu, aplikasimu menjadi interaktif!

Contoh Kasus atau Ilustrasi

Mari kita coba langsung membuat aplikasi ”Hello World”. Ikuti langkah-langkah berikut:

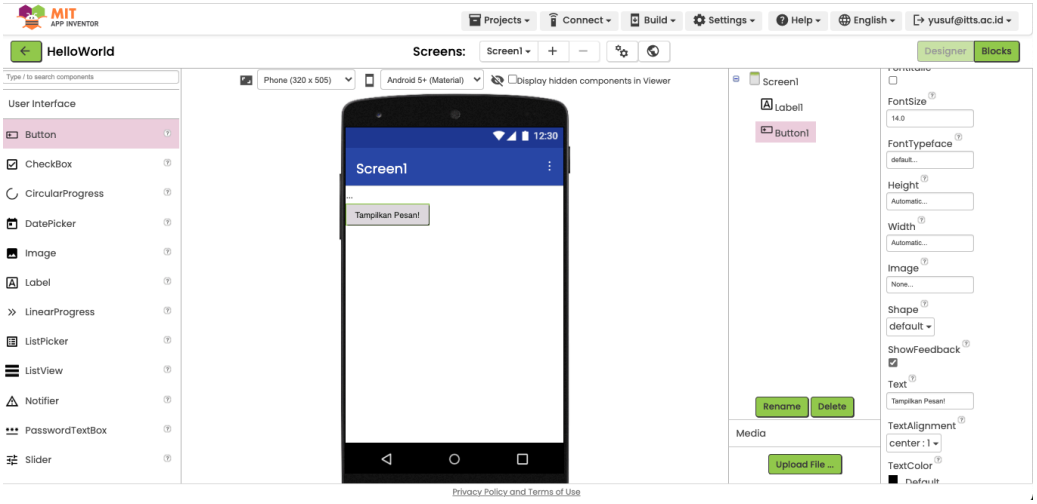
Kasus 1: Mendesain Tampilan

Pertama, kita siapkan dulu "wajah" aplikasi kita. Pastikan kamu berada di halaman **Designer**.

> Jika kamu bingung baca lagi panduan pada halaman **Membuat Proyek Baru**.

1. **Buat Proyek Baru:** Dengan nama **"HelloWorld"**.
2. **Tambahkan Label:**
   - Dari panel **Palette** di sebelah kiri, seret komponen **Label** ke tengah layar **Viewer**.
   - Klik pada `Label1` yang baru kamu tambahkan.
   - Di panel **Properties** di sebelah kanan, cari properti **Text**. Hapus tulisan yang ada dan ganti dengan **"..."** (tiga titik saja). Ini akan menjadi tulisan awal kita.
3. **Tambahkan Tombol:**
   - Dari panel **Palette**, seret komponen **Button** ke bawah Label.
   - Klik pada `Button1`.
   - Di panel **Properties**, ubah properti **Text**-nya menjadi **"Tampilkan Pesan!"**.

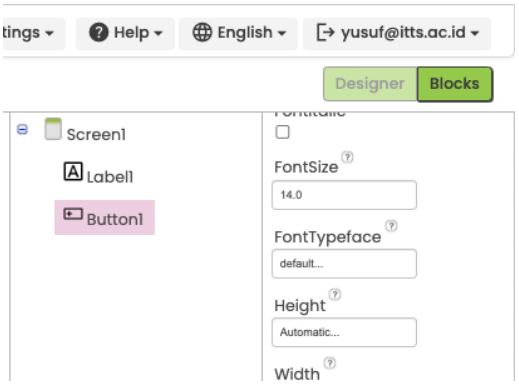
Tampilan desainmu sekarang seharusnya sudah siap.



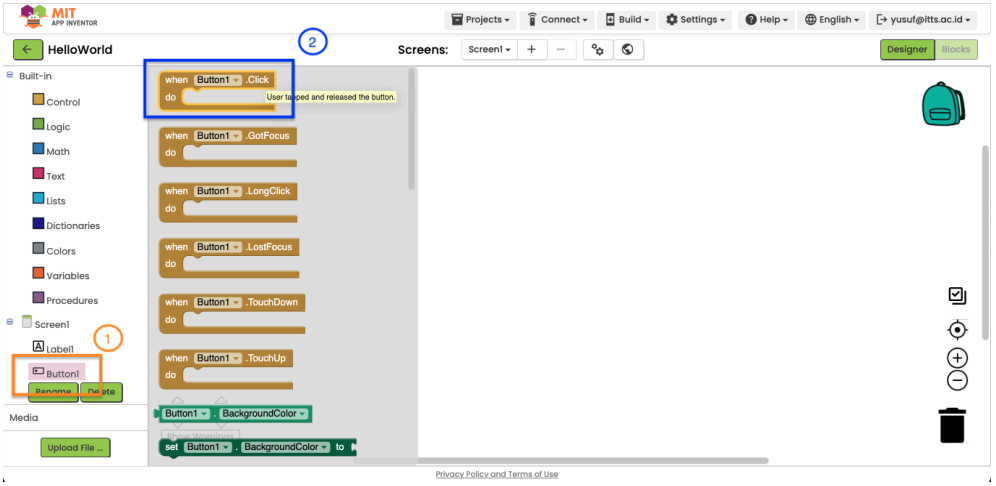
Kasus 2: Memberi Perintah

Sekarang, kita akan memberi "otak" pada aplikasi kita agar bisa melakukan sesuatu.

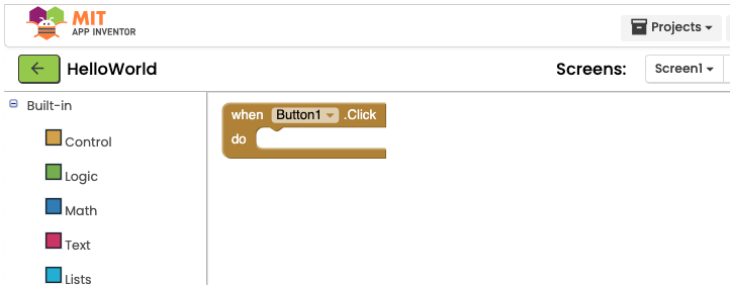
1. **Pindah ke Halaman Blocks:** Klik tombol **"Blocks"** di pojok kanan atas untuk pindah ke halaman editor blok.



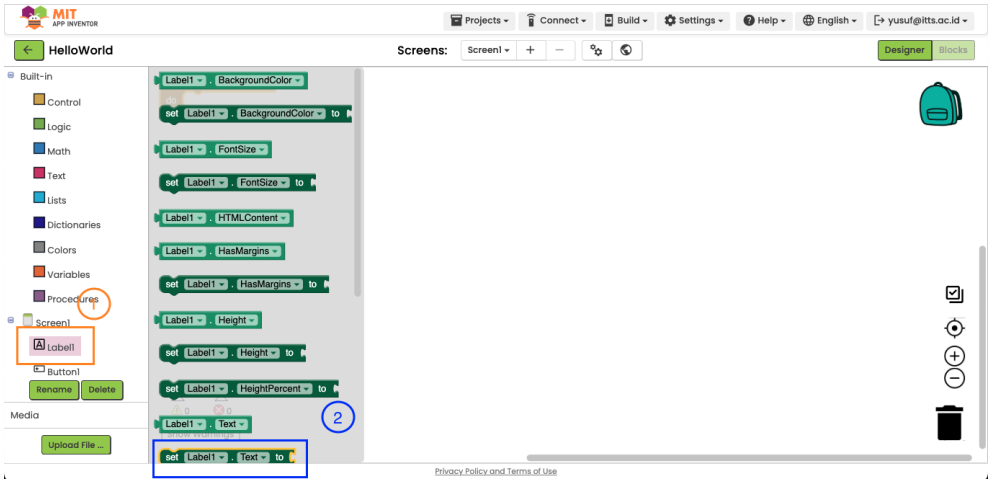
2. **Siapkan Pemicu (Event):**
    - Di panel **Blocks** sebelah kiri, klik pada **Button1**. Kamu akan melihat semua blok perintah yang berhubungan dengan tombol itu.
    - Cari dan seret blok `when Button1.Click do` ke area kerja. Blok emas ini adalah pemicunya, yang artinya "Ketika Tombol1 diklik, lakukan...".



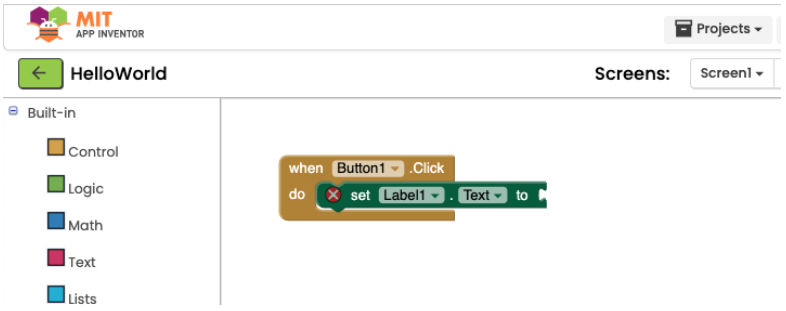
- Sehingga tampilan Blocks menjadi:



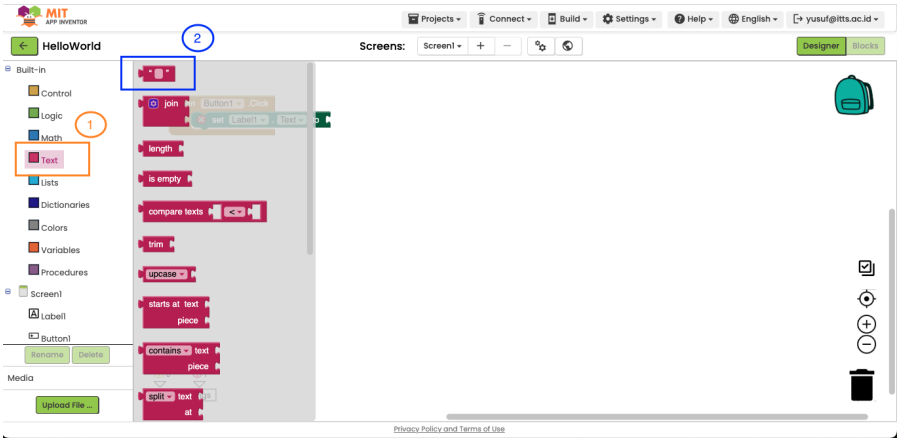
3. **Atur Perintah Mengubah Teks:**
   - Sekarang, klik pada **Label1** di panel kiri.
   - Cari blok `set Label1.Text to` dan seret ke dalam "mulut" blok `when Button1.Click do`. Blok ini bertugas untuk mengubah tulisan di Label1.



- Sehingga tampilan Block menjadi:

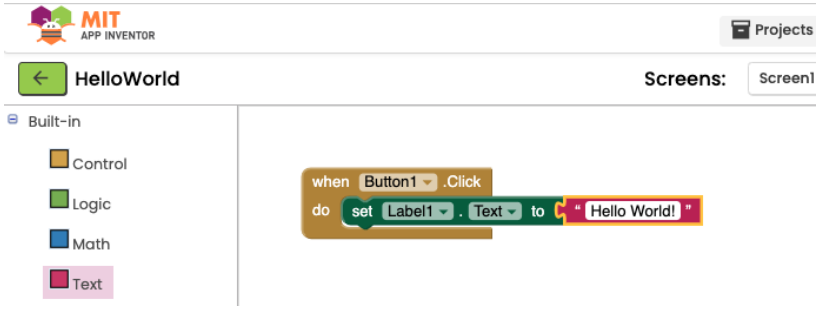


4. **Siapkan Teks Baru:**
   - Di panel kiri, klik pada kategori **Text** (berwarna pink).
   - Ambil blok paling atas yang berisi tanda kutip kosong `" "`.



   - Pasangkan blok pink ini ke ujung blok `set Label1.Text to`.
   - Klik di dalam blok pink tersebut dan ketik **"Hello World!"**.

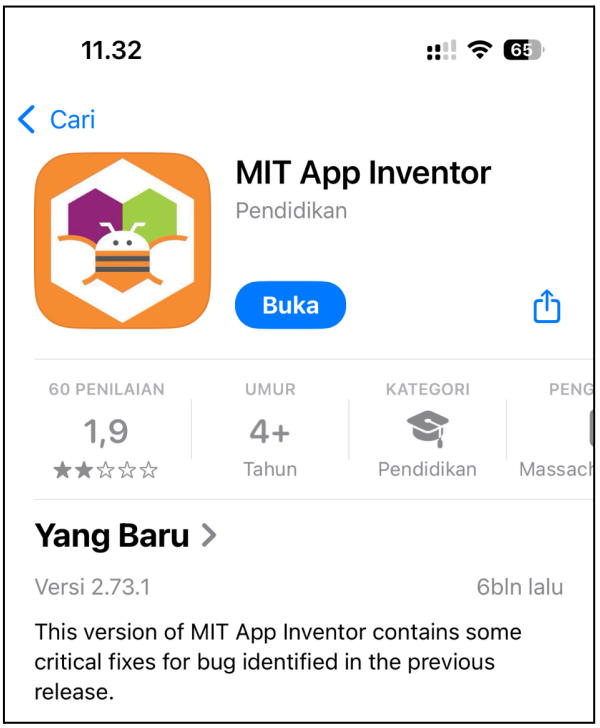
Jika semua langkah benar, susunan blok perintahmu akan terlihat seperti ini:



Kasus 3: Uji Coba Aplikasimu

Waktunya mencoba hasil karyamu!

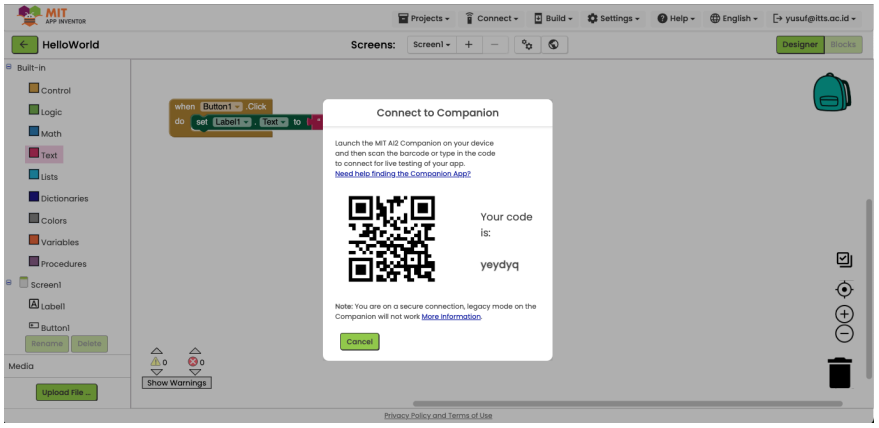
1. Install Aplikasi MIT App Inventor di Play Store (Android) atau Di App Store (iPhone).



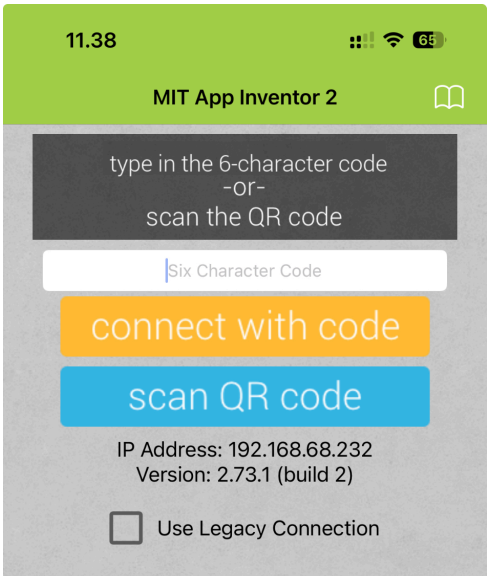
2. Buku Aplikasi MIT App Inventor di HP mu, kemudian hubungkan aplikasi menggunakan aplikasi **MIT AI Companion**.



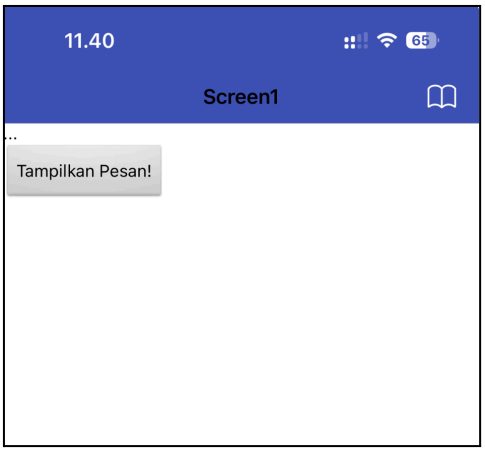
Sehingga tampilannya seperti ini:



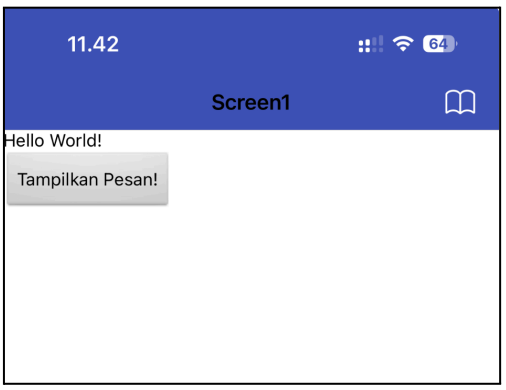
3. Kamu bisa melakukan scan di Aplikasi MIT App Inventor atau bisa juga dengan masukan 6 digit kode.



4. Saat aplikasi terbuka di HP, kamu akan melihat tulisan "..." dan sebuah tombol "Tampilkan Pesan!".



5. Tekan tombol “Tampilkan Pesan!”.
6. Jika berhasil, tulisan "..." akan langsung berubah menjadi **"Hello World!"**.



Selamat! Kamu baru saja berhasil membuat aplikasi interaktif pertamamu. Kamu sudah memberi perintah pada aplikasi untuk merespons sentuhan pengguna.

Contoh Soal dan Pembahasan

Soal 1: Si Tombol Ajaib

Pertanyaan:
Bayangkan kamu membuat aplikasi dengan satu tombol dan satu label. Saat tombol diklik, tulisan pada label berubah dari "..." menjadi "Halo Dunia!".
Blok perintah mana yang seharusnya kamu gunakan agar perintah ini bisa berjalan?

Jawaban & Pembahasan:
Kita butuh tiga blok utama:

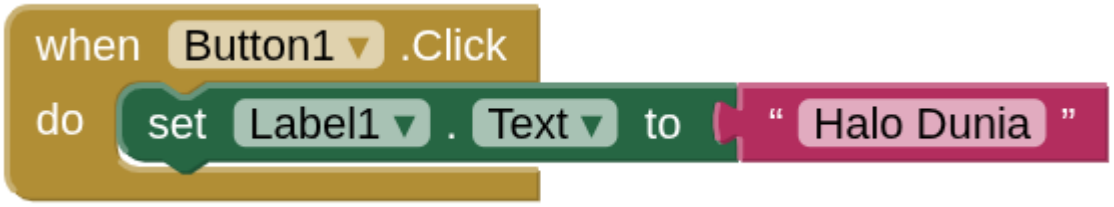
- `when Button1.Click do` → sebagai pemicu event saat tombol diklik.
- `set Label1.Text to` → untuk mengubah teks pada label.
- Blok teks `"Halo Dunia!"` → sebagai string yang akan ditampilkan.

Logika blok:

Saat tombol ditekan → ubah teks label menjadi "Halo Dunia!"

Jadi, urutannya adalah:

```
when Button1.Click do
    set Label1.Text to "Halo Dunia!"
```



Soal 2: Siapa yang Bergerak Duluan?

Pertanyaan:
Apa yang terjadi jika kamu membuat blok `set Label1.Text to "Selamat Datang!"` **tanpa** menempatkannya di dalam blok `when Button1.Click do`?

Jawaban & Pembahasan:
Jika blok `set Label1.Text to "Selamat Datang!"` **tidak berada di dalam** event `when Button1.Click do`, maka **perintah itu tidak akan pernah dijalankan**.

Dalam aplikasi, semua perintah harus dipicu oleh sebuah **kejadian (event)**. Tanpa pemicu, seperti tombol diklik, aplikasi tidak tahu **kapan** harus menjalankan perintah tersebut.

Ini seperti kamu menulis perintah tapi tidak pernah bilang "kapan harus mulai".

Soal 3: Menyusun Puzzle Interaktif

Pertanyaan:
Susunlah urutan blok berikut agar aplikasi bisa menampilkan teks "Kamu Hebat!" saat tombol ditekan:

A. set Label1.Text to
B. when Button1.Click do
C. "Kamu Hebat!" (blok teks pink)

Jawaban & Pembahasan:
Urutan yang benar:

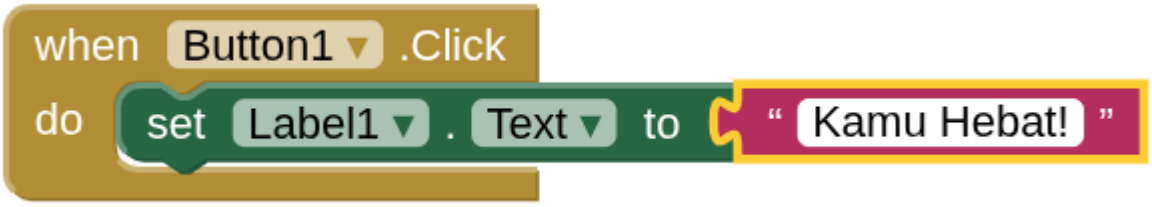
1. B. `when Button1.Click do`
2. A. `set Label1.Text to`
3. C. `"Kamu Hebat!"`

Kemudian, masukkan blok A ke dalam blok B, dan blok C ke dalam blok A.

Hasil akhir:

```
when Button1.Click do
```

set Label1.Text to "Kamu Hebat!"



Ini adalah cara kerja event handler yang benar: kejadian → aksi → hasil.

Soal 4: Misteri String

Pertanyaan:
Di dunia pemrograman, semua bentuk tulisan seperti kata, kalimat, atau satu huruf disebut apa?

Jawaban & Pembahasan:
Disebut **String**.

String adalah tipe data yang mewakili teks. Di App Inventor, string dibuat menggunakan **blok teks berwarna pink**, misalnya `"Hello World!"`.

Tanpa string, kita tidak bisa menampilkan tulisan di layar. Jadi string = tulisannya, bukan tombol atau labelnya.

Soal 5: Tombol Tak Berguna?

Pertanyaan:
Adit membuat aplikasi dengan tampilan seperti berikut:

- Ada Label dengan teks "..."
- Ada tombol "Klik Aku!"

Tapi saat tombol ditekan, tidak terjadi apa-apa.
Pertanyaan: Apa kemungkinan yang salah dalam blok perintah Adit?

Jawaban & Pembahasan:
Kemungkinan besar Adit **belum menambahkan blok event** `when Button1.Click do` atau ia sudah membuat event tersebut, tapi **tidak memasukkan perintah di dalamnya**.

Jadi, ada dua kemungkinan:

- Event tidak dibuat (tombol tak tahu harus melakukan apa).
- Event dibuat, tapi tidak ada perintah di dalamnya.

Solusinya:
Pastikan ada event seperti ini:

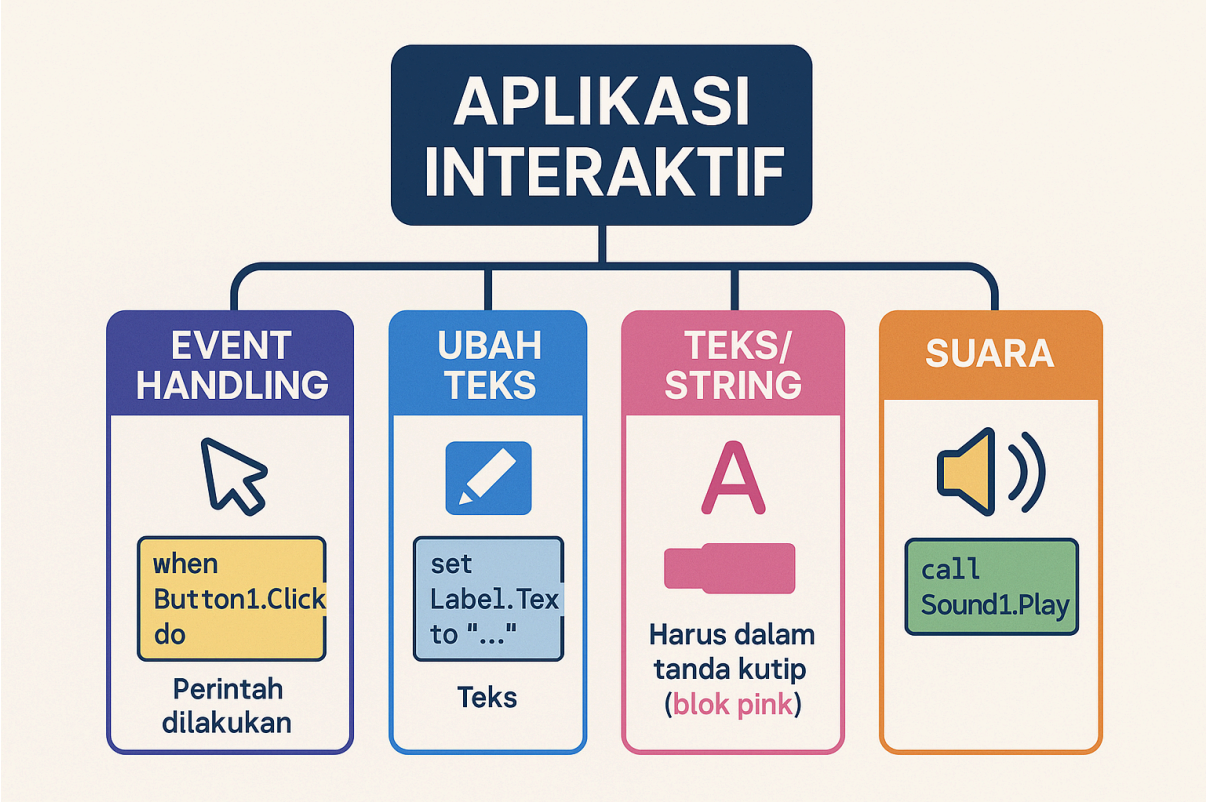
```
when Button1.Click do
    set Label1.Text to "Halo Dunia!"
```

Fakta Menarik / Fun Facts

- "Hello World!" adalah program pertama yang sering dibuat oleh pemula dalam banyak bahasa pemrograman!
- Banyak aplikasi besar juga diawali dari interaksi sederhana seperti ini.

Tips Belajar atau Tips Cepat Menghafal

- Ingat: "Setiap aksi (klik) punya reaksi (perubahan)."
- "Cari blok `when` untuk kejadian, cari blok `set` untuk mengubah."
- **Mind Mapping:**



- **Cerita Mudah Diingat:**

  Bayangkan kamu membuat **robot lucu** bernama "Appy". Appy tidak bisa bergerak atau bicara sampai kamu **menekan tombol** di tubuhnya. Saat kamu menekan tombol "Bicara", dia berkata "Hello World!". Kalau kamu tambahkan **speaker** ke Appy, dia juga bisa **mengeluarkan suara!**

- **Tips Cepat Hafal:**
    - **Aksi → Reaksi!** (Klik tombol → Teks berubah / Suara keluar)
    - Blok **when** = "Kalau ini terjadi..."
    - Blok **set** = "Ubah ini menjadi..."

Aktivitas Siswa

Aplikasi "Light Switch"

Deskripsi:
Kita akan membuat aplikasi sederhana seperti saklar lampu. Saat tombol diklik, warna latar belakang layar akan berubah dari **putih ke hitam**, atau sebaliknya.

Langkah-langkah:

1. Buka App Inventor (atau platform coding lainnya).
2. Buat satu **Button**, beri nama misalnya "btnSaklar".
3. Atur warna awal layar menjadi **putih**.
4. Klik bagian **Blocks** (kode).
5. Tambahkan blok kode seperti ini:
    - Saat tombol "btnSaklar" diklik:
        - Jika warna latar sekarang **putih**, ubah jadi **hitam**.
        - Kalau bukan putih (berarti hitam), ubah jadi **putih**.

Aplikasi "Fun Fact Generator"

Deskripsi:
Aplikasi ini akan menampilkan **fakta menarik** yang berbeda setiap kali tombol diklik, seperti:

- "Gajah tidak bisa lompat"
- "Air bisa memotong baja"
- Dan lainnya...

Langkah-langkah (versi sederhana):

1. Buat satu **Button** (misalnya "btnFakta").
2. Buat beberapa **Label** (misalnya "LabelFakta1", "LabelFakta2", dll), masing-masing diisi satu fakta.
3. Atur semua label menjadi **tidak terlihat** saat awal aplikasi dijalankan.
4. Di bagian kode (Blocks):
    - Saat tombol diklik, ubah salah satu label menjadi **terlihat**, dan sembunyikan yang lainnya.
    - Kamu bisa pakai **angka acak (random)** untuk memilih label mana yang ditampilkan (opsional, lebih lanjut di bab berikutnya).

Aplikasi "Goodbye Button"

Deskripsi:
Tombol ini akan mengganti teks menjadi "Goodbye!" dan memutar suara saat diklik.

Langkah-langkah:

1. Buat satu **Button** bernama "btnGoodbye".
2. Buat satu **Label** untuk menampilkan teks.
3. Tambahkan file suara perpisahan (misalnya "goodbye.mp3").
4. Tambahkan komponen **Sound** dan upload suara tadi.
5. Di bagian kode:
    - Saat tombol diklik:
        - Ubah teks label menjadi "Goodbye!"
        - Jalankan suara menggunakan komponen **Sound**.

Aplikasi "Hello World with Sound"

Deskripsi:
Aplikasi yang menampilkan teks **"Hello World!"** dan memutar suara ketika tombol ditekan.

Langkah-langkah:

1. Buat satu **Button**, bisa dinamai "btnHello".
2. Buat satu **Label**, tulis "Hello World!" di dalamnya.
3. Tambahkan komponen **Sound**, lalu upload suara (misalnya "hello.mp3").
4. Di bagian kode:
    - Saat tombol di klik:
        - Putar suara menggunakan komponen Sound.

Rangkuman

Aplikasi yang **interaktif** di *App Inventor* dirancang untuk **merespons aksi pengguna**, misalnya saat **tombol diklik**. Untuk menangani aksi ini, kita menggunakan **blok *event handler*** seperti `when Button1.Click do`. Blok ini akan menjalankan perintah tertentu ketika suatu *event* terjadi.

Setelah *event* terdeteksi, kita bisa **memberikan respons**, seperti **mengubah teks** pada label dengan `set Label1.Text to`, atau **memutar suara** menggunakan `call Sound1.Play`. Untuk menampilkan teks, kita memakai **blok string**—yakni teks yang **ditulis dalam tanda kutip** dan ditampilkan lewat **blok berwarna pink**.

Kombinasi antara **blok *event*** dan **blok pengatur properti** memungkinkan kita membuat aplikasi yang tidak hanya fungsional, tapi juga terasa **hidup dan menyenangkan**. Interaksi sederhana ini adalah **dasar penting** sebelum membangun fitur-fitur yang lebih kompleks.

Latihan Soal

A. Benar atau Salah (5 Soal)

1. Ketika tombol diklik, blok `when Button1.Click do` akan dijalankan.
   Benar

2. Blok `set Label1.Text to` digunakan untuk mengganti warna latar belakang layar.
   Salah

3. Blok teks seperti `"Hello World"` harus dimasukkan dalam tanda kutip.
   Benar

4. Aplikasi tidak dapat menampilkan suara jika tidak memiliki Label.
   Salah

5. `Sound1.Play` digunakan untuk memutar suara di aplikasi.
   Benar

B. Pilihan Ganda (10 Soal)

6. Blok apa yang digunakan untuk merespons klik pada tombol?
   A. set Label.Text to
   B. when Button.Click do
   C. call Sound.Play
   D. set Screen.BackgroundColor to
   Jawaban: B

7. Apa fungsi dari `set Label1.Text to "Hello World!"`?
   A. Mengubah ukuran label
   B. Menghapus teks pada Label1
   C. Menambahkan suara ke Label
   D. Mengganti teks Label1 menjadi "Hello World!"
   Jawaban: D

8. Untuk menampilkan tulisan, blok teks harus...
   A. Berwarna merah
   B. Dalam tanda kurung
   C. Diapit tanda kutip
   D. Dimulai dengan angka
   Jawaban: C

9. Jika ingin aplikasi mengeluarkan suara saat tombol ditekan, perlu menambahkan komponen...
   A. Image

B. Sound
C. Clock
D. Notifier
Jawaban: B

10. Bagian mana yang disebut **event handler**?
    A. `call Sound1.Play`
    B. `set Label1.Text to "Hai!"`
    C. `when Button1.Click do`
    D. `"Hello"`
    Jawaban: C

11. Saat tombol ditekan, Label1 menunjukkan "Goodbye!". Blok yang benar adalah...
    A. `set Button1.Text to "Goodbye!"`
    B. `when Label1.Click do set Text to "Goodbye!"`
    C. `set Label1.Text to "Goodbye!"` di dalam `when Button1.Click do`
    D. `set Label1.Text to Label2.Text`
    Jawaban: C

12. Jika kita ingin Label menampilkan "Aku Hebat!", kita harus mengetik...
    A. `set Label.Text to Aku Hebat!`
    B. `set Label1.Text to "Aku Hebat!"`
    C. `when Label.Click set to "Aku Hebat!"`
    D. `set Button1.Text to "Aku Hebat!"`
    Jawaban: B

13. Untuk membuat aplikasi menampilkan tulisan saat tombol diklik, kita perlu...
    A. Menghapus semua blok
    B. Menambahkan list
    C. Menggunakan event dan setter
    D. Mengatur clock
    Jawaban: C

14. Saat pengguna mengetuk tombol "Like", itu disebut sebagai...
    A. Teks
    B. Event
    C. Suara
    D. Property
    Jawaban: B

15. Apa tujuan dari blok `call Sound1.Play`?
    A. Mengganti teks
    B. Menjalankan animasi
    C. Memutar suara
    D. Menyimpan file
    Jawaban: C

C. Isian Singkat (5 Soal)

16. Blok yang digunakan untuk menangani kejadian tombol ditekan adalah `__________`.
    Jawaban: when Button1.Click do

17. Untuk mengganti teks Label1 menjadi “Hello!”, kita gunakan blok `__________`.
    Jawaban: set Label1.Text to "Hello!"

18. Komponen yang digunakan untuk memutar suara dalam aplikasi adalah `__________`.
    Jawaban: Sound

19. Saat Label.Text ingin diubah, kita membutuhkan blok `set __________`.
    Jawaban: Label1.Text

20. Teks yang ditampilkan di Label harus ditulis dalam tanda `__________`.
    Jawaban: kutip dua (" ")

D. Soal Explorasi

Soal:

Bayangkan kamu membuat aplikasi dengan satu tombol dan satu label. Saat tombol diklik pertama kali, label menampilkan "Selamat Pagi". Saat diklik kedua kali, label berubah menjadi "Selamat Siang". Jelaskan bagaimana kamu bisa membuat fungsi ini. Blok apa saja yang kamu butuhkan?

Soal:

Kamu ingin membuat aplikasi yang menampilkan fakta menarik setiap kali tombol diklik. Ada 3 fakta yang ingin ditampilkan secara bergantian. Jelaskan logika dan langkah-langkah membuat aplikasi ini agar teks label berubah setiap kali tombol ditekan.

Soa:

Dalam sebuah aplikasi, kamu ingin menambahkan efek suara ketika tombol ditekan, dan juga mengubah teks Label dari “Halo” menjadi “Halo Dunia”. Jelaskan susunan blok yang kamu butuhkan, komponen yang digunakan, dan urutan eksekusi yang terjadi ketika tombol diklik.

Tugas Proyek

Proyek 1: Aplikasi “Hello World!” Interaktif

Deskripsi:
Membuat aplikasi yang akan menampilkan pesan “Hello World!” saat tombol diklik.

Langkah-langkah:

1. **Desain Tampilan**: Tambahkan Label dan Button di layar.
2. **Ganti Properti**:
    - Label: Teks diubah menjadi "...".
    - Button: Teks diubah menjadi "Tampilkan Pesan!".
3. **Atur Blok Kode**:
    - Gunakan blok `when Button1.Click do`.
    - Tambahkan blok `set Label1.Text to`.
    - Tambahkan blok teks berisi "Hello World!".
4. **Uji Coba di Ponsel** dengan MIT AI Companion.

Tujuan: Memahami cara kerja event handling dasar dan mengubah tampilan teks.

Proyek 2: Aplikasi "Tebak Nama Saya"

Deskripsi:
Pengguna akan menekan tombol dan aplikasi akan menampilkan nama yang sudah ditentukan.

Langkah-langkah:

1. Tambahkan Label dan Button ke layar.
2. Ganti teks Button menjadi "Siapa Aku?".
3. Gunakan blok `when Button1.Click do`.
4. Tambahkan blok `set Label1.Text to "Namaku Aplikasi!"`.

Variasi Menantang:

- Tambahkan beberapa tombol dengan nama berbeda.
- Tampilkan nama acak dari daftar nama.

Tujuan: Mempraktekkan penggunaan teks dinamis dan interaksi sederhana.

Proyek 3: Aplikasi "Motivasi Hari Ini"

Deskripsi:
Setiap kali tombol ditekan, aplikasi menampilkan pesan motivasi berbeda.

Langkah-langkah:

1. Tambahkan Label dan Button.
2. Siapkan beberapa blok teks berisi pesan motivasi.
3. Gunakan logika acak (random) untuk memilih salah satu pesan.
4. Gabungkan dengan blok `set Label1.Text to`.

Saran Tambahan:

- Tambahkan gambar latar belakang agar lebih menarik.

- Buat pesan-pesan dengan emoji 🎯🌟💪

Tujuan: Mengenal variasi output dan membangkitkan kreativitas siswa.

Proyek 4: Aplikasi “Hitung Klik Saya!”

Deskripsi:
Aplikasi menghitung berapa kali tombol ditekan.

Langkah-langkah:

1. Tambahkan Label dan Button.
2. Tambahkan variabel (misalnya: `klik`).
3. Setiap klik tombol akan menambah nilai variabel `klik` dan mengubah teks Label.
4. Gunakan blok `set Label1.Text to` dengan gabungan teks dan angka.

Contoh Output: “Tombol ditekan 3 kali!”

Tujuan: Melatih penggunaan variabel dan penghitungan sederhana.

Proyek 5: Aplikasi “Ucapan Selamat” Berdasarkan Waktu

Deskripsi:
Menampilkan ucapan seperti “Selamat Pagi” atau “Selamat Malam” berdasarkan waktu saat ini.

Langkah-langkah:

1. Tambahkan Label dan Button.
2. Gunakan blok waktu (Clock component).
3. Ambil jam saat ini dan buat logika IF:
    - < 12 : “Selamat Pagi!”
    - 12–17 : “Selamat Siang!”
    - > 17 : “Selamat Malam!”

Tujuan: Memperkenalkan penggunaan kondisi (if-else) dan waktu sistem.

Proyek 6: Aplikasi “Tebakan Konyol”

Deskripsi:
Pengguna menekan tombol dan aplikasi memberikan tebakan lucu seperti “Kamu suka mie ayam!”

Langkah-langkah:

1. Tambahkan Label dan Button.
2. Siapkan daftar teks lucu.

3. Gunakan pilihan acak untuk menampilkan satu teks.

Variasi Menyenangkan:

- Tambahkan suara atau gambar pendukung.

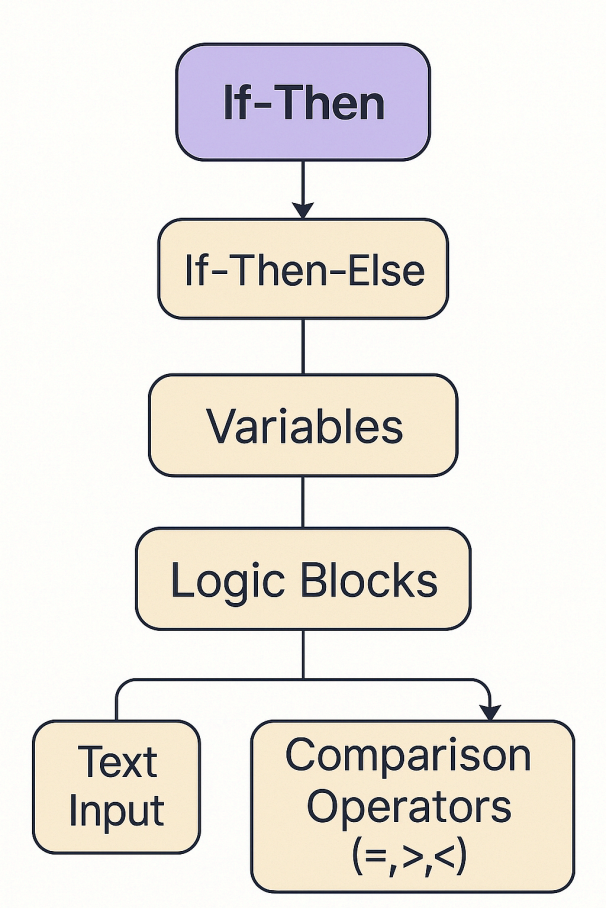
Tujuan: Meningkatkan rasa humor dan kreativitas sambil belajar coding.

BAB 3: Logika Kondisi (If-Then-Else) dan Variabel Dasar

Tujuan Pembelajaran

Siswa mampu menerapkan logika If-Then-Else (kondisi) dan menggunakan variabel dasar dalam program App Inventor 2.

Peta Konsep



Apersepsi

Bayangkan kalian sedang menyeberang jalan:

- **Jika tidak ada mobil**, maka kalian **boleh menyeberang**.
- **Jika ada mobil**, maka kalian **harus menunggu**.

Nah, cara berpikir ini adalah contoh nyata dari logika **"jika-maka-lainnya"** atau dalam istilah pemrograman dikenal sebagai **If-Then-Else**.

Penjelasan Konsep (Teori)

Pada bab ini, kita akan mempelajari dua hal penting dalam pemrograman menggunakan **App Inventor 2**, yaitu **logika kondisi (If-Then-Else)** dan **variabel dasar**. Keduanya adalah pondasi utama dalam membuat program yang bisa berpikir dan mengambil keputusan, layaknya manusia.

Apa Itu Logika Kondisi?

Logika kondisi memungkinkan program **membuat keputusan** berdasarkan suatu situasi atau syarat. Dalam App Inventor, logika ini digunakan dengan blok **if**, **then**, dan **else**.

Jenis-jenis logika kondisi:

- **`if ... then`**
  Artinya: *Jika suatu kondisi terpenuhi, maka lakukan sesuatu.*
  Contoh:

  Jika nilai ujian lebih dari 75, maka tampilkan “Lulus”.



- **`if ... then ... else`**
  Artinya: *Jika suatu kondisi terpenuhi, lakukan A. Jika tidak, lakukan B.*
  Contoh:

  Jika hujan, maka bawa payung. Jika tidak hujan, tidak perlu bawa payung.



Operator Perbandingan

Agar program bisa mengevaluasi kondisi, kita butuh **operator perbandingan**, yaitu simbol yang digunakan untuk membandingkan dua nilai. Beberapa operator yang sering digunakan:

- `=` → sama dengan (equal to)
- `>` → lebih besar dari (greater than)
- `<` → lebih kecil dari (less than)
- `≠` → tidak sama dengan (not equal to)

Contoh penggunaan:

> Jika skor pemain **> 100**, maka tampilkan pesan “Kamu menang!”

Apa Itu Variabel?

Dalam dunia pemrograman, **variabel** adalah seperti **wadah atau kotak penyimpanan** sementara untuk data. Variabel bisa menyimpan berbagai jenis informasi seperti angka, teks, atau nilai logika (benar/salah).

Blok-blok yang digunakan:

- **`initialize global [nama] to`**
  → Untuk **membuat dan memberi nilai awal** ke sebuah variabel.
  Contoh: `initialize global skor to 0`



- **`get global [nama]`**
  → Untuk **mengambil nilai** dari variabel yang sudah dibuat.
  Contoh: menampilkan skor saat ini ke layar.



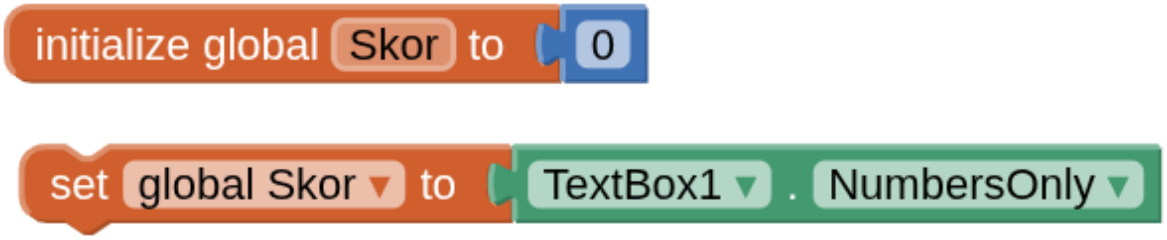
- **`set global [nama] to`**
  → Untuk **mengubah nilai** dalam variabel.

Contoh: `set global skor to skor + 10`



Komponen TextBox

Komponen **TextBox** di App Inventor digunakan untuk **menerima input dari pengguna**, misalnya nama, umur, atau jawaban kuis. Nilai yang dimasukkan bisa digunakan dalam logika kondisi atau disimpan dalam variabel.



Konversi Tipe Data Secara Otomatis (Implisit)

Kadang, kalian memasukkan angka ke dalam TextBox (padahal input TextBox adalah **teks**). Untungnya, App Inventor cukup **cerdas** untuk secara otomatis mencoba mengubah teks menjadi angka jika diperlukan—misalnya dalam perhitungan matematika atau perbandingan nilai.

Contoh:

> Jika kalian memasukkan "100" di TextBox, dan program membandingkan apakah nilai itu lebih besar dari 50, App Inventor akan mengubah "100" menjadi angka 100 secara otomatis.

Kesimpulan

Dengan memahami konsep **logika kondisi** dan **variabel**, kalian bisa:

- Membuat program yang bisa **berpikir dan memilih tindakan**
- Menyimpan dan mengolah **data pengguna**
- Menciptakan aplikasi yang **lebih interaktif dan cerdas**

Contoh Kasus atau Ilustrasi

Kasus 1: Mesin Penjual Otomatis

Ilustrasi:
Bayangkan kalian sedang berdiri di depan mesin penjual minuman otomatis. Kalian ingin membeli teh botol seharga Rp5.000. Lalu kalian memasukkan uang.

Logika If-Then-Else:

- Jika uang yang dimasukkan ≥ 5000, maka mesin mengeluarkan teh botol.
- Jika uang yang dimasukkan < 5000, maka mesin menampilkan pesan: "Uang tidak cukup!"

Variabel yang Digunakan:

- `uangMasuk` (nilai dari TextBox)
- `hargaTeh` (nilai tetap: 5000)

Kasus ini menunjukkan bagaimana keputusan dibuat berdasarkan **syarat tertentu** dan **nilai yang tersimpan dalam variabel**.

Kasus 2: Penentu Lulus Tidaknya Ujian

Ilustrasi:
Kalian baru saja selesai ujian matematika. Nilai kalian akan diproses oleh aplikasi.

Logika If-Then-Else:

- Jika nilai ≥ 75, maka tampilkan pesan "Selamat, kamu lulus!"
- Jika nilai < 75, maka tampilkan "Maaf, kamu belum lulus. Tetap semangat!"

Variabel yang Digunakan:

- `nilaiUjian` (diinput dari TextBox)

Ini mengajarkan kalian bagaimana program bisa memberikan respon berbeda **berdasarkan input nilai**.

Kasus 3: Lampu Otomatis di Ruangan

Ilustrasi:
Bayangkan sebuah ruang kelas canggih yang lampunya bisa menyala atau mati secara otomatis.

Logika If-Then-Else:

- Jika sensor mendeteksi orang masuk, maka lampu menyala.
- Jika tidak ada orang, maka lampu mati.

Variabel yang Digunakan:

- `adaOrang` (bernilai true/false)

Kalian bisa melihat bagaimana logika **benar/salah (boolean)** digunakan untuk mengatur aksi otomatis.

Kasus 4: Game Poin Pemain

Ilustrasi:
Dalam sebuah game sederhana, kalian mendapatkan poin setiap kali menjawab dengan benar.

Logika If-Then-Else:

- Jika skor pemain > 100, tampilkan "Kamu menang!"
- Jika skor ≤ 100, tampilkan "Terus semangat, tambah poinmu!"

Variabel yang Digunakan:

- `skor` (berubah setiap kali menjawab)

Dari sini, kalian bisa memahami bagaimana **angka berubah-ubah disimpan dalam variabel dan digunakan untuk mengambil keputusan.**

Kasus 5: Pengingat Membawa Payung

Ilustrasi:
Kalian akan berangkat ke sekolah dan melihat ramalan cuaca dari aplikasi.

Logika If-Then-Else:

- Jika cuaca = "hujan", maka tampilkan "Jangan lupa bawa payung!"
- Jika cuaca ≠ "hujan", maka tampilkan "Cuaca cerah, tetap semangat ke sekolah!"

Variabel yang Digunakan:

- `cuacaHariIni` (bisa diinput manual atau di-set oleh sistem)

Kalian belajar bagaimana program bisa **berinteraksi dengan data dari luar** dan **mengambil keputusan berdasar teks**.

Kasus 6: Ucapan Selamat Ulang Tahun

Ilustrasi:
Aplikasi akan memberikan ucapan khusus jika hari ini adalah tanggal ulang tahunmu.

Logika If-Then-Else:

- Jika tanggal hari ini = tanggal lahir kamu, tampilkan "Selamat ulang tahun!"
- Jika tidak, tampilkan "Semoga harimu menyenangkan!"

Variabel yang Digunakan:

- `tanggalHariIni`, `tanggalLahir`

Ini menunjukkan bagaimana program bisa **membandingkan dua nilai** dan memberikan hasil **yang personal dan menyenangkan**.

Setiap kasus di atas menyajikan **pengalaman nyata yang dekat dengan kehidupan kalian**. Dengan memahami bagaimana kondisi dan variabel bekerja, kalian akan semakin jago membuat aplikasi yang **pintar, interaktif, dan menyenangkan**! Siap menjelajah dunia pemrograman?

Contoh Soal dan Pembahasan

Soal 1: Siapakah yang Lulus?

Bayangkan kamu sedang membuat aplikasi pengumuman kelulusan. Nilai ujian disimpan dalam variabel `nilaiUjian`.

Jika nilainya lebih dari 75, maka tampilkan “Selamat, kamu lulus!”. Jika tidak, tampilkan “Maaf, kamu belum lulus.”

Pertanyaannya:
Bagaimana logika If-Then-Else dalam blok App Inventor untuk menyelesaikan ini?

Pembahasan:

Gunakan blok:

```
if nilaiUjian > 75
  then tampilkan "Selamat, kamu lulus!"
  else tampilkan "Maaf, kamu belum lulus."
```

Penjelasan:

- Kita membandingkan `nilaiUjian` dengan 75 menggunakan operator `>`.
- Jika benar, maka tampilkan pesan lulus, jika salah maka tampilkan pesan tidak lulus.

Soal 2: Game Tebak Angka

Kamu membuat game tebak angka. Pemain memasukkan angka ke dalam `TextBox`, dan angka yang harus ditebak disimpan dalam variabel `angkaBenar = 7`.

Pertanyaannya:
Tulislah logika If-Then-Else untuk kasus berikut:

- Jika angka tebakan sama dengan `angkaBenar`, tampilkan “Tebakanmu benar!”
- Jika tidak sama, tampilkan “Coba lagi!”

Pembahasan:

```
if inputTextBox = angkaBenar
  then tampilkan "Tebakanmu benar!"
  else tampilkan "Coba lagi!"
```

Catatan:

- App Inventor akan secara otomatis mengubah input dari TextBox (teks) menjadi angka agar bisa dibandingkan.

Soal 3: Bonus Skor!

Dalam permainan, setiap kali pemain menjawab benar, skor ditambah 10. Variabel `skor` awalnya adalah 0.

Pertanyaannya:
Bagaimana cara membuat logika penambahan skor menggunakan variabel?

Pembahasan:

Gunakan blok:

```
initialize global skor to 0
...
set global skor to skor + 10
```

Penjelasan:

- Pertama, buat variabel `skor` dan beri nilai awal 0.
- Setelah menjawab benar, gunakan `set global skor to skor + 10` untuk menambah nilainya.

Soal 4: Hujan atau Cerah?

Kamu membuat aplikasi cuaca. Input TextBox diisi dengan kata “hujan” atau “cerah”. Jika isinya “hujan”, tampilkan “Bawa payung”. Jika “cerah”, tampilkan “Jangan lupa pakai sunscreen”.

Pertanyaannya:
Tulislah logika If-Then-Else-nya!

Pembahasan:

```
if TextBox = "hujan"
  then tampilkan "Bawa payung"
else if TextBox = "cerah"
  then tampilkan "Jangan lupa pakai sunscreen"
```

Penjelasan:

- Menggunakan dua kondisi: satu untuk hujan, satu untuk cerah.
- Bisa menggunakan `if ... then ... else if ... then ...`.

Soal 5: Umurmu Cukup?

Aplikasi akan mengecek apakah umur seseorang (dari TextBox) cukup untuk mendaftar kursus (minimal 13 tahun).

Pertanyaannya:
Buat logika kondisi If-Then-Else yang mengecek apakah input cukup umur atau tidak.

Pembahasan:

```
if umur >= 13
  then tampilkan "Kamu bisa mendaftar!"
  else tampilkan "Maaf, umurmu belum cukup."
```

Catatan:

- App Inventor akan mengkonversi input dari teks ke angka agar bisa dibandingkan dengan operator `>=`.

Fakta Menarik / Fun Facts

- Semua aplikasi yang membutuhkan keputusan (misalnya, game, sistem login) menggunakan logika `if-then-else`!
- Kalkulator sederhana juga menggunakan variabel untuk menyimpan angka dan hasil.

Tips Belajar atau Tips Cepat Menghafal

Tips Singkat:

- "Kondisi hanya dua: **True** atau **False**."
- "**Variabel** = kotak penyimpan data."

Mind Mapping Mini:



Cerita Mini:

Dina membuat game tebak angka. Komputer memilih angka 1–10. Dina menebak angka:

- **Jika benar** → "Tebakan tepat!"
- **Jika lebih kecil** → "Terlalu kecil!"
- **Jika lebih besar** → "Terlalu besar!"

Aktivitas Siswa

Aktivitas 1: Game Tebak Angka

Deskripsi Aktivitas:

Di game ini, **komputer memilih satu angka rahasia antara 1 sampai 10**. Tugasmu adalah **menebak angka itu**. Setelah kamu mengetik tebakan, **komputer akan memberi tahu** apakah tebakanmu **terlalu kecil**, **terlalu besar**, atau **tepat**.

Langkah-langkah:

1. Jalankan aplikasi "Game Tebak Angka".
2. Komputer akan memilih satu angka rahasia secara otomatis (1 - 10).

3. Kamu ketikkan angka tebakanmu di kotak teks (`TextBox`).
4. Tekan tombol `Cek` atau `Tebak`.
5. Komputer akan memberi respons:
    - “Terlalu kecil” → kalau tebakanmu lebih kecil.
    - “Terlalu besar” → kalau tebakanmu lebih besar.
    - “Tepat!” → kalau tebakanmu benar.
6. Coba terus sampai kamu menebak dengan tepat!

Pseudocode:

```
Saat aplikasi dimulai:
  angkaRahasia ← angka acak dari 1 sampai 10

Saat tombol "Tebak" ditekan:
  tebakan ← ambil nilai dari TextBox
  Jika tebakan < angkaRahasia:
     tampilkan "Terlalu kecil"
  Jika tebakan > angkaRahasia:
     tampilkan "Terlalu besar"
  Jika tebakan = angkaRahasia:
     tampilkan "Tepat!"
```

Aktivitas 2: Kalkulator Mini

Deskripsi Aktivitas:

Kamu akan membuat **kalkulator penjumlahan sederhana**. Kamu cukup memasukkan dua angka, lalu menekan tombol “Jumlahkan” untuk melihat hasilnya.

Langkah-langkah:

1. Jalankan aplikasi "Kalkulator Mini".
2. Masukkan angka pertama ke `TextBox1`.
3. Masukkan angka kedua ke `TextBox2`.
4. Tekan tombol “Jumlahkan”.
5. Lihat hasil penjumlahan di label hasil (`LabelHasil`).

Pseudocode:

```
Saat tombol "Jumlahkan" ditekan:
  angka1 ← ambil dari TextBox1
  angka2 ← ambil dari TextBox2
  hasil ← angka1 + angka2
  tampilkan hasil di LabelHasil
```

Aktivitas 3: Kalkulator Pengurangan

Deskripsi Aktivitas:

Hampir sama seperti Kalkulator Mini, tapi sekarang kamu akan menghitung **selisih dua angka** (pengurangan).

Langkah-langkah:

1. Masukkan angka pertama ke `TextBox1`.
2. Masukkan angka kedua ke `TextBox2`.
3. Tekan tombol “Kurangkan”.
4. Hasil pengurangan akan muncul di `LabelHasil`.

Pseudocode:

```
Saat tombol "Kurangkan" ditekan:
  angka1 ← ambil dari TextBox1
  angka2 ← ambil dari TextBox2
  hasil ← angka1 - angka2
  tampilkan hasil di LabelHasil
```

Aktivitas 4: Game Tebak Ganjil atau Genap

Deskripsi Aktivitas:

Komputer akan memilih angka acak antara 1 sampai 20. Kamu harus **menebak** apakah angka itu **ganjil atau genap**. Komputer akan memberitahu apakah tebakanmu benar atau salah.

Langkah-langkah:

1. Komputer memilih angka rahasia dari 1 sampai 20.
2. Kamu pilih “Ganjil” atau “Genap” (misalnya dengan tombol atau radio button).
3. Tekan tombol “Cek Jawaban”.
4. Komputer tampilkan:
    - Angka sebenarnya.
    - Apakah tebakanmu benar atau salah.

Pseudocode:

```
Saat tombol "Cek Jawaban" ditekan:
  angka ← angka acak dari 1 sampai 20
  tebakan ← pilihan pengguna (Ganjil atau Genap)

  Jika angka mod 2 = 0 maka:
     jenis ← "Genap"
  jika tidak:
     jenis ← "Ganjil"
```

```
tampilkan angka sebenarnya

Jika tebakan = jenis:
    tampilkan "Tebakanmu benar!"
jika tidak:
    tampilkan "Tebakanmu salah!"
```

Rangkuman

Logika *If-Then-Else* memungkinkan aplikasi **mengambil keputusan secara otomatis** berdasarkan kondisi tertentu—misalnya, memilih antara “Ya” atau “Tidak”. Kemampuan ini **sangat penting dalam membangun fitur interaktif** seperti **game**, **form login**, atau **kalkulator**.

Di sisi lain, **variabel** berfungsi sebagai **wadah penyimpanan sementara** untuk data seperti angka, teks, atau input dari pengguna. Variabel bisa diibaratkan sebagai **"kotak" di dalam aplikasi** yang menyimpan nilai yang bisa berubah-ubah.

Dalam *App Inventor*, pemahaman tentang **blok logika**, **operator perbandingan**, dan **manipulasi variabel** menjadi **pondasi utama** untuk membuat aplikasi yang menarik dan interaktif. Dengan pendekatan *visual programming* yang digunakan oleh App Inventor, siswa bisa **lebih mudah memahami cara kerja logika dalam pemrograman** secara nyata dan menyenangkan.

Latihan Soal

A. Benar atau Salah (5 Soal)

1. Operator “greater than” ditulis dengan simbol `<`.
   Jawaban: Salah

2. Variabel dalam App Inventor digunakan untuk menyimpan data sementara.
   Jawaban: Benar

3. Blok `if ... then` digunakan untuk memilih di antara dua kemungkinan.
   Jawaban: Salah

4. Blok `get global` digunakan untuk mengambil nilai dari variabel.
   Jawaban: Benar

5. Teks "7" dan angka 7 akan dianggap sama oleh App Inventor saat dibandingkan dengan `=`.
   Jawaban: Benar

B. Pilihan Ganda (10 Soal)

6. Manakah blok yang digunakan untuk membuat variabel baru?
   A. `get global`
   B. `set global`
   C. `initialize global`
   D. `define variable`
   Jawaban: C

7. Apa arti dari simbol `=` dalam logika kondisi?
   A. Menambahkan
   B. Menyimpan nilai
   C. Membandingkan apakah sama
   D. Menampilkan hasil
   Jawaban: C

8. Blok `if ... then ... else` digunakan ketika:
   A. Kita hanya punya satu kondisi
   B. Kita tidak butuh variabel
   C. Kita ingin menangani dua kemungkinan
   D. Kita ingin mengulang kode
   Jawaban: C

9. Fungsi dari TextBox dalam App Inventor adalah:
   A. Menyimpan angka
   B. Menghitung hasil
   C. Menampilkan label
   D. Menerima input teks dari pengguna
   Jawaban: D

10. Apa hasil dari kondisi `if TextBox1.Text > 50` jika pengguna mengetik "75"?
    A. Salah
    B. Benar
    C. Error
    D. Tidak pasti
    Jawaban: B

11. Bagaimana cara mengubah nilai variabel bernama `score` menjadi 10?
    A. `get global score = 10`
    B. `initialize global score to 10`
    C. `set global score to 10`
    D. `define score as 10`
    Jawaban: C

12. Untuk membandingkan apakah dua nilai **tidak** sama, kita bisa menggunakan:
    A. =

B. ≠
C. >
D. <
Jawaban: B

13. Jika kita ingin menampilkan "Selamat" jika umur lebih dari atau sama dengan 17, kita pakai:
    A. `if umur > 17`
    B. `if umur >= 17`
    C. `if umur < 17`
    D. `if umur = 17`
    Jawaban: B

14. Untuk mengambil isi dari variabel bernama `nama`, kita gunakan:
    A. `set global nama`
    B. `get global nama`
    C. `initialize global nama`
    D. `update global nama`
    Jawaban: B

15. Variabel yang belum diisi akan:
    A. Menampilkan 0
    B. Menampilkan "null"
    C. Tidak bisa digunakan
    D. Error jika diakses
    Jawaban: D

C. Isian Singkat (5 Soal)

16. Blok untuk membuat variabel baru dan memberinya nilai disebut ___.
    Jawaban: initialize global

17. Simbol untuk "kurang dari" dalam perbandingan adalah ___.
    Jawaban: <

18. Untuk mengecek apakah isi dari TextBox1 adalah "admin", kita gunakan blok ___.
    Jawaban: if TextBox1.Text = "admin"

19. Saat kita ingin menampilkan hasil ke pengguna, kita gunakan komponen ___.
    Jawaban: Label

20. Jika pengguna memasukkan angka di TextBox, maka data masuk sebagai tipe ___.
    Jawaban: teks

D. Soal Explorasi

Soal:

Bayangkan kamu membuat aplikasi untuk memeriksa apakah seseorang bisa ikut lomba. Syaratnya, umur minimal 12 tahun. Buatlah logika If-Then-Else yang akan menampilkan "Boleh ikut" jika pengguna mengisi umur 12 atau lebih, dan "Belum boleh ikut" jika kurang dari itu.

Soal:

Kamu membuat aplikasi ramalan sederhana seperti “Magic 8 Ball”. Ketika tombol ditekan, aplikasi memilih angka acak dari 1 sampai 3. Jika hasilnya 1, tampilkan “Yes”, jika 2 tampilkan “No”, dan jika 3 tampilkan “Maybe”. Tulis logika if-then-else untuk menangani semua kemungkinan tersebut.

Soal:

Buatlah sebuah simulasi mini kalkulator. Pengguna memasukkan dua angka di TextBox1 dan TextBox2. Ketika tombol dijalankan, aplikasi menambahkan kedua angka dan menampilkan hasilnya di Label. Tulislah urutan logika yang tepat, termasuk cara mengambil data, mengubahnya ke angka, menjumlahkannya, dan menampilkannya.

Tugas Proyek

Kasus 1: Aplikasi Kuis Pintar

Deskripsi:
Buat aplikasi kuis sederhana yang menampilkan pertanyaan dan memeriksa jawaban pengguna. Jika jawaban benar, skor bertambah. Jika salah, tampilkan pesan semangat.

Langkah-Langkah:

- Siapkan komponen **Label**, **TextBox**, dan **Button**.
- Gunakan blok **if...then...else** untuk memeriksa apakah jawaban benar.
- Gunakan **variabel** untuk menyimpan skor dan tampilkan di layar.
- Tambahkan beberapa soal untuk membuat kuis lebih menantang.

Kasus 2: Aplikasi "Boleh Menyeberang atau Tidak?"

Deskripsi:
Simulasi lampu lalu lintas: pengguna mengetik warna lampu, dan aplikasi memberi keputusan (boleh jalan atau tidak).

Langkah-Langkah:

- Buat TextBox untuk input warna lampu (merah, kuning, hijau).
- Gunakan logika kondisi **if...then...else** untuk memutuskan:
    - Hijau → "Silakan menyeberang"
    - Kuning → "Bersiap-siap"
    - Merah → "Berhenti!"

- Gunakan blok **set label** untuk menampilkan keputusan.

Kasus 3: Kalkulator Nilai Akhir

Deskripsi:
Siswa memasukkan nilai ujian. Jika nilai ≥ 75, tampilkan "Lulus", jika tidak, tampilkan "Remedial".

Langkah-Langkah:

- Tambahkan TextBox untuk input nilai.
- Gunakan **operator perbandingan** (>=) dan blok **if...then...else** untuk mengevaluasi.
- Buat variabel **status** untuk menyimpan hasil dan tampilkan dengan Label.

Kasus 4: Tebak Angka Rahasia

Deskripsi:
Aplikasi menyimpan angka rahasia (misalnya 7). Jika pengguna menebak benar, tampilkan "Tepat!". Jika tidak, beri petunjuk (lebih besar/kecil).

Langkah-Langkah:

- Gunakan variabel global `angkaRahasia` dan tetapkan nilainya.
- Buat TextBox untuk tebakannya.
- Gunakan blok **if...then...else** dan **operator perbandingan** untuk mengevaluasi.
- Beri respon sesuai hasil tebakan.

Kasus 5: Hitung Usia dan Kategori

Deskripsi:
Pengguna memasukkan tahun lahir, dan aplikasi menghitung usia serta menentukan kategori (anak-anak, remaja, dewasa).

Langkah-Langkah:

- Buat TextBox untuk tahun lahir.
- Hitung usia: tahun sekarang - tahun lahir.
- Gunakan logika kondisi:
  - < 12 → "Anak-anak"
  - 12–17 → "Remaja"
  - ≥ 18 → "Dewasa"
- Tampilkan hasil menggunakan Label.

Proyek 6: Game "Skor Naik-Turun"

Deskripsi:
Game sederhana di mana tombol "Tambah Skor" menaikkan skor, dan tombol "Kurangi Skor" menurunkan. Jika skor mencapai 100, tampilkan "Menang!".

Langkah-Langkah:

- Gunakan variabel `skor` dan inisialisasi ke 0.
- Tambahkan dua tombol: satu untuk tambah, satu untuk kurangi skor.
- Gunakan **set global skor to skor ± 10**.
- Tambahkan kondisi **if skor ≥ 100** untuk menampilkan pesan kemenangan.

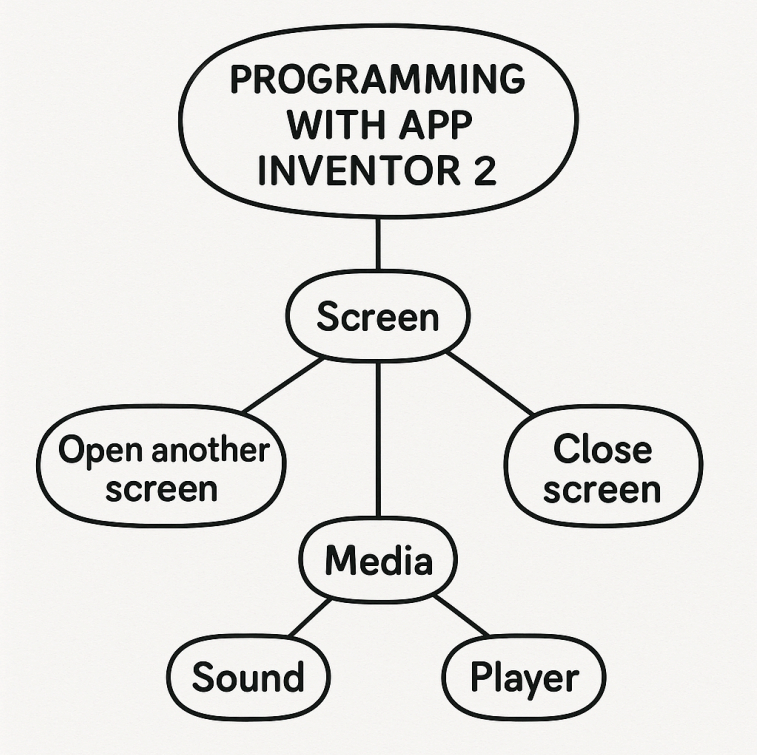
Proyek-proyek ini dirancang agar **menantang**, **seru**, dan tetap **mendidik**, serta mendorong siswa untuk **berpikir logis dan kreatif** dalam memanfaatkan logika kondisi dan variabel. Inilah awal petualangan seru kalian sebagai programmer muda! 🚀

BAB 4: Interaksi Antar Layar dan Suara

Tujuan Pembelajaran

Peserta didik mampu membuat aplikasi dengan lebih dari satu layar dan menggunakan elemen suara/media

Peta Konsep



Apersepsi

Pernahkah kalian melihat aplikasi yang punya banyak halaman, seperti aplikasi belanja online atau game? Bagaimana cara berpindah dari satu halaman ke halaman lain?

Penjelasan Konsep (Teori)

Pernahkah kalian menggunakan aplikasi seperti game atau toko online yang memiliki banyak halaman? Misalnya, ketika membuka aplikasi game, kalian bisa berpindah dari layar menu utama ke layar level permainan, lalu ke layar skor. Semua itu disebut sebagai **aplikasi multi-layar**.

Di bab ini, kita akan belajar bagaimana membuat aplikasi seperti itu, termasuk bagaimana **berpindah antar layar** dan bagaimana **menggunakan suara** dalam aplikasi agar lebih hidup dan menarik!

Apa Itu Aplikasi Multi-Layar?

Aplikasi multi-layar adalah aplikasi yang memiliki lebih dari satu **Screen** atau layar. Setiap layar bisa memiliki tampilan dan fungsi yang berbeda. Misalnya:

- Layar 1: Menu utama
- Layar 2: Halaman permainan
- Layar 3: Halaman skor

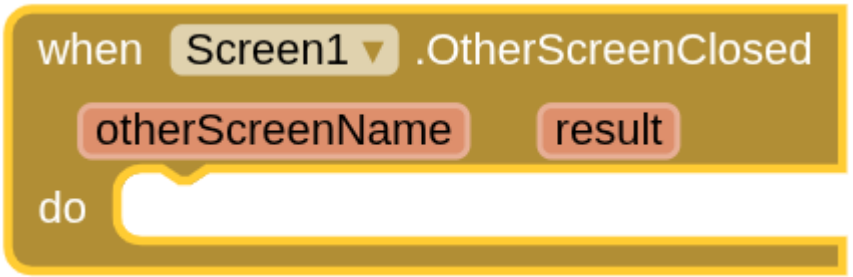
Bayangkan layar-layar ini seperti halaman-halaman di sebuah buku cerita digital. Kita bisa membuka dan menutup halaman sesuai kebutuhan.

Blok yang Digunakan untuk Navigasi Antar Layar:

1. `open another screen screenName`
    - Fungsi: Membuka layar baru berdasarkan nama layar yang diinginkan.
    - Contoh: Jika kamu ingin berpindah dari layar menu ke layar permainan, kamu bisa menggunakan blok ini.



2. `close screen`
    - Fungsi: Menutup layar yang sedang aktif dan kembali ke layar sebelumnya.
    - Contoh: Ketika permainan selesai dan kamu ingin kembali ke menu utama, gunakan blok ini.



Menambahkan Suara dan Musik ke Aplikasi

Aplikasi akan menjadi lebih hidup jika kita menambahkan suara. Misalnya:

- Efek suara ketika tombol diklik
- Musik latar saat permainan dimulai
- Suara notifikasi saat berhasil atau gagal

Di App Inventor, kita bisa menambahkan **komponen suara** melalui dua cara, tergantung kebutuhan:

1. **Komponen Sound**
    - Cocok untuk efek suara yang pendek dan cepat (misalnya: klik, tepuk tangan, suara lonceng).
    - Ringan dan cepat dimuat.



2. **Komponen Player**
    - Cocok untuk audio atau video berdurasi panjang (misalnya: musik latar, lagu pembuka).
    - Bisa digunakan untuk memutar, menjeda, dan menghentikan suara.

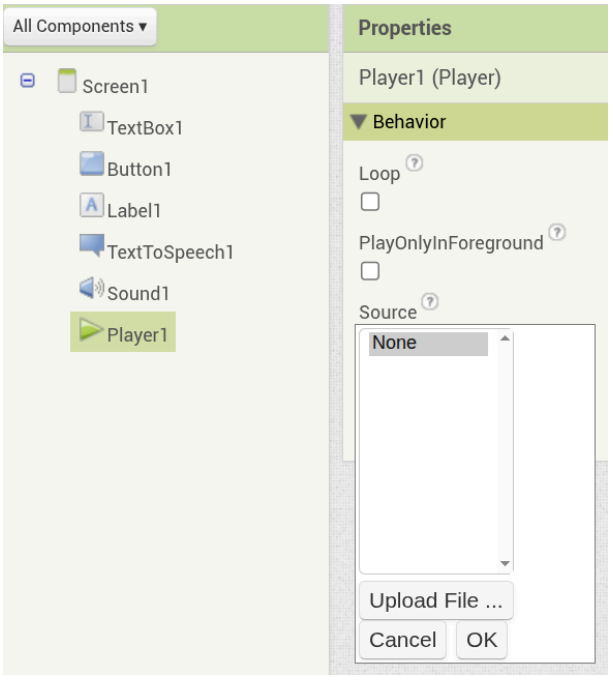


Mengunggah Media ke Aplikasi

Agar aplikasi bisa memutar suara atau menampilkan gambar, kita perlu **mengunggah file media** ke App Inventor.

Langkah-langkah:

- Buka bagian **Designer**
- Klik pada menu **Media**
- Pilih file suara atau gambar dari komputer kalian untuk diunggah



Setelah itu, kita atur **Properti Source** pada komponen suara/gambar untuk menentukan media mana yang akan digunakan.

Kesimpulan

- Aplikasi bisa memiliki banyak layar, dan kita bisa berpindah di antaranya menggunakan blok khusus.
- Suara membuat aplikasi lebih menyenangkan, dan bisa ditambahkan menggunakan komponen Sound atau Player.
- File media harus diunggah terlebih dahulu agar bisa digunakan di aplikasi.

Dengan memahami konsep ini, kalian akan bisa menciptakan aplikasi yang **lebih interaktif, seru, dan profesional!** Siap membuat aplikasi yang bisa "berbicara" dan berpindah layar seperti aplikasi sungguhan? Yuk, lanjut ke praktiknya! 💡🎮📱

Contoh Kasus atau Ilustrasi

Kasus 1: Aplikasi Game Petualangan

Ilustrasi:
Bayangkan kalian membuat game petualangan digital seperti "Jelajah Pulau Harta Karun". Saat pemain membuka aplikasi, mereka akan melihat **Layar Menu**. Setelah menekan tombol “Mulai”, pemain masuk ke **Layar Level Permainan**, dan setelah menyelesaikan tantangan, muncul **Layar Skor** dengan suara drum kemenangan.

Tantangan Seru: Bagaimana kalian mengatur agar pemain bisa berpindah antar layar dan menambahkan efek suara ketika mendapatkan harta?

Kasus 2: Aplikasi Quiz Interaktif

Ilustrasi:
Kalian membuat aplikasi kuis seru berjudul “Si Paling Tahu!”. Di **Layar Utama**, pengguna memilih kategori soal. Setelah itu, pindah ke **Layar Pertanyaan**. Jika jawaban benar, muncul efek suara “Yeay!” dan lanjut ke soal berikutnya. Jika salah, muncul suara buzzer dan kembali ke soal sebelumnya.

Tantangan Seru: Bagaimana caranya agar pengguna bisa berpindah layar sesuai hasil jawabannya, dan bagaimana suara bisa memberikan efek emosi?

Kasus 3: Aplikasi Belanja Mini

Ilustrasi:
Kalian menciptakan “Toko Jajan Digital”. Di **Layar Menu**, pengguna memilih kategori makanan. Saat memilih, pindah ke **Layar Daftar Produk**. Jika menekan "Beli", muncul **Layar Konfirmasi** dengan suara "ting!" khas mesin kasir.

Tantangan Seru: Bisakah kalian mengatur perpindahan antar layar secara logis sesuai proses belanja, dan menambahkan suara yang sesuai di setiap langkah?

Kasus 4: Aplikasi Cerita Bergambar

Ilustrasi:
Buat aplikasi dongeng digital berjudul “Putri dan Burung Emas”. Di setiap layar ada satu halaman cerita. Saat pengguna menekan tombol “Lanjut”, berpindah ke halaman berikutnya. Narasi dongeng diputar sebagai suara latar, dan setiap karakter bisa mengeluarkan suara tertentu saat disentuh.

Tantangan Seru: Bagaimana kalian mengatur agar cerita mengalir dari satu layar ke layar lainnya, sambil menjaga suara tetap menyatu?

Kasus 5: Aplikasi Panduan Eksperimen Sains

Ilustrasi:
Bayangkan kalian membuat “Lab Mini”. Di **Layar Menu**, siswa memilih eksperimen: misalnya “Membuat Gunung Api Mini”. Lalu pindah ke **Layar Panduan Langkah**. Tiap langkah punya efek suara (seperti bunyi letupan saat lava muncul) agar terasa nyata!

Tantangan Seru: Bagaimana mengatur agar suara hanya diputar di langkah tertentu, dan pengguna bisa kembali ke langkah sebelumnya tanpa kehilangan efek suara?

Kasus 6: Aplikasi Kalender Harian Motivasi

Ilustrasi:
Kalian merancang “Mood Booster Harian”. Di **Layar Awal**, pengguna memilih hari. Di **Layar Motivasi**, muncul kutipan positif dan musik latar yang lembut. Jika ingin mendengar motivasi lain, pindah ke layar lain dengan kata-kata baru.

Tantangan Seru: Bisakah kalian membuat suara yang berubah setiap hari dan tetap membuat pengguna semangat?

Kesimpulan Menarik:
Setiap kasus ini menantang kalian untuk berpikir layaknya **desainer aplikasi profesional**: mengatur alur layar dengan cerdas, dan memilih suara yang tepat untuk menciptakan suasana. Dengan memahami blok “open another screen” dan “close screen”, serta cara mengunggah media, kalian akan bisa membuat aplikasi yang *bukan hanya berfungsi*, tapi juga **hidup dan menyenangkan**!

Siap menaklukkan tantangan digital ini?

Contoh Soal dan Pembahasan

Soal 1: Jelajahi Dunia Multi-Layar!

Soal:
Bayangkan kamu sedang membuat aplikasi kuis dengan 3 layar:

- Layar Menu Utama
- Layar Soal
- Layar Skor

Blok navigasi mana yang akan kamu gunakan untuk:

1. Berpindah dari Layar Menu ke Layar Soal?
2. Kembali ke Menu setelah melihat Skor?

Pembahasan:

1. Untuk berpindah ke layar soal → gunakan blok `open another screen screenName`
   (misalnya: `open another screen "LayarSoal"`).
2. Untuk kembali ke menu → gunakan blok `close screen`, yang akan menutup layar skor dan kembali ke layar sebelumnya.

Dengan memahami ini, kalian bisa membuat aplikasi seperti game atau kuis yang memiliki banyak layar seperti aplikasi profesional!

Soal 2: Siapa Aku? Komponen Misterius

Soal:
Aku sering digunakan untuk memainkan suara klik, tepuk tangan, atau suara lonceng. Aku ringan, cepat, dan cocok untuk efek suara singkat.
Siapakah aku? Dan apa perbedaan utama dibanding saudaraku yang bisa memainkan musik latar?

Pembahasan:
Jawabannya adalah **komponen Sound**.
Perbedaan utama:

- **Sound**: untuk suara pendek seperti efek klik
- **Player**: untuk suara panjang seperti musik latar

Mengenali perbedaan ini akan membantumu memilih komponen yang tepat agar aplikasi tidak lemot dan lebih menyenangkan!

Soal 3: Misi Suara Rahasia

Soal:
Kamu ingin menambahkan suara latar saat aplikasi dimulai, dan juga efek klik saat tombol ditekan.
Komponen apa yang kamu gunakan untuk masing-masing suara tersebut?

Pembahasan:

- Untuk **musik latar**: gunakan **komponen Player**, karena musik latar biasanya berdurasi panjang.
- Untuk **efek klik**: gunakan **komponen Sound**, karena hanya membutuhkan suara singkat.

Dengan strategi ini, kamu bisa membuat aplikasi terasa hidup seperti game sungguhan!

Soal 4: Uji Logika App Inventor!

Soal:
Seorang siswa membuat aplikasi dengan dua layar: "LayarMenu" dan "LayarGame". Ia menggunakan blok berikut:

```
open another screen "LayarGame"
```

Kemudian, saat permainan selesai, ia ingin kembali ke "LayarMenu".
Namun, ia malah menulis:

```
open another screen "LayarMenu"
```

Apa yang akan terjadi? Apakah cara ini tepat?

Pembahasan:
Cara tersebut **kurang tepat**.
Dengan membuka "LayarMenu" lagi menggunakan `open another screen`, maka "LayarMenu" yang lama tidak ditutup. Lama-lama bisa membuat aplikasi berat karena terlalu banyak layar terbuka.

Solusi yang tepat: Gunakan `close screen` agar kembali ke layar sebelumnya **dengan menutup layar aktif**, bukan membuka layar baru.

Soal 5: Tantangan Media

Soal:
Kamu sudah membuat aplikasi musik sederhana di App Inventor. Namun saat dicoba, musik tidak muncul.
Setelah dicek, ternyata komponen Player-nya sudah ada, tapi suaranya tidak terdengar. Apa kemungkinan penyebabnya?

Pembahasan:
Salah satu penyebab umum: **file media belum diunggah** atau **Properti Source belum diatur**.
Solusinya:

1. Masuk ke **Designer**
2. Klik **Media** dan **unggah file musik dari komputer**
3. Atur **Source** pada komponen Player ke nama file musik tersebut.

Media yang tidak diunggah = aplikasi yang “bisu”! Jadi, pastikan kamu selalu cek bagian ini sebelum menjalankan aplikasi.

Jika kamu berhasil memahami semua soal ini, kamu sudah siap membuat aplikasi multi-layar yang penuh suara dan interaksi! Siap membuat karya aplikasi sendiri?

Fakta Menarik / Fun Facts

- Hampir semua aplikasi yang kalian gunakan sehari-hari memiliki lebih dari satu layar atau tampilan!
- Suara dan musik sangat penting untuk membuat game dan aplikasi terasa lebih hidup!

Tips Belajar atau Tips Cepat Menghafal

Tips Cepat Hafal:

- "Ingat `open another screen` untuk pindah, `close screen` untuk kembali."
- "Suara pendek pakai `Sound`, suara panjang pakai `Player`."

Mind Mapping Sederhana:



Cerita Ringan:

Bayangkan kamu buat game kuis:

- Jawab “Benar”? Suara “Yeay!” muncul, lanjut ke pertanyaan baru.
- Jawab “Salah”? Suara “Oh tidak!” terdengar, tetap di pertanyaan yang sama.

Aktivitas Siswa

Aktivitas 1: Aplikasi “Fun Facts with Sound”

Deskripsi:

Aplikasi ini menampilkan fakta-fakta unik. Di setiap layar (screen), akan muncul satu fakta unik dan sebuah tombol untuk mendengarkan suara yang membacakan fakta tersebut (suara kalian sendiri).

Langkah-langkah:

Screen1 (Fakta 1):

- Tambahkan Label → tulis fakta unik (contoh: “Gajah tidak bisa melompat!”)
- Tambahkan Button → beri nama “btnDengar”
- Tambahkan Sound → unggah suara kalian yang membaca fakta

Pseudocode blok:

```
when btnDengar.Click
 call Sound1.Play
```

Screen2 (Fakta 2):

- Sama seperti Screen1, tapi ganti isi fakta dan suara.

Aktivitas 2: Mini Quiz

Deskripsi:

Aplikasi ini berisi pertanyaan sederhana. Jika jawaban benar ditekan, akan muncul suara “Benar!” dan pindah ke layar pertanyaan berikutnya. Jika salah, suara “Salah!” akan dimainkan dan tetap di layar yang sama.

Langkah-langkah:

Screen1:

- Tambahkan Label → pertanyaan (contoh: “Ibukota Indonesia?”)
- Tambahkan dua Button → “Jakarta” (benar) dan “Surabaya” (salah)
- Tambahkan dua Sound → rekaman “Benar!” dan “Salah!”

Pseudocode blok:

```
when ButtonJakarta.Click
  call SoundBenar.Play
  open another screen Screen2

when ButtonSurabaya.Click
  call SoundSalah.Play
```

Screen2:
Pertanyaan dan pilihan lain, ulangi proses.

Aktivitas 3: “Tebak Suara”

Deskripsi:

Siswa mendengar sebuah suara dan menebak bunyinya dari pilihan yang ada.

Langkah-langkah:

- Tambahkan Sound → suara hewan atau benda (misal: kucing mengeong)
- Tambahkan Button untuk memutar suara
- Tambahkan 2 Button jawaban (contoh: “Kucing” dan “Anjing”)
- Tambahkan Sound untuk jawaban benar/salah

Pseudocode blok:

```
when btnPlaySound.Click
  call SoundKucing.Play

when btnKucing.Click
  call SoundBenar.Play
  open next screen

when btnAnjing.Click
  call SoundSalah.Play
```

Aktivitas 4: “Drag and Match” (Seret dan Cocokkan)

Deskripsi:

Siswa mencocokkan gambar dengan teks yang sesuai, misalnya gambar buah dengan nama buah.

Langkah-langkah:

- Tambahkan Gambar (contoh: Gambar apel)
- Tambahkan Label kosong
- Tambahkan daftar pilihan nama buah (Drag & Drop)
- Gunakan event `When Dragged` dan `When Dropped`

Pseudocode blok:

```
when NamaApel.Dragged to KotakJawaban
  if NamaApel = GambarApel
    call SoundBenar.Play
  else
    call SoundSalah.Play
```

Rangkuman

Kita bisa membuat **aplikasi yang lebih seru dan interaktif** dengan menggunakan **lebih dari satu layar (screen)** dan menambahkan **suara atau media** lainnya.

Di bab ini, kamu belajar cara membuat aplikasi yang bisa **berpindah layar**, misalnya dari **halaman utama ke halaman cerita**, dengan menggunakan blok *open another screen*. Kalau ingin **kembali ke layar sebelumnya**, kamu bisa pakai blok *close screen*.

Kamu juga sudah mengenal dua komponen penting:

- **Sound**: untuk memainkan **suara pendek**, seperti bunyi *klik* atau efek suara kecil.
- **Player**: untuk **memutar musik atau audio panjang**, seperti lagu atau narasi.

Kalau kamu bisa **menggabungkan suara dan navigasi antar layar**, aplikasi yang kamu buat akan terasa **lebih hidup, menarik, dan menyenangkan** untuk digunakan.

Latihan Soal

A. Benar atau Salah (5 Soal)

1. Aplikasi hanya bisa memiliki satu layar di App Inventor.
   Jawaban: Salah

2. Blok `open another screen screenName` digunakan untuk berpindah ke layar lain.
   Jawaban: Benar

3. Komponen Sound digunakan untuk memainkan lagu berdurasi panjang.
   Jawaban: Salah

4. Kita bisa menambahkan suara ke tombol agar terdengar efek klik saat ditekan.
   Jawaban: Benar

5. Blok close screen akan membuka layar baru.
   Jawaban: Salah

B. Pilihan Ganda (10 Soal)

6. Apa fungsi dari blok `open another screen screenName`?
   A. Menutup aplikasi
   B. Membuka halaman web
   C. Membuka layar baru
   D. Mengubah warna layar
   Jawaban: C

7. Untuk memainkan efek suara singkat seperti bunyi klik, komponen yang digunakan adalah:

A. Image
B. Player
C. Sound
D. Button
Jawaban: C

8. Komponen yang digunakan untuk memainkan lagu latar adalah:
   A. Sound
   B. Label
   C. Clock
   D. Player
   Jawaban: D

9. Di mana kita mengunggah file suara ke dalam proyek?
   A. Blocks Editor
   B. Designer – Media section
   C. Settings
   D. Control Panel
   Jawaban: B

10. Fungsi `close screen` digunakan untuk:
    A. Menyimpan data
    B. Mengubah gambar
    C. Kembali ke layar sebelumnya
    D. Menutup komputer
    Jawaban: C

11. File suara harus diunggah ke bagian ___ sebelum bisa digunakan.
    A. Control
    B. Blocks
    C. Media
    D. Variables
    Jawaban: C

12. Dalam aplikasi "Story Book", tombol pada Screen1 digunakan untuk:
    A. Menampilkan skor
    B. Keluar aplikasi
    C. Berpindah ke Screen2
    D. Menyalakan musik
    Jawaban: C

13. Saat ingin memutar musik latar yang bisa dijeda (pause), kita harus menggunakan:
    A. Sound
    B. Player
    C. Image
    D. Label

Jawaban: B

14. Properti yang harus diatur agar suara bisa diputar adalah:
    A. Text
    B. Height
    C. Source
    D. FontSize
    Jawaban: C

15. Komponen yang digunakan untuk tombol adalah:
    A. Player
    B. Sound
    C. Button
    D. Image
    Jawaban: C

C. Isian Singkat (5 Soal)

16. Blok yang digunakan untuk kembali ke layar sebelumnya adalah `_____`.
    J**awaban: close screen**

17. Untuk berpindah ke Screen2 dari Screen1, kita menggunakan blok `_____`.
    Jawaban: open another screen screenName "Screen2"

18. Untuk menambahkan suara klik tombol, kita menggunakan komponen `_____`.
    Jawaban: Sound

19. Agar komponen Player bisa memainkan musik, kita harus mengatur properti `_____`.
    Jawaban: Source

20. File suara harus diunggah di bagian `_____` pada tampilan Designer.
    Jawaban: Media

D. Soal Explorasi

Soal:

Bayangkan kamu sedang membuat aplikasi edukasi sederhana dengan dua layar: Screen1 (halaman utama) dan Screen2 (halaman materi). Buatlah urutan langkah dan blok apa saja yang akan kamu gunakan agar saat tombol "Mulai Belajar" ditekan di Screen1, aplikasi berpindah ke Screen2 dan memainkan suara sambutan.

Soal:

Kamu ingin membuat aplikasi kuis dengan suara. Jika jawaban benar, aplikasi akan memutar suara “benar” dan pindah ke pertanyaan berikutnya. Jika salah, akan memutar suara “salah” dan tetap di layar yang sama. Tuliskan bagaimana kamu akan menyusun struktur logikanya dan blok apa yang kamu gunakan.

Soal:

Deskripsikan bagaimana kamu akan membuat aplikasi “Cerita Anak” dengan tiga layar (Judul, Cerita 1, Cerita 2), di mana setiap layar memiliki tombol kembali dan suara yang berbeda. Jelaskan komponen yang digunakan di setiap layar dan bagaimana suara dimainkan secara otomatis saat layar dibuka.

Tugas Proyek

Proyek 1: Aplikasi Kuis Interaktif "Si Pintar"

Deskripsi:
Siswa membuat aplikasi kuis yang terdiri dari beberapa layar: halaman pembuka, halaman soal, dan halaman skor akhir.

Langkah-langkah:

1. Buat **3 layar**: `MenuUtama`, `SoalKuis`, dan `SkorAkhir`.
2. Tambahkan **tombol navigasi** untuk berpindah dari menu ke soal.
3. Gunakan komponen **Sound** untuk memberikan efek suara saat siswa menjawab benar atau salah.
4. Hitung skor dan tampilkan di layar akhir menggunakan label.
5. Tambahkan tombol untuk **kembali ke menu utama**.

Tujuan: Melatih logika, pemahaman alur, dan interaksi antar layar.

Proyek 2: Game Sederhana "Tebak Suara"

Deskripsi:
Siswa membuat aplikasi permainan tebak suara, di mana pengguna mendengar suara dan memilih jawaban yang sesuai dari beberapa pilihan.

Langkah-langkah:

1. Gunakan **komponen Player** untuk memainkan suara (misalnya: suara hewan atau alat musik).
2. Buat beberapa layar: `Menu`, `MainGame`, dan `Hasil`.
3. Tampilkan skor berdasarkan jawaban yang benar.
4. Tambahkan **musik latar** menggunakan Player agar suasana lebih hidup.
5. Gunakan blok `open another screen` untuk transisi antar tahap permainan.

Tujuan: Mengasah kepekaan terhadap suara dan kemampuan membuat aplikasi berbasis media.

Proyek 3: Buku Cerita Digital Interaktif

Deskripsi:
Siswa membuat aplikasi cerita bergambar dengan narasi suara, di mana tiap halaman memiliki teks, gambar, dan suara.

Langkah-langkah:

1. Buat **beberapa layar** mewakili halaman-halaman cerita.
2. Tambahkan gambar latar dan teks di tiap layar.
3. Gunakan komponen **Sound** untuk efek suara singkat atau **Player** untuk narasi panjang.
4. Tambahkan tombol "Lanjut" dan "Kembali" untuk berpindah antar layar.
5. Tambahkan suara latar untuk menciptakan suasana cerita.

Tujuan: Melatih kreativitas, desain multimedia, dan pemahaman alur naratif.

Proyek 4: Aplikasi Edukasi "Belajar Bahasa Inggris"

Deskripsi:
Siswa membuat aplikasi pembelajaran bahasa Inggris dasar, seperti mengenal nama buah, hewan, atau warna dengan gambar dan suara.

Langkah-langkah:

1. Setiap layar mewakili kategori (misal: buah, hewan, warna).
2. Tampilkan gambar dan teks dalam bahasa Inggris.
3. Gunakan **komponen Sound** untuk menyebutkan kata dalam bahasa Inggris.
4. Gunakan tombol untuk berpindah antar kategori.
5. Tambahkan kuis sederhana di akhir menggunakan layar baru.

Tujuan: Meningkatkan keterampilan bahasa dan pembuatan aplikasi edukatif.

Proyek 5: Aplikasi Profil Sekolah

Deskripsi:
Siswa membuat aplikasi profil sekolah yang menampilkan informasi seperti visi-misi, guru, ekstrakurikuler, dan galeri foto, dengan narasi suara.

Langkah-langkah:

1. Buat beberapa layar untuk masing-masing informasi.
2. Tambahkan teks dan gambar yang relevan.
3. Gunakan **Player** untuk menambahkan narasi pada tiap layar.
4. Tambahkan tombol navigasi antar halaman.
5. Tambahkan efek suara saat tombol ditekan.

Tujuan: Memperkenalkan siswa pada konsep aplikasi presentasi interaktif.

Proyek 6: Aplikasi Alarm Motivasi

Deskripsi:
Siswa membuat aplikasi alarm sederhana yang memutar suara motivasi saat tombol ditekan.

Langkah-langkah:

1. Buat satu layar utama dengan tombol “Putar Motivasi”.
2. Gunakan **komponen Player** untuk memutar audio motivasi.
3. Tambahkan efek visual seperti animasi sederhana saat audio diputar.
4. Tambahkan tombol "Ganti Suara" untuk memilih suara lainnya (gunakan layar baru).

Tujuan: Belajar mengatur audio, interaksi, dan pemilihan konten dinamis.

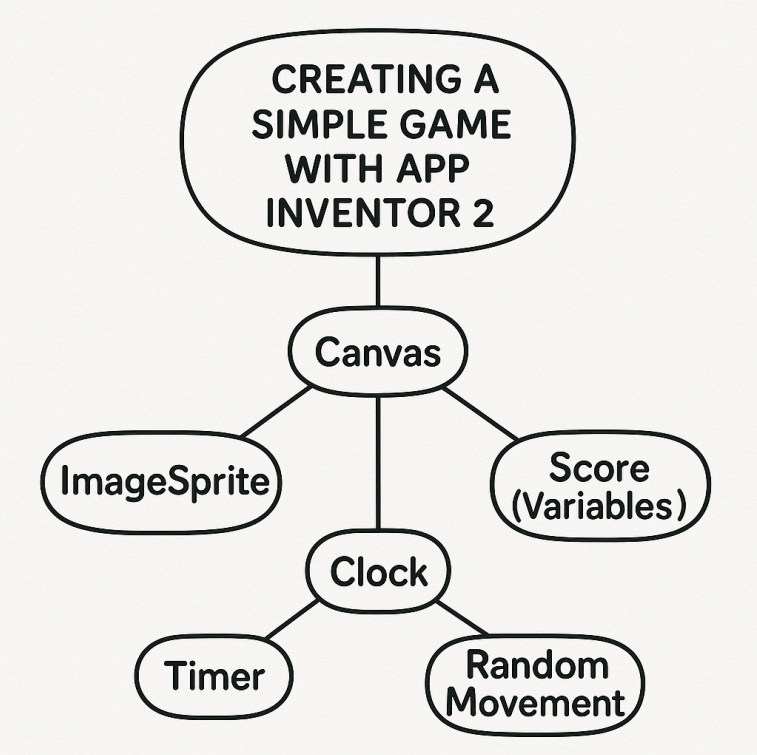
Proyek-proyek ini **mendorong kreativitas, problem solving, dan keterampilan digital siswa SMP** secara menyenangkan dan menantang. Cocok untuk praktek langsung setelah teori! Siap untuk mencoba?

BAB 5: Game Sederhana: "Catch the Dot" atau "Whack-a-Mole"

Tujuan Pembelajaran

Peserta didik mampu membuat game sederhana dengan konsep skor, objek bergerak, dan interaksi sentuhan.

Peta Konsep



Apersepsi

Apa saja elemen penting dalam sebuah game sederhana yang kalian sukai, seperti "Flappy Bird" atau "Candy Crush"? Bagaimana objek-objek di dalamnya bisa bergerak dan berinteraksi?

Penjelasan Konsep (Teori)

Pada bab ini, kamu akan belajar membuat game sederhana seperti “Catch the Dot” atau “Whack-a-Mole”. Meskipun terlihat simpel, game seperti ini punya banyak elemen menarik yang bisa kamu pelajari dan kembangkan. Yuk, kita kenalan dulu dengan komponen-komponen penting dalam game ini!

Apa Saja yang Dibutuhkan untuk Membuat Game Interaktif?

Agar sebuah game bisa berjalan dengan seru, ada beberapa komponen utama yang harus dipahami. Komponen-komponen ini akan membantumu menciptakan pengalaman bermain yang dinamis dan menyenangkan.

1. Canvas Component

Bayangkan kamu sedang menggambar di sebuah papan tulis digital—nah, **Canvas** adalah papan tempat semua objek game digambar dan bergerak.
Fungsi: Tempat utama di mana semua aksi terjadi—objek bisa muncul, bergerak, dan berinteraksi di sini.

2. ImageSprite Component

Objek seperti karakter, musuh, atau target yang bisa kamu sentuh di layar disebut **ImageSprite**.
Fungsi: Bisa bergerak kesana kemari, berubah gambar, dan merespon sentuhan atau tabrakan.

Contoh:

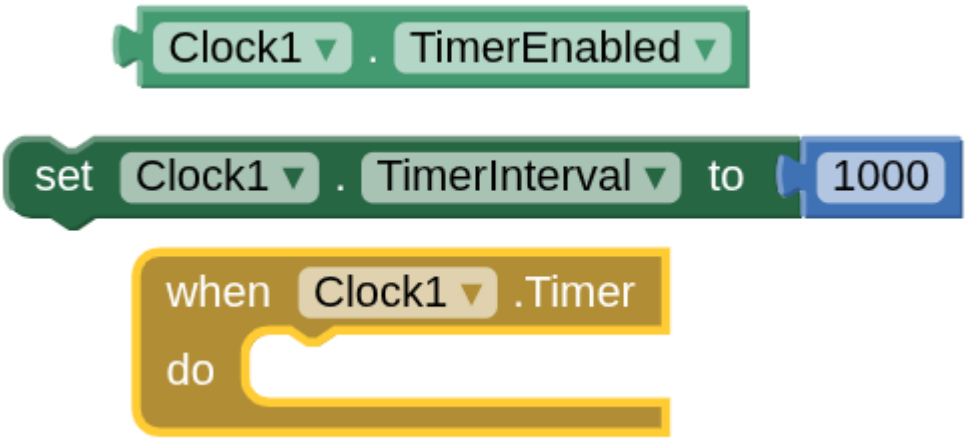
- Titik (dot) yang harus kamu "catch"
- Tikus lucu di “Whack-a-Mole” yang harus dipukul

3. Clock Component

Waktu adalah segalanya dalam game! Nah, komponen **Clock** bertugas sebagai pengatur waktu otomatis.
Fungsi: Membuat peristiwa terjadi secara berulang, seperti memindahkan objek setiap 1 detik.

Dilengkapi dengan:

- **TimerEnabled**: Untuk mengaktifkan atau mematikan fungsi jamnya.
- **TimerInterval**: Untuk mengatur seberapa sering peristiwa terjadi (misalnya setiap 1000 milidetik = 1 detik).
- **Event Clock.Timer**: Perintah yang terjadi setiap kali waktu yang diatur habis. Cocok untuk membuat objek berpindah terus-menerus.



4. ImageSprite.MoveTo(x, y)

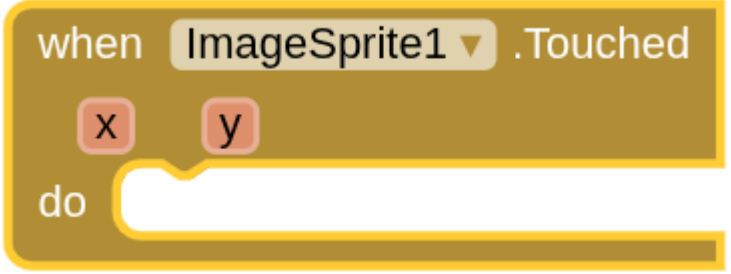
Kalau kamu ingin membuat objek berpindah ke lokasi tertentu, kamu bisa gunakan blok ini.
Fungsi: Memindahkan ImageSprite ke titik koordinat (x, y) yang kamu tentukan.



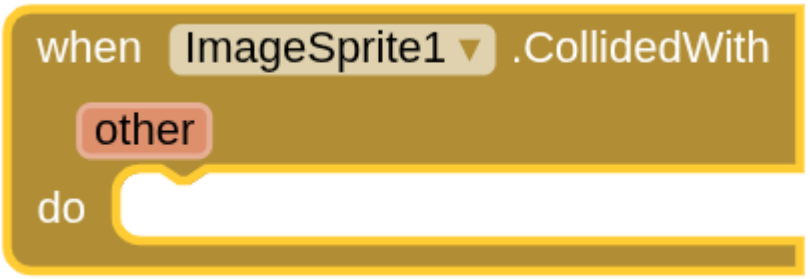
5. Interaksi dalam Game

Game tanpa interaksi seperti menonton film yang tidak bisa di pause. Makanya, interaksi penting banget!

- **ImageSprite.Touched Event**
  Terjadi ketika kamu menyentuh objek di layar. Cocok untuk mencatat skor jika pemain berhasil "menangkap" titik atau memukul target.



- **ImageSprite.CollidedWith Event**
  Terjadi jika dua objek bertabrakan. Misalnya: karakter utama bertabrakan dengan musuh.



6. Variabel untuk Skor

Game yang bagus harus bisa mencatat seberapa hebat pemainnya.
Gunakan: **Variabel global** untuk menyimpan dan menambah skor setiap kali target disentuh.

Contoh:

```
skor = skor + 1
```

7. Random Blocks

Biar gak bosan, objek dalam game bisa muncul di tempat yang berbeda-beda secara acak.
Gunakan: Blok **random integer from… to…** untuk membuat posisi objek selalu berubah-ubah.

Misalnya:

- X dari 0 sampai 300
- Y dari 0 sampai 400



Kesimpulan

Dengan memahami komponen-komponen di atas, kamu bisa mulai membangun game yang punya:

- Objek yang bisa berpindah tempat
- Skor yang bertambah ketika objek disentuh
- Gerakan acak yang membuat permainan jadi lebih menantang!

Siap beraksi? Mari kita lanjut ke tahap pembuatan game-nya dan lihat seberapa seru hasil kreasimu!

Contoh Kasus atau Ilustrasi

Kasus 1: Papan Ajaib di Dunia Digital

Ilustrasi:
Bayangkan kamu sedang masuk ke dunia game, dan kamu melihat sebuah papan kosong berwarna putih seperti papan tulis. Papan ini ajaib! Di sinilah semua karakter game hidup dan bergerak.

Tantangannya:
Kamu diminta membuat dunia itu aktif! Papan kosong itu disebut **Canvas**. Semua aksi seperti objek muncul, bergerak, bahkan meledak, harus terjadi di atasnya.

Konsep yang dipelajari:

- Canvas sebagai tempat utama objek game tampil
- Tempat segala interaksi dan animasi terjadi

Kasus 2: Tikus Bandel yang Suka Muncul Sembunyi

Ilustrasi:
Bayangkan kamu bermain di taman, tiba-tiba tikus-tikus lucu muncul dari lubang dan kamu harus cepat memukulnya sebelum mereka hilang. Nah, itu dia game *Whack-a-Mole*.

Tantangannya:
Tikus-tikus itu disebut **ImageSprite**! Kamu perlu mengatur gambar dan pergerakan mereka agar bisa muncul dan pindah tempat dengan cepat. Dan yang lebih seru—kamu bisa membuat mereka ganti gambar saat disentuh!

Konsep yang dipelajari:

- Komponen ImageSprite
- Gerakan dan interaksi objek

Kasus 3: Jam Ajaib Pengatur Dunia Game

Ilustrasi:
Bayangkan kamu punya jam ajaib. Setiap detik berdetak, sesuatu akan terjadi di dunia game: objek muncul, berpindah tempat, atau kamu kehilangan nyawa!

Tantangannya:
Gunakan **Clock** untuk mengatur kapan objek harus bergerak, seberapa cepat, dan kapan pemain harus bertindak cepat. Jangan biarkan game menjadi diam!

Konsep yang dipelajari:

- Clock untuk membuat peristiwa berulang otomatis
- TimerInterval untuk mengatur ritme game

Kasus 4: Objek Kabur ke Tempat Acak!

Ilustrasi:
Bayangkan kamu sedang mencoba menangkap balon yang terus berpindah ke tempat acak setiap detik. Lokasinya tidak bisa ditebak!

Tantangannya:
Gunakan logika **random** untuk membuat objek tidak muncul di tempat yang sama. Ini membuat pemain harus lebih fokus dan cepat!

Konsep yang dipelajari:

- Fungsi `random integer from…to…` untuk posisi acak
- Kombinasi dengan Clock dan MoveTo untuk perpindahan objek

Kasus 5: Sentuhan Sakti Penambah Skor

Ilustrasi:
Setiap kali kamu berhasil menyentuh target, kamu mendapatkan poin. Tapi hanya jika kamu cepat dan tepat. Kalau tidak, target akan menghilang!

Tantangannya:
Gunakan event **ImageSprite.Touched** untuk mencatat setiap sentuhan dan menambahkan poin ke variabel **skor**. Semakin banyak kamu menyentuh, semakin tinggi skor kamu.

Konsep yang dipelajari:

- Interaksi layar sentuh
- Penggunaan variabel skor
- Menambah skor saat event terjadi

Kasus 6: Tabrakan yang Menegangkan!

Ilustrasi:
Kamu adalah karakter utama yang harus menghindari monster. Jika kamu bertabrakan, kamu kehilangan poin atau bahkan kalah!

Tantangannya:
Buat dua ImageSprite: satu pemain, satu musuh. Jika bertabrakan, gunakan event

ImageSprite.CollidedWith untuk membuat reaksi (seperti mengurangi nyawa atau mengakhiri game).

Konsep yang dipelajari:

- Event tabrakan antar objek
- Respon dalam bentuk efek atau perubahan skor

Setiap kasus ini membantu siswa memahami **konsep pemrograman visual** dengan cara **cerita dan tantangan yang nyata**, namun tetap seru dan penuh tantangan. Apakah kamu siap membangun game pertama buatanmu sendiri? Let's level up!

Contoh Soal dan Pembahasan

Soal:

Canvas itu apa sih, dan kenapa penting dalam pembuatan game?

Pembahasan:
Canvas ibarat panggung utama dalam game. Semua gambar, objek, dan aksi terjadi di atas Canvas. Tanpa Canvas, game seperti film tanpa layar—tidak akan terlihat! Misalnya, kalau kamu membuat game "Catch the Dot", titik yang harus ditangkap akan muncul dan bergerak di dalam Canvas. Jadi, komponen ini sangat penting agar semua elemen bisa muncul dan saling berinteraksi.

Soal:

Kamu ingin membuat seekor tikus dalam game "Whack-a-Mole" muncul di berbagai tempat secara acak. Komponen apa yang harus kamu gunakan dan bagaimana caranya?

Pembahasan:
Untuk membuat tikus muncul di posisi acak, kamu perlu menggunakan:

- **ImageSprite.MoveTo(x, y)**: untuk memindahkan objek ke koordinat tertentu.
- **Blok Random Integer from...to...**: untuk menghasilkan nilai acak pada sumbu X dan Y.

Contoh:

```
X = random integer from 0 to 300
Y = random integer from 0 to 400
```

Kemudian, gunakan blok:

```
ImageSprite.MoveTo(X, Y)
```

Dengan ini, setiap beberapa detik (misalnya 1 detik menggunakan Clock), tikus akan muncul di tempat berbeda-beda. Seru, kan?

Soal:

Dalam sebuah game, titik biru harus disentuh agar pemain mendapat skor. Apa komponen dan logika yang kamu butuhkan agar ini bisa terjadi?

Pembahasan:
Kamu memerlukan:

- **ImageSprite.Touched Event**: event ini aktif saat pemain menyentuh objek.
- **Variabel skor**: untuk menyimpan nilai skor.
- **Logika penambahan skor**:

  skor = skor + 1

Jadi, setiap kali pemain menyentuh titik biru, event `ImageSprite.Touched` dijalankan, dan variabel `skor` akan bertambah 1. Skor bisa ditampilkan di Label agar pemain tahu seberapa jago mereka!

Soal:

Bayangkan kamu membuat game di mana seekor musuh harus bergerak otomatis setiap detik. Komponen apa yang kamu gunakan, dan bagaimana cara kerjanya?

Pembahasan:
Komponen yang digunakan adalah:

- **Clock**

Langkah-langkahnya:

1. Aktifkan `TimerEnabled` = true.
2. Atur `TimerInterval` = 1000 (berarti 1 detik).
3. Gunakan `Clock.Timer` event untuk membuat musuh berpindah.

Misalnya:

  Clock.Timer → ImageSprite.MoveTo(X, Y)

Kombinasikan dengan posisi acak untuk membuat gerakan lebih tidak terduga!

Soal:

Mengapa penting menggunakan variabel dalam game, khususnya untuk mencatat skor? Apa yang terjadi jika kita tidak menggunakan variabel?

Pembahasan:
Variabel seperti "kotak penyimpanan" yang menyimpan nilai skor. Dengan menggunakan variabel:

- Kita bisa menambah nilai setiap kali target disentuh.
- Skor bisa ditampilkan dan diperbarui secara real-time.

Jika tidak menggunakan variabel:

- Skor tidak akan tersimpan.
- Pemain tidak tahu berapa poin yang sudah dikumpulkan.
- Game jadi tidak menantang!

Contoh penggunaan:

```
global skor = 0
ImageSprite.Touched → set skor = skor + 1
```

Jika siswa dapat menjawab dan memahami soal-soal ini, berarti mereka siap untuk naik ke level selanjutnya: **membuat game sendiri!**

Fakta Menarik / Fun Facts

- Game "Pong" tahun 1972 adalah salah satu video game pertama yang menggunakan konsep pergerakan objek dan interaksi!
- Game-game modern juga menggunakan konsep `collision detection` untuk menentukan kapan dua objek saling bersentuhan.

Tips Belajar atau Tips Cepat Menghafal

- "Canvas adalah panggungnya, `ImageSprite` adalah aktornya."
- "`Clock` mengatur waktu gerakan, `random` memberi kejutan."

Aktivitas Siswa

Aktivitas 1: Game "Bouncing Ball" (Bola Pantul)

Tujuan:

Bola memantul di dalam area *Canvas* dan berubah arah saat menyentuh sisi.

Langkah-langkah:

Desain:

- Tambahkan komponen **Canvas** (misalnya ukuran 300x300).
- Tambahkan komponen **Ball** ke dalam Canvas.

Logika:

- Atur kecepatan dan arah awal bola.
- Gunakan event `EdgeReached` agar bola memantul.

Pseudocode:

```
Saat Screen1 dimulai:
  Atur arah bola ke acak (misalnya 30 derajat)
  Atur kecepatan bola menjadi 5

Saat bola menyentuh tepi (EdgeReached):
  Bola berubah arah otomatis (menggunakan fungsi Bounce)
```

Aktivitas 2: Game "Target Practice" (Latihan Menembak Target)

Tujuan:

Pemain menyentuh target yang muncul acak di layar untuk mencetak poin, dengan batas waktu.

Langkah-langkah:

Desain:

- Tambahkan Canvas.
- Tambahkan **ImageSprite** (target).
- Tambahkan **Clock** (untuk waktu permainan).
- Tambahkan **Label** (untuk skor dan waktu).

Logika:

- Target muncul di posisi acak.
- Saat disentuh, skor bertambah dan target pindah.
- Timer berjalan dan saat waktu habis, game berakhir.

Pseudocode:

```
Saat Screen1 dimulai:
```

```
Atur skor = 0
Atur waktu = 30 detik
Aktifkan Clock

Setiap 1 detik (Clock.Timer):
  Kurangi waktu 1
  Jika waktu = 0:
    Tampilkan "Game Over"
    Nonaktifkan Clock

Saat target disentuh:
  Tambah skor 1
  Pindahkan target ke posisi acak
```

Tambahan: Efek Suara

Langkah-langkah:

1. Tambahkan **komponen Sound** (misalnya Sound1 dan Sound2).
2. Upload suara "score" dan "game over" ke media.

Pseudocode:

```
Saat target disentuh:
  Putar suara Sound1

Saat waktu = 0:
  Putar suara Sound2
```

Tambahan: Ubah Desain Game

Langkah-langkah:

1. Ubah warna latar Canvas.
2. Ubah gambar target atau bola menjadi lebih lucu atau menarik (misalnya gambar buah, monster, dll).

Aktivitas 3: Game "Avoid the Enemy" (Hindari Musuh)

Tujuan:

Pemain menggerakkan karakter untuk menghindari musuh yang bergerak acak.

Fitur:

- Tambahkan 2 ImageSprite: satu pemain, satu musuh.
- Gunakan tombol arah atau sentuhan layar untuk menggerakkan pemain.
- Jika bersentuhan dengan musuh, game over.

Pseudocode:

Jika pemain menyentuh musuh:
  Tampilkan "Game Over"

Aktivitas 4: Game "Catch the Falling Object" (Tangkap Benda Jatuh)

Tujuan:

Pemain menggerakkan objek untuk menangkap benda jatuh dari atas layar.

Fitur:

- Gunakan komponen Clock untuk membuat benda jatuh.
- Gunakan sentuhan atau tombol untuk menggerakkan "keranjang".
- Tambah skor setiap kali berhasil menangkap.

Pseudocode:

Setiap 0.5 detik (Clock.Timer):
  Benda turun sedikit

Jika benda menyentuh keranjang:
  Tambah skor
  Kembalikan benda ke atas

Rangkuman

Dengan menggunakan **App Inventor 2**, kita dapat membuat **game sederhana** yang seru seperti *“Catch the Dot”* atau *“Whack-a-Mole”* hanya dengan beberapa **komponen inti**, yaitu **Canvas**, **ImageSprite**, dan **Clock**. Komponen-komponen ini memungkinkan kita mengatur **gerakan objek**, **interaksi sentuhan**, serta elemen **waktu** agar permainan terasa **dinamis dan interaktif**.

Tak hanya itu, kita juga bisa menambahkan berbagai fitur seperti **skor**, **batas waktu permainan**, dan **efek suara** untuk menjadikan game lebih **menarik dan hidup**. Dengan sedikit *eksperimen* dan banyak **kreativitas**, siswa dapat merancang dan mengembangkan **game buatan sendiri** yang **menantang sekaligus menyenangkan** untuk dimainkan.

Latihan Soal

A. Benar atau Salah (5 Soal)

1. Komponen Canvas digunakan untuk menampilkan teks dalam game.
   Jawaban: Salah

2. ImageSprite dapat dipindahkan ke posisi acak menggunakan blok `MoveTo`.
   Jawaban: Benar

3. Clock digunakan untuk menghitung jumlah skor dalam game.
   Jawaban: Salah

4. Event `ImageSprite.Touched` terjadi saat pemain menyentuh objek tersebut.
   Jawaban: Benar

5. Blok `random integer from 1 to 10` menghasilkan angka acak antara 1 sampai 10.
   Jawaban: Benar

B. Pilihan Ganda (10 Soal)

6. Apa fungsi utama dari komponen Canvas?
   A. Menghitung skor
   B. Menampilkan gambar diam
   C. Menjadi tempat gambar dan gerakan ImageSprite
   D. Menyimpan data game
   Jawaban: C

7. Untuk membuat ImageSprite berpindah tempat setiap detik, kita harus menggunakan:
   A. Button
   B. Label
   C. Clock
   D. Notifier
   Jawaban: C

8. Kapan event `Clock.Timer` terjadi?
   A. Saat game dimulai
   B. Saat pemain menyentuh layar
   C. Setiap interval waktu tertentu
   D. Saat Canvas disentuh
   Jawaban: C

9. Apa yang terjadi saat `ImageSprite1.CollidedWith` aktif?
   A. Sprite bergerak lebih cepat
   B. Sprite berubah warna
   C. Dua objek bertabrakan
   D. Skor langsung 0
   Jawaban: C

10. Blok apa yang digunakan untuk membuat objek berpindah ke posisi acak?
    A. set Label.Text
    B. call MoveTo (x, y)

C. call ShowAlert
D. set Score = 0
Jawaban: B

11. Fungsi dari variabel skor dalam game adalah:
    A. Menyimpan waktu permainan
    B. Menentukan ukuran sprite
    C. Menyimpan jumlah poin pemain
    D. Mengatur warna sprite
    Jawaban: C

12. Jika kita ingin sprite muncul dan menghilang, properti apa yang harus diubah?
    A. TimerInterval
    B. Visible
    C. Height
    D. Clock.Enabled
    Jawaban: B

13. Komponen yang digunakan untuk mengatur seberapa sering sprite berpindah posisi adalah:
    A. Label
    B. Canvas
    C. Clock
    D. Button
    Jawaban: C

14. Di mana kita menempatkan blok `set global score to get global score + 1`?
    A. Ketika layar dibuka
    B. Dalam event ImageSprite.Touched
    C. Dalam Clock.Timer
    D. Dalam Button.Click
    Jawaban: B

15. Apa kegunaan dari `TimerEnabled = true`?
    A. Menghentikan sprite
    B. Mengaktifkan timer agar berjalan
    C. Menyimpan data
    D. Menambahkan skor
    Jawaban: B

C. Isian Singkat (5 Soal)

16. Komponen yang digunakan untuk menggambar dan menempatkan objek bergerak disebut ____.
    Jawaban: Canvas

17. Untuk memindahkan objek ke posisi tertentu, kita menggunakan blok ____.
    Jawaban: MoveTo

18. Ketika dua objek bertabrakan, event yang dipicu adalah ____.
    Jawaban: CollidedWith

19. Jika ingin membuat sprite muncul secara acak, kita gunakan blok ____.
    Jawaban: random integer

20. Untuk menampilkan skor, kita gunakan komponen ____.
    Jawaban: Label

D. Soal Explorasi

Soal:

Bayangkan kamu sedang membuat game "Catch the Dot". Dot (titik) muncul secara acak di layar, dan setiap kali disentuh, pemain mendapatkan poin. Jelaskan langkah-langkah yang harus kamu lakukan di App Inventor 2 agar dot bisa muncul di tempat acak setiap 1 detik.

Soal:

Kamu diminta membuat game “Whack-a-Mole”. Ada tiga objek ImageSprite (Mole1, Mole2, Mole3). Hanya satu mole yang muncul setiap saat, dan posisinya selalu berubah. Jelaskan bagaimana kamu menggunakan Clock dan blok acak (random) untuk mengatur agar hanya satu mole muncul secara acak, sementara dua lainnya tersembunyi.

Soal:

Dalam game “Target Practice”, kamu ingin menambahkan fitur waktu. Game hanya berlangsung selama 30 detik. Setelah itu, muncul tulisan "Game Over" dan pemain tidak bisa menambah skor lagi. Jelaskan bagaimana kamu mengatur waktu menggunakan Clock dan kondisi tertentu agar game bisa berakhir dengan baik.

Tugas Proyek

Proyek 1: Tangkap Titik Ajaib! (Catch the Magic Dot)

Deskripsi Singkat:
Kamu akan membuat game di mana pemain harus menangkap titik yang terus berpindah tempat secara acak. Setiap kali titik disentuh, skor akan bertambah.

Langkah Saran:

1. Siapkan *Canvas* dan satu *ImageSprite* berbentuk titik.
2. Gunakan *Clock* untuk memindahkan titik ke posisi acak setiap 1 detik.
3. Tambahkan *event* `ImageSprite.Touched` untuk menambah skor saat titik disentuh.

4. Tampilkan skor di layar menggunakan *Label*.
5. Tambahkan suara saat titik disentuh agar lebih seru!

Tantangan Tambahan: Buat titik bergerak lebih cepat jika skor sudah melewati angka tertentu.

Proyek 2: Pukul Tikus Lucu! (Whack-a-Mole Remix)

Deskripsi Singkat:
Pemain harus memukul tikus yang muncul di berbagai posisi sebelum ia menghilang.

Langkah Saran:

1. Tambahkan beberapa *ImageSprite* berbentuk tikus di *Canvas*.
2. Atur agar satu tikus muncul secara acak menggunakan *Clock* dan *random integer*.
3. Aktifkan *event* `ImageSprite.Touched` untuk menambah skor saat tikus dipukul.
4. Buat waktu kemunculan semakin singkat jika pemain makin jago.
5. Tambahkan efek suara “plak!” saat tikus terkena pukulan.

Tantangan Tambahan: Batasi waktu permainan menjadi 30 detik dan tampilkan skor akhir.

Proyek 3: Hindari Bom, Tangkap Permata!

Deskripsi Singkat:
Pemain harus menyentuh permata untuk mendapatkan skor, tetapi harus menghindari bom agar tidak kehilangan poin.

Langkah Saran:

1. Tambahkan dua jenis *ImageSprite* (permata dan bom) di *Canvas*.
2. Gunakan *Clock* untuk memindahkan posisi keduanya secara acak.
3. Buat `ImageSprite.Touched` untuk permata menambah skor, dan untuk bom mengurangi skor.
4. Tampilkan skor secara real-time.
5. Tambahkan efek visual saat pemain menyentuh bom (misalnya perubahan warna layar sebentar).

Tantangan Tambahan: Buat jumlah bom bertambah seiring meningkatnya skor.

Proyek 4: Balapan Menangkap Harta Karun

Deskripsi Singkat:
Pemain berperan sebagai karakter utama yang harus berlomba menangkap harta karun sebelum tenggelam (menghilang).

Langkah Saran:

1. Tambahkan karakter pemain dan harta karun sebagai *ImageSprite*.
2. Gunakan panah (button) untuk menggerakkan karakter ke posisi tertentu.
3. Jika karakter bertabrakan dengan harta karun (`CollidedWith`), skor bertambah dan harta karun pindah tempat.
4. Gunakan *Clock* untuk mengatur waktu harta karun muncul.
5. Tambahkan batas waktu untuk menambah ketegangan permainan.

Tantangan Tambahan: Tambahkan rintangan yang bergerak yang harus dihindari pemain.

Proyek 5: Game Tepat Waktu – Tangkap dalam 3 Detik!

Deskripsi Singkat:
Pemain harus menangkap titik yang hanya muncul selama 3 detik. Jika terlambat, titik menghilang!

Langkah Saran:

1. Gunakan satu *ImageSprite* yang muncul di posisi acak.
2. Setel *Clock* untuk membuat titik hilang jika tidak disentuh dalam 3 detik.
3. Tambahkan skor jika pemain berhasil menyentuh sebelum waktu habis.
4. Tampilkan pesan “Terlambat!” jika gagal menyentuh titik tepat waktu.

Tantangan Tambahan: Buat level bertambah secara otomatis jika pemain berhasil menangkap 5 titik.

Proyek 6: Mini Turnamen – Siapa Skornya Tertinggi?

Deskripsi Singkat:
Buat game sederhana dari proyek sebelumnya, lalu ajak teman-teman bermain dan catat skor tertinggi!

Langkah Saran:

1. Gunakan salah satu proyek sebelumnya (misalnya Catch the Dot).
2. Tambahkan sistem pencatatan skor tertinggi (*high score*).
3. Setiap pemain bermain selama 30 detik.
4. Buat tabel skor untuk membandingkan hasil pemain satu per satu.
5. Tampilkan pemenangnya di akhir permainan.

Tantangan Tambahan: Tambahkan pilihan nama pemain dan simpan skor mereka.

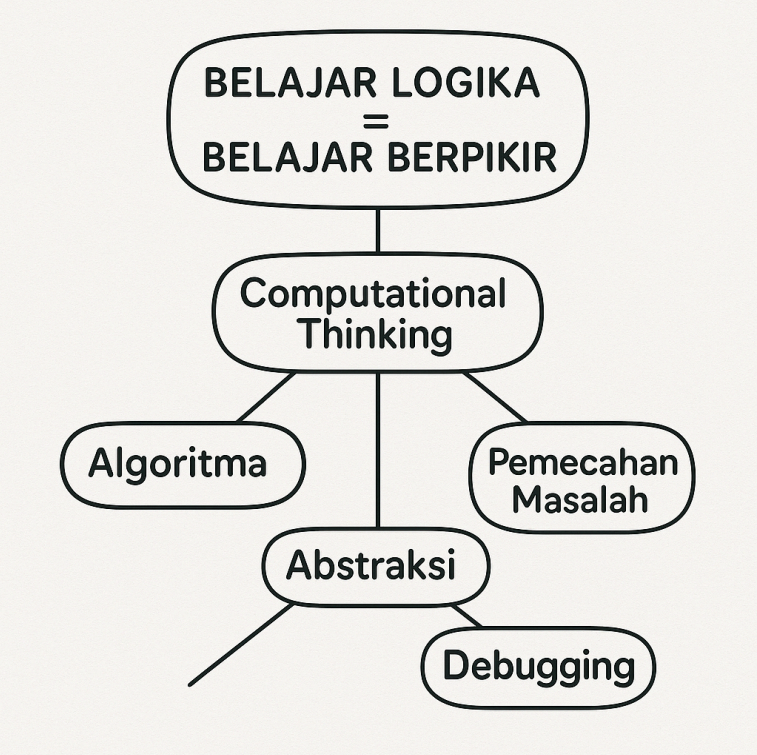
Setiap proyek ini dirancang agar siswa bisa belajar sambil berkreasi dan berpikir logis. Proyek-proyek ini juga bisa dikembangkan lebih lanjut menjadi game yang lebih kompleks seiring kemampuan yang meningkat. Selamat berkarya!

BAB 6: Refleksi: Belajar Logika = Belajar Berpikir

Tujuan Pembelajaran

Peserta didik mampu merefleksikan pentingnya belajar logika dan berpikir komputasional dalam kehidupan sehari-hari, serta memahami bagaimana pemrograman visual App Inventor 2 mendukungnya.

Peta Konsep



Apersepsi

Apakah kalian merasa lebih mudah menyelesaikan masalah atau merencanakan sesuatu setelah belajar membuat aplikasi? Apa hubungannya dengan pelajaran ini?

Penjelasan Konsep (Teori)

Belajar Logika = Belajar Berpikir Cerdas!

Pernahkah kalian merasa lebih mudah membuat keputusan, merancang rencana, atau menyelesaikan masalah setelah belajar membuat aplikasi dengan App Inventor 2? Jika iya, itu bukan kebetulan! Itu karena kalian sedang melatih *cara berpikir komputasional*—sebuah keterampilan berpikir yang sangat berguna dalam kehidupan sehari-hari.

Mari kita bahas bagaimana membuat aplikasi ternyata bisa mengasah logika dan membantu kita menjadi pemecah masalah yang hebat!

Hubungan App Inventor 2 dengan *Computational Thinking*

App Inventor 2 bukan hanya alat untuk membuat aplikasi, tetapi juga sarana untuk melatih cara berpikir logis, sistematis, dan kreatif. Dalam proses membuat aplikasi, kita secara tidak sadar mempraktikkan empat pilar penting dari *Computational Thinking*:

1. **Dekomposisi** – *Memecah masalah menjadi bagian-bagian kecil*
   Seperti menyusun puzzle, membuat aplikasi dimulai dengan membagi tugas besar menjadi langkah-langkah kecil:
   - **Designer**: Mengatur tampilan aplikasi.
   - **Blocks**: Mengatur logika dan fungsi kerja.
   - **Komponen**: Mengatur bagian-bagian penting seperti tombol, teks, dan gambar.

   Dengan dekomposisi, kita belajar bahwa masalah besar akan terasa lebih mudah jika kita selesaikan sedikit demi sedikit.

2. **Pengenalan Pola (Pattern Recognition)** – *Menemukan kemiripan dan pengulangan*
   Saat membuat aplikasi, kita sering menggunakan blok kode yang mirip, seperti:
   - when Button.Click → Saat tombol diklik.
   - set Label.Text to → Mengubah teks.

   Melihat pola ini membantu kita mempercepat proses pembuatan aplikasi dan menghindari pengulangan yang tidak perlu.

3. **Abstraksi** – *Fokus pada hal penting, abaikan detail yang rumit*
   Kita tidak perlu tahu bagaimana komputer membaca blok kode kita. Yang penting:
   - Kita tahu apa fungsi tiap blok.
   - Kita tahu apa hasil akhirnya.

   Abstraksi membantu kita fokus pada inti masalah, tanpa tenggelam dalam detail teknis yang membingungkan.

4. **Algoritma** – *Langkah-langkah logis untuk menyelesaikan tugas*
   Sama seperti mengikuti resep masakan, kita menyusun blok-blok App Inventor agar aplikasi bisa "berjalan" sesuai keinginan.
   Contohnya:
   - Jika tombol diklik → Tampilkan teks → Putar suara.

   Ini adalah bentuk nyata dari berpikir logis dan sistematis!

Pentingnya Debugging

Saat membuat aplikasi, seringkali kita menemukan error (kesalahan). Tenang, itu wajar dan bagian dari proses belajar!
Debugging adalah proses menemukan dan memperbaiki kesalahan pada aplikasi.

- Kesalahan bukan musuh, tapi sahabat yang mengajari kita menjadi lebih teliti.

- Semakin sering kita menemukan dan memperbaiki kesalahan, semakin kuat logika kita!

Aplikasi dalam Kehidupan Sehari-hari

Siapa bilang logika dan pemrograman hanya untuk komputer? Tanpa kita sadari, kita menggunakan *Computational Thinking* dalam banyak aktivitas sehari-hari, seperti:

- **Membuat jadwal harian**
  → Ini seperti menyusun *algoritma* sederhana!

- **Mencari solusi saat menghadapi masalah**
  → Misalnya, bagaimana menghadapi teman yang kesal? Kita *menganalisis situasi*, mencari solusi, dan mencoba langkah terbaik!

- **Mengambil keputusan berdasarkan kondisi**
  → Contoh: "Kalau hujan, saya bawa payung. Kalau tidak, cukup topi."
  Ini adalah contoh nyata dari *logika if-then-else*!

- **Mengikuti instruksi secara berurutan**
  → Saat membuat kue atau bermain game, kita mengikuti langkah demi langkah yang harus dilakukan—*seperti menjalankan program!*

Tantangan untuk Kalian

Sekarang bayangkan:
Bagaimana kalau kalian bisa menggunakan cara berpikir ini untuk:

- Menyusun strategi belajar yang lebih efisien?
- Membantu teman memecahkan masalah?
- Membuat ide aplikasi yang bisa membantu orang lain?

Semua dimulai dari sini—dari belajar logika lewat App Inventor dan berpikir komputasional.

Kesimpulan

Belajar logika dan berpikir komputasional bukan hanya soal teknologi. Ini tentang bagaimana kita bisa menjadi **pemikir yang tangguh**, **pemecah masalah yang kreatif**, dan **pengambil keputusan yang bijak**—baik di dunia digital maupun dunia nyata.

Yuk, terus asah logika kita dan jadi pembuat solusi, bukan penonton masalah!

Contoh Kasus atau Ilustrasi

Kasus 1: "Tersesat di Gedung Sekolah Baru!"

Ilustrasi:
Bayangkan kalian baru pindah ke sekolah baru yang sangat besar. Kalian harus mencari ruang kelas IPS, tapi gedungnya punya banyak lorong dan lantai.

Tantangan Logika:
Kalian perlu **memecah masalah** (dekomposisi): lantai berapa? sayap kiri atau kanan?
Kemudian, gunakan **algoritma** sederhana: "Kalau di lantai 1 tidak ada papan nama kelas IPS, naik ke lantai 2 → Cek setiap lorong dari kiri ke kanan."

Refleksi:
Ternyata, berpikir seperti ini sama seperti menyusun langkah di App Inventor. Kalian sedang mempraktekkan Computational Thinking di dunia nyata!

Kasus 2: "Drama Jadwal Ekskul dan PR"

Ilustrasi:
Kalian ingin ikut futsal, pramuka, dan coding club. Tapi semuanya punya jadwal yang tumpang tindih. Di sisi lain, PR menumpuk!

Tantangan Logika:
Gunakan **abstraksi**: fokus pada kegiatan yang paling penting dulu.
Gunakan **if-then-else**: "Kalau hari Senin ada coding dan futsal → pilih yang lebih prioritas minggu ini → ganti minggu depan."

Refleksi:
Kalian sedang membuat algoritma untuk hidup kalian sendiri—menyusun strategi seperti menyusun blok kode!

Kasus 3: "Teman Kesal, Apa Solusinya?"

Ilustrasi:
Seorang teman tampak kesal dan tidak mau bicara. Kalian ingin membantu, tapi tidak tahu apa yang salah.

Tantangan Logika:
Gunakan **pengenalan pola**: Apakah dia seperti ini setiap kali nilai ujiannya jelek? Atau kalau sedang bertengkar dengan orang lain?
Lalu buat **strategi pendekatan**: "Kalau sedang kesal → beri waktu dulu → lalu ajak ngobrol santai."

Refleksi:
Ternyata berpikir logis bukan hanya soal komputer, tapi juga soal memahami perasaan orang lain dengan langkah-langkah yang masuk akal.

Kasus 4: "Misi Rahasia: Menyusup ke Dapur Tanpa Ketahuan!"

Ilustrasi:
Kalian ingin mengambil cemilan tengah malam tanpa ketahuan orang tua. Kalian harus menyusun rencana yang sistematis!

Tantangan Logika:
Gunakan **algoritma rahasia**:

1. Matikan suara langkah (pakai kaus kaki).
2. Periksa kondisi: "Kalau lampu dapur menyala → tunggu 5 menit → coba lagi."
3. Uji rencana dan siap debug kalau alarm rumah bunyi!

Refleksi:
Ini cara berpikir ala pemrogram: susun langkah → jalankan → uji → perbaiki kalau gagal. Seru kan?

Kasus 5: "Game Seru: Membuat Permainan Mini Sendiri"

Ilustrasi:
Kalian ingin membuat permainan kuis sederhana tentang pelajaran di App Inventor 2. Tapi bingung harus mulai dari mana.

Tantangan Logika:
Gunakan **dekomposisi**: desain tampilan dulu → lalu buat soal → kemudian logika penilaian.
Gunakan **pola** untuk setiap pertanyaan dan jawaban.
Gunakan **debugging** saat nilai tidak sesuai harapan.

Refleksi:
Dari game kecil ini, kalian belajar bagaimana ide bisa diubah jadi kenyataan dengan logika dan kreativitas.

Kasus 6: "Tantangan Liburan: Rute Tercepat ke 3 Tempat Wisata"

Ilustrasi:
Kalian dan keluarga ingin pergi ke kebun binatang, museum, dan tempat makan dalam satu hari. Tapi bingung urutan mana dulu biar hemat waktu dan bensin.

Tantangan Logika:
Gunakan **abstraksi**: abaikan hal-hal kecil dulu, fokus pada lokasi dan jarak.
Gunakan **algoritma rute**: coba urutan yang memungkinkan → ukur waktu → evaluasi hasil.

Refleksi:
Kalian sebenarnya sedang melakukan *routing logic*, sama seperti logika yang digunakan oleh Google Maps dan... ya, pemrograman!

Semua kasus ini **bukan hanya cerita seru**, tapi contoh nyata bagaimana **belajar logika dan berpikir komputasional** bisa membuat kita **lebih pintar, lebih kreatif, dan lebih siap menghadapi dunia**—baik di layar komputer maupun di kehidupan sehari-hari. Siap jadi pemecah masalah sejati?

Contoh Soal dan Pembahasan

Soal 1: Puzzle Rencana Liburan

Bayangkan kamu ingin merencanakan liburan keluarga ke pantai. Tapi kamu hanya punya waktu 1 hari. Kamu harus menyusun rencana:

1. Bangun pagi
2. Siapkan bekal
3. Berangkat
4. Main di pantai
5. Pulang

Pertanyaannya:
Urutkan aktivitas di atas seperti algoritma sederhana. Jika kamu menukar urutannya, apa akibatnya?

Pembahasan:
Ini contoh dari **algoritma**, yaitu langkah-langkah logis yang harus dilakukan secara berurutan.
Jika "Berangkat" dilakukan sebelum "Siapkan bekal", maka kita akan lupa membawa makanan.
Urutan yang benar:
Bangun → Siapkan Bekal → Berangkat → Main di Pantai → Pulang.
Ini melatih siswa menyusun langkah sistematis dan berpikir logis!

Soal 2: Tombol Ajaib

Di App Inventor, kamu membuat tombol yang jika ditekan akan menampilkan tulisan: *"Selamat Datang!"* dan memutar suara musik.

Pertanyaannya:
Tuliskan algoritma logika sederhananya. Lalu, apa yang terjadi jika kamu lupa menyusun salah satu langkah?

Pembahasan:
Algoritma logika:

- Jika tombol ditekan → Tampilkan tulisan
- Jika tombol ditekan → Putar suara

ika kita lupa satu langkah, misalnya tidak memutar suara, maka fitur aplikasi tidak lengkap.
Ini contoh bagaimana logika mempengaruhi fungsi aplikasi dan pentingnya **menyusun langkah dengan lengkap dan benar**.

Soal 3: Detektif Pola

Saat membuat aplikasi, kamu sering melihat blok seperti:

> when Button1.Click → set Label1.Text to "Halo!"
> when Button2.Click → set Label2.Text to "Selamat pagi!"

Pertanyaannya:
Apa pola yang bisa kamu temukan dari dua blok tersebut? Bagaimana pola ini membantumu saat membuat aplikasi?

Pembahasan:
Pola:

- Ketika tombol diklik → Mengubah teks label.

Dengan melihat **pattern recognition** seperti ini, siswa belajar bahwa kode memiliki pola berulang.
Mereka bisa membuat satu template dasar lalu menggandakan atau menyesuaikannya untuk fitur lain, sehingga lebih efisien. Ini melatih **kreativitas dan efisiensi**!

Soal 4: Misteri Abstraksi

Dalam membuat aplikasi, kamu hanya fokus pada:

- Fungsi setiap tombol
- Tampilan yang nyaman
- Hasil akhir yang ingin dicapai

Pertanyaannya:
Mengapa kamu tidak perlu tahu bagaimana komputer memproses semua blok di dalamnya? Apa manfaat dari berpikir seperti ini?

Pembahasan:
Ini adalah contoh **abstraksi**, yaitu memilih fokus pada hal penting dan mengabaikan detail teknis yang rumit.
Dengan abstraksi, siswa diajak berpikir strategis: tidak semua hal harus dipahami secara mendalam, cukup pahami **apa input dan output-nya**. Ini mempermudah pengambilan keputusan!

Soal 5: Misi Debugging!

Kamu membuat aplikasi kuis sederhana, tapi saat tombol "Periksa Jawaban" diklik, tidak muncul hasilnya. Setelah dicek, ternyata kamu lupa menambahkan blok **"set Label.Text to..."**

Pertanyaannya:
Apa yang harus kamu lakukan? Apa pelajaran yang bisa kamu ambil dari kesalahan ini?

Pembahasan:
Kamu harus melakukan proses **debugging**—mengecek kesalahan satu per satu, memperbaiki blok yang hilang.
Ini mengajarkan bahwa **kesalahan bukan kegagalan**, tapi bagian penting dari proses belajar.
Semakin sering kamu memperbaiki, semakin **tajam logika dan teliti pikiranmu!**

Soal 6: Logika dalam Kehidupan Nyata

Kamu hendak pergi ke sekolah. Cuaca terlihat mendung.

Pertanyaannya:
Terapkan logika *if-then-else* dalam situasi ini. Tulis dalam bentuk kalimat dan logika pemrograman.

Pembahasan:
Kalimat:
"Jika hujan, maka saya bawa payung. Jika tidak, cukup pakai jaket."

Logika:

```
if cuaca == "hujan":
    bawa_payung()
else:
    pakai_jaket()
```

Siswa diajak memahami bahwa logika pemrograman **berasal dari cara berpikir sehari-hari**, dan itu sangat relevan!

Semua soal di atas bertujuan agar siswa:

- Merefleksikan konsep logika secara menyenangkan.
- Menghubungkan teori dengan praktik nyata.
- Merasa tertantang dan bangga saat bisa menyelesaikan soal.

Fakta Menarik / Fun Facts

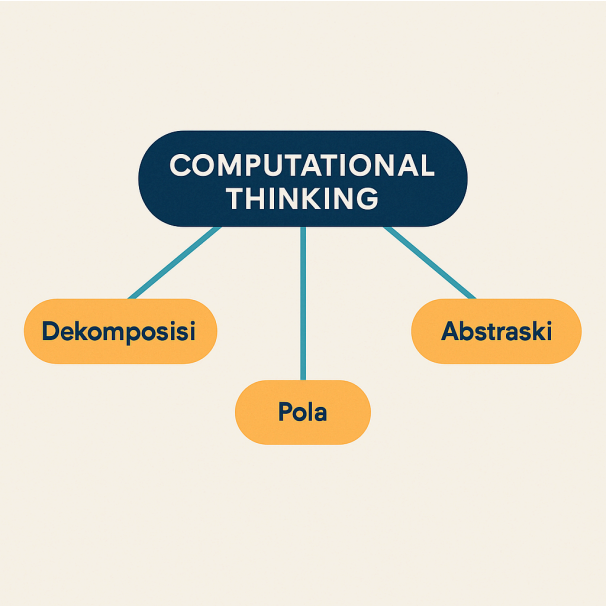
- Para ilmuwan dan insinyur sering menggunakan Computational Thinking untuk memecahkan masalah-masalah besar di dunia nyata!
- Berpikir logis membantu kita menjadi lebih efisien dan terorganisir dalam segala hal yang kita lakukan.

Tips Belajar atau Tips Cepat Menghafal

Tips Cepat Hafal:

- Ingat CT adalah "D.A.P.A." (Dekomposisi, Abstraksi, Pengenalan Pola, Algoritma).
- "Kesalahan itu guru terbaikmu saat ngoding!"

Mind Mapping: Ingat D.A.P.A.



Cerita Ringan:

Bayangkan kamu seorang detektif:

- Kamu memecah kasus jadi bagian kecil **(dekomposisi)**
- Kamu abaikan detail yang tidak penting **(abstraksi)**

- Kamu cari pola dari petunjuk **(pengenalan pola)**
- Lalu susun langkah untuk menemukan pelaku (algoritma)

Aktivitas Siswa

Aktivitas 1: Diskusi Kelompok: Penerapan Computational Thinking

Tujuan: Siswa berdiskusi bagaimana mereka menggunakan *Computational Thinking* seperti **Dekomposisi, Pola, Abstraksi, dan Algoritma** saat belajar atau mengatur kegiatan.

Langkah-langkah:

1. **Bentuk kelompok kecil** (3–5 orang).
2. **Pilih topik**: Mau bahas soal “mempersiapkan ujian” atau “mengatur kegiatan harian di rumah”.
3. **Diskusikan**:
    - Apa saja langkah-langkah yang biasa dilakukan? (gunakan **Dekomposisi**: pecah jadi bagian kecil).
    - Apakah ada pola yang berulang setiap hari? (**Pengenalan Pola**)
    - Bagian mana yang penting dan mana yang bisa disederhanakan? (**Abstraksi**)
    - Buat urutan cara menyelesaikan tugas itu. (**Algoritma**)
4. **Tulis hasil diskusi** di kertas atau papan.
5. **Satu orang dari kelompok** mempresentasikan hasilnya ke kelas (2 menit saja).

Aktivitas 2: Main Detektif Masalah

Tujuan: Melatih siswa berpikir logis saat menyelesaikan masalah sehari-hari.

Langkah-langkah:

1. Guru memberikan **masalah sederhana**, misalnya:
    - "HP tidak bisa nyala"
    - "Laptop lemot"
    - "Air galon habis terus di rumah"
2. Dalam kelompok, **siswa menyelidiki**:
    - Apa saja kemungkinan penyebabnya? (gunakan **logika dan dekomposisi**)
    - Apa saja solusi yang mungkin?
3. Siswa membuat **daftar penyebab dan solusi**.
4. Presentasikan hasil temuan dengan gaya seperti detektif (bisa lucu atau serius).

Aktivitas 3: Drama Mini: Menjelaskan Konsep CT

Tujuan: Membuat siswa memahami dan menjelaskan 1 konsep *Computational Thinking* lewat drama.

Langkah-langkah:

1. Tiap kelompok **diberi 1 konsep**:
    - Contoh: Debugging, Algoritma, Dekomposisi, Pola, atau Abstraksi.
2. Buat **skenario cerita pendek** yang menjelaskan konsep itu.
    - Contoh: Drama tentang orang yang terus gagal buat kue, tapi akhirnya sukses setelah memperbaiki resep (debugging).
3. **Latihan peran** selama 10–15 menit.
4. **Tampilkan dramanya** di depan kelas (maks. 3 menit).
5. Setelah tampil, beri penjelasan singkat: “Kami menjelaskan konsep … dengan cara …”.

Aktivitas 4: Tantangan D.A.P.A.

Tujuan: Siswa mengenal dan menerapkan 4 konsep CT (Dekomposisi, Abstraksi, Pengenalan Pola, Algoritma) dalam kehidupan sehari-hari.

Langkah-langkah:

1. Setiap siswa atau kelompok **membuat contoh nyata dari 4 konsep D.A.P.A.**, misalnya:
    - **Dekomposisi:** Membagi tugas bersih-bersih rumah jadi bagian kecil (nyapu, pel, cuci piring).
    - **Abstraksi:** Fokus pada hal penting saat menyiapkan tas sekolah (buku, alat tulis).
    - **Pola:** Menyadari bahwa setiap pagi rutinitasnya mirip (mandi, sarapan, berangkat).
    - **Algoritma:** Cara memasak mie instan (urutan langkahnya jelas).
2. Tulis contoh tersebut di kertas atau slide sederhana.
3. **Presentasikan di depan kelas** atau **dipajang di dinding kelas**.

Aktivitas 5: Presentasi Kilat (1 Menit)

Tujuan: Melatih siswa berpikir cepat dan menyampaikan ide secara singkat.

Langkah-langkah:

1. Tiap siswa **memilih contoh kecil dalam hidup** yang pakai logika atau algoritma.
    - Contoh: Menentukan pakaian saat cuaca mendung → lihat langit, lihat suhu, pilih jaket.
2. Buat kerangka presentasi:
    - Masalahnya apa?
    - Bagaimana cara kamu mengambil keputusan?
    - Apa urutan langkahnya? (algoritma/logika)
3. Siswa maju satu per satu, **bicara selama maksimal 1 menit**.

4. Teman-teman boleh memberi komentar atau tepuk tangan semangat.

Rangkuman

Belajar membuat aplikasi dengan App Inventor 2 bukan hanya tentang coding, tetapi juga melatih **Computational Thinking** yang terdiri dari dekomposisi, pengenalan pola, abstraksi, dan algoritma. Kemampuan ini sangat penting untuk **memecahkan masalah** dan berpikir secara logis dalam berbagai aspek kehidupan.

Belajar logika dan berpikir komputasional sangat penting bukan hanya untuk membuat aplikasi, tapi juga dalam kehidupan sehari-hari. Dengan **App Inventor 2**, siswa dilatih untuk berpikir seperti seorang pemecah masalah: memecah tugas besar (dekomposisi), mencari pola (pola), menyusun rencana langkah demi langkah (algoritma), dan mengabaikan hal tidak penting (abstraksi).

Lebih dari sekadar coding, Computational Thinking membantu kita saat menyusun jadwal, memecahkan masalah, atau mengambil keputusan. Konsep seperti **debugging** juga membuat kita terbiasa mengevaluasi dan memperbaiki kesalahan. Ini semua membantu kita menjadi pribadi yang lebih logis, terorganisir, dan siap menghadapi tantangan!

Latihan Soal

A. Benar atau Salah (5 Soal)

Tulis “Benar” atau “Salah” untuk setiap pernyataan berikut.

1. App Inventor 2 mengharuskan siswa menghafal semua bahasa pemrograman.
   Jawaban: Salah

2. Dekomposisi membantu kita memecah masalah besar menjadi tugas-tugas kecil.
   Jawaban: Benar

3. Debugging tidak penting karena program biasanya langsung berjalan dengan benar.
   Jawaban: Salah

4. Algoritma adalah langkah-langkah logis yang digunakan untuk menyelesaikan masalah.
   Jawaban: Benar

5. Computational Thinking hanya berguna saat membuat aplikasi saja.
   Jawaban: Salah

B. Pilihan Ganda (10 Soal)

Pilih jawaban yang paling tepat (A, B, C, atau D).

6. Apa tujuan dari dekomposisi dalam berpikir komputasional?
   A. Menghindari kesalahan saat coding
   B. Menyusun urutan logika
   C. Memecah masalah besar menjadi bagian kecil
   D. Menghapus komponen yang tidak perlu
   Jawaban: C

7. Contoh nyata dari penggunaan algoritma di kehidupan sehari-hari adalah:
   A. Bermain game tanpa rencana
   B. Menulis puisi secara acak
   C. Menyiapkan sarapan dengan urutan langkah
   D. Menggambar bebas tanpa aturan
   Jawaban: C

8. Dalam App Inventor 2, blok "when Button.Click do" menunjukkan:
   A. Algoritma
   B. Abstraksi
   C. Pengenalan pola
   D. Debugging
   Jawaban: C

9. Apa yang dimaksud dengan debugging?
   A. Menyalin kode dari internet
   B. Menambah fitur aplikasi
   C. Mencari dan memperbaiki kesalahan
   D. Membuat desain aplikasi
   Jawaban: C

10. Pengenalan pola membantu kita:
    A. Menghindari belajar kode
    B. Mengenali struktur yang sering muncul
    C. Mengganti blok dengan cepat
    D. Menyimpan aplikasi
    Jawaban: B

11. Berikut ini yang merupakan contoh penggunaan logika if-then-else di rumah adalah:
    A. Menonton TV setiap malam
    B. Jika hujan, maka pakai jas hujan
    C. Memasak tanpa resep
    D. Membaca buku setiap waktu
    Jawaban: B

12. Mengapa penting belajar berpikir komputasional?
    A. Agar tidak perlu belajar matematika
    B. Untuk jadi artis
    C. Agar mampu menyusun solusi masalah secara sistematis

D. Supaya bisa main game lebih baik
Jawaban: C

13. Apa yang dimaksud dengan abstraksi dalam App Inventor?
    A. Menambahkan tombol
    B. Fokus pada ide penting dan mengabaikan hal tidak perlu
    C. Menghapus semua kode
    D. Menyembunyikan semua komponen
    Jawaban: B

14. Saat sebuah aplikasi tidak berfungsi, langkah pertama yang perlu dilakukan adalah:
    A. Menutup laptop
    B. Membuat ulang aplikasi
    C. Melakukan debugging
    D. Mengganti komponen
    Jawaban: C

15. Dalam berpikir komputasional, langkah menyusun urutan logis disebut:
    A. Algoritma
    B. Pola
    C. Abstraksi
    D. Blok
    Jawaban: A

C. Isian Singkat (5 Soal)

Isilah bagian kosong dengan jawaban yang tepat.

16. Langkah mencari dan memperbaiki kesalahan dalam program disebut _____.
    Jawaban: debugging

17. Dalam membuat rencana perjalanan, membagi tugas seperti pesan tiket dan cari hotel termasuk _____.
    Jawaban: dekomposisi

18. “Jika suhu lebih dari 38°C, maka minum obat.” adalah contoh _____.
    Jawaban: logika if-then-else

19. Mengenali pola dalam kode seperti “when Button.Click” adalah contoh_____.
    Jawaban: pengenalan pola

20. Menyusun langkah membuat kue dari awal hingga akhir adalah contoh dari _____.
    Jawaban: algoritma

D. Soal Explorasi

Tuliskan jawabanmu dalam bentuk paragraf pendek.

Soal: Contoh Penggunaan Dekomposisi dan Algoritma dalam Piknik Sekolah

Saat merencanakan piknik sekolah, saya menggunakan dekomposisi dengan membagi tugas ke dalam bagian-bagian seperti memilih tempat, menyusun jadwal, mengatur transportasi, dan menyiapkan makanan. Setelah itu, saya menyusun algoritma untuk setiap bagian, misalnya: pertama survei tempat, lalu konfirmasi, dan terakhir pesan bus. Semua langkah ini disusun secara berurutan agar kegiatan berjalan lancar.

Soal: Contoh Debugging dalam Kehidupan Sehari-hari

Waktu saya membuat presentasi, tampilannya tidak muncul saat diputar. Saya mulai debugging dengan memeriksa apakah proyektor tersambung, apakah file presentasi rusak, dan apakah formatnya sesuai. Ternyata file saya disimpan dalam format yang tidak cocok, jadi saya ubah ke format .pptx dan berhasil ditampilkan. Ini menunjukkan pentingnya debugging dalam menyelesaikan masalah.

Soal: Contoh Logika If-Then-Else di Kehidupan Sehari-hari

Suatu pagi, saya bangun dan melihat langit mendung. Saya berpikir, "Jika hujan turun, maka saya akan membawa payung. Jika tidak, saya hanya memakai jaket." Ternyata hujan benar-benar turun, jadi saya membawa payung. Dengan logika if-then-else, saya bisa membuat keputusan yang tepat berdasarkan kondisi yang terjadi.

Tugas Proyek

Proyek 1: Aplikasi “Jadwal Pintar”

Tujuan: Membuat aplikasi pengatur jadwal harian yang membantu siswa mengatur waktu belajar dan istirahat.

Langkah-langkah:

1. **Dekomposisi:** Tentukan fitur utama, seperti: tambah jadwal, lihat jadwal hari ini, dan pengingat waktu.
2. **Desain (Designer):** Buat tampilan sederhana dengan daftar hari, input jadwal, dan tombol simpan.
3. **Logika (Blocks):** Tambahkan logika if-then untuk menampilkan jadwal sesuai hari yang dipilih.
4. **Debugging:** Uji fitur pengingat, pastikan semua hari bisa ditampilkan dengan benar.
5. **Refleksi:** Diskusikan bagaimana menyusun langkah-langkah ini membantu berpikir sistematis.

Proyek 2: Game “Tebak Kata Logika”

Tujuan: Membuat game sederhana yang melatih logika dan pengenalan pola.

Langkah-langkah:

1. **Dekomposisi:** Buat daftar kata rahasia dan input jawaban dari pemain.
2. **Desain:** Buat tampilan pertanyaan, tombol “Cek Jawaban”, dan skor pemain.
3. **Pattern Recognition:** Gunakan pola yang sama untuk tiap soal agar mudah disalin dan dimodifikasi.
4. **Algoritma:** Buat urutan logika: input jawaban → cek benar atau salah → tambah skor atau tampilkan pesan.
5. **Debugging:** Coba berbagai skenario jawaban, pastikan hasil sesuai logika.

Proyek 3: Aplikasi “Cuaca dan Keputusan”

Tujuan: Membuat aplikasi prediksi aktivitas berdasarkan kondisi cuaca (simulasi logika if-then-else).

Langkah-langkah:

1. **Dekomposisi:** Buat 3 kondisi cuaca (hujan, panas, berawan) dan keputusan yang diambil.
2. **Desain:** Sediakan dropdown pilihan cuaca dan hasil saran kegiatan.
3. **Logika:** Gunakan struktur if-then-else untuk menampilkan rekomendasi aktivitas.
4. **Debugging:** Cek semua kemungkinan pilihan cuaca apakah logikanya sesuai.
5. **Diskusi:** Bandingkan dengan keputusan sehari-hari, refleksikan penerapan logika di kehidupan nyata.

Proyek 4: Aplikasi “Cerita Pilihan Ganda”

Tujuan: Membuat cerita interaktif di mana pengguna memilih alur cerita berdasarkan keputusan.

Langkah-langkah:

1. **Dekomposisi:** Tentukan tema cerita dan titik keputusan (cabang alur cerita).
2. **Desain:** Buat tampilan cerita dengan dua tombol pilihan di setiap bagian.
3. **Abstraksi:** Fokus pada struktur cerita, bukan detail setiap kalimat.
4. **Logika:** Buat alur percabangan menggunakan blok if-then dan variabel.
5. **Refleksi:** Bahas bagaimana alur logika mempengaruhi hasil cerita.

Proyek 5: Kalkulator Mini “Keputusan Sehat”

Tujuan: Aplikasi yang memberi saran gaya hidup sehat berdasarkan input sederhana (seperti jam tidur, makanan, atau aktivitas).

Langkah-langkah:

1. **Dekomposisi:** Bagi menjadi kategori: tidur, makan, olahraga.
2. **Desain:** Input pengguna berupa pilihan atau angka (misal: tidur <6 jam, >8 jam).
3. **Logika:** Gunakan kombinasi logika if-then untuk memberi saran berdasarkan input.
4. **Debugging:** Uji semua kombinasi untuk melihat apakah aplikasi merespon dengan tepat.
5. **Diskusi:** Diskusikan bagaimana logika sederhana bisa memberi pengaruh besar pada kebiasaan sehari-hari.

Proyek 6: Aplikasi “Teman Curhat”

Tujuan: Membuat chatbot sederhana yang merespons perasaan pengguna (senang, sedih, marah).

Langkah-langkah:

1. **Dekomposisi:** Rancang respons berdasarkan tiga jenis emosi utama.
2. **Desain:** Input teks atau tombol perasaan → output pesan dukungan.
3. **Logika:** Buat blok logika yang cocokkan input pengguna dengan jawaban tepat.
4. **Abstraksi:** Tidak perlu memproses kalimat kompleks, cukup kata kunci.
5. **Refleksi:** Bahas bagaimana logika membantu kita bersikap empati dan solutif.

Masing-masing proyek ini **melatih berpikir logis, kreatif, dan sistematis**, sekaligus memberi dampak langsung dalam kehidupan siswa. Proyek-proyek ini juga membuka ruang eksplorasi ide, mendorong kolaborasi, serta menunjukkan bahwa teknologi bukan hanya untuk ahli, tapi juga untuk pelajar muda yang berani berpikir cerdas!

BAB 7: Mengenal Domain .id



Apa Itu Domain?

Bayangkan internet seperti **kota besar** yang punya **jutaan rumah**. Setiap rumah di kota itu punya alamat agar bisa dikunjungi. Nah, nama **domain** itu seperti **alamat kita di internet**, nanti bisa digunakan sebagai alamat website maupun alamat email kita. Dan kita bisa mempunyai alamat email sendiri misalnya halo@meita.id. Misalnya alamat website

Contohnya:

- pandi.id
- home.s.id
- meita.my.id

Alamat itu disebut **nama domain**, dan tanpa domain, kamu harus mengingat angka-angka ribet yang disebut **IP address**. Pasti gak seru, kan?

Apa Itu .id?

Sekarang coba perhatikan domain ini: www.sekolahku.id

Nah, yang bagian **".id"** itu disebut **ekstensi domain**. ".id" menunjukkan bahwa website itu **berasal dari Indonesia**. Setiap negara punya ekstensinya masing-masing, contohnya:

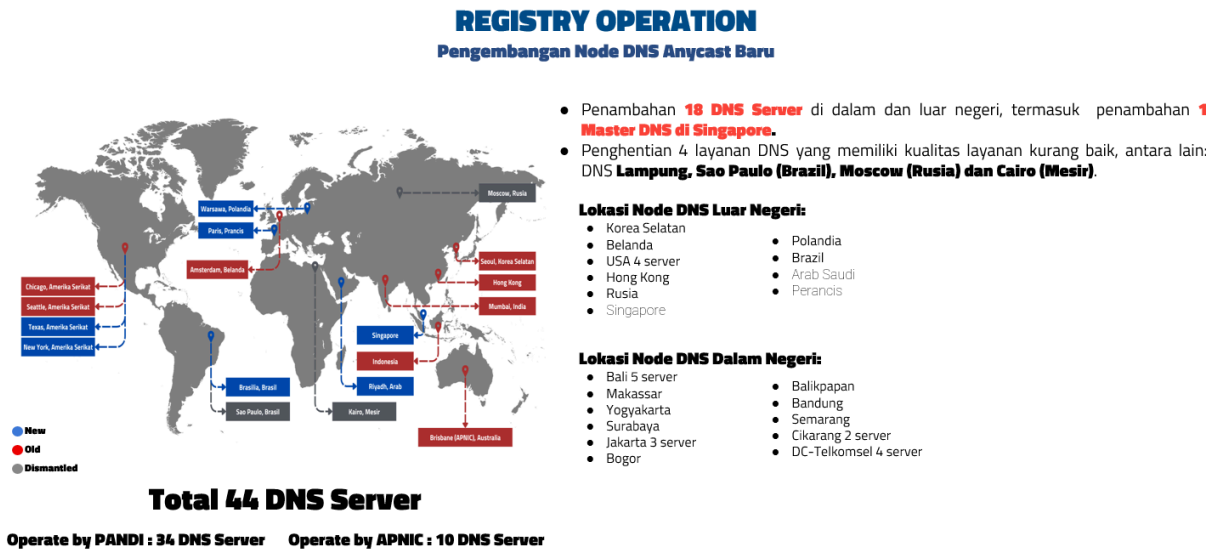
- .jp → Jepang
- .us → Amerika Serikat
- .id → Indonesia 🇮🇩

Menggunakan domain .id itu keren, karena:

- Menunjukkan kebanggaan sebagai orang Indonesia
- Membuat situsmu terlihat lebih terpercaya di mata orang Indonesia
- Cocok banget buat bisnis lokal, sekolah, komunitas, atau bahkan blog pribadi

Keunggulan domain .id

- Representasi Indonesia
  Menunjukan bahwa ini entitas ataupun personal dari Indonesia karena .id adalah kode negara Indonesia
- Sebagai identity
  Selain reprensentasi Indonesia .id juga bisa berarti sebagai identity atau identitas, sehingga sangat menarik untuk dijadikan sebuah alamat website, email maupun brand di internet



Siapa Itu PANDI?

PANDI adalah singkatan dari **Pengelola Nama Domain Internet Indonesia.**
Coba bayangkan **PANDI** seperti **penjaga gerbang dunia .id**. Mereka:

- Mengatur dan mengelola semua domain dengan **akhiran .id**
- Menjaga agar **domain .id aman, tertata, dan tidak disalahgunakan**
- Memberi kesempatan buat siapa saja — termasuk kamu — **punya domain sendiri!**

Misalnya kamu ingin punya website "tokohfavoritku.id", kamu bisa daftar lewat penyedia domain resmi, dan **PANDI** yang mengatur sistemnya agar website-mu bisa diakses semua orang.

Kenapa Kamu Harus Tahu Ini?

- Siapa tahu **nanti kamu bikin website** untuk tugas sekolah, bisnis kecil-kecilan, atau komunitas temanmu
- Supaya kamu melek teknologi, karena **dunia digital adalah masa depan**
- Karena kamu bisa bikin identitas online-mu sendiri dan itu sangat keren!

Kalau kamu bisa punya akun Instagram, kamu juga bisa punya domain sendiri. Bayangkan: www.namamu.id – tempat kamu membagikan karya, foto, tulisan, atau bahkan jualan!

Lebih dalam tentang PANDI

PANDI adalah Registry Nama Domain Indonesia yang berperan untuk mengoperasikan, memelihara dan mengelola Nama Domain Indonesia (.id)

Pengelola Nama Domain Internet Indonesia (PANDI) merupakan perkumpulan yang terdiri dari berbagai pemangku kepentingan, termasuk perwakilan dari pemerintah, operator industri internet, dan akademisi.

Didirikan pada tahun 2006, **PANDI** menerima Redelegasi dari **Internet Assigned Numbers Authority (IANA)** sebagai Registry .id pada tahun 2013.

Hingga 31 Desember 2024 jumlah domain .id yang terdaftar tumbuh menjadi yang terbesar di Asia Tenggara, dengan total 1.215.714 Nama Domain. Nama Domain .id sendiri bisa diartikan sebagai Indonesia, identitas, ide, dan lainnya. Sejalan dengan makna tersebut, **PANDI** juga memiliki kebijakan khusus dalam menyelesaikan perselisihan nama domain .id.

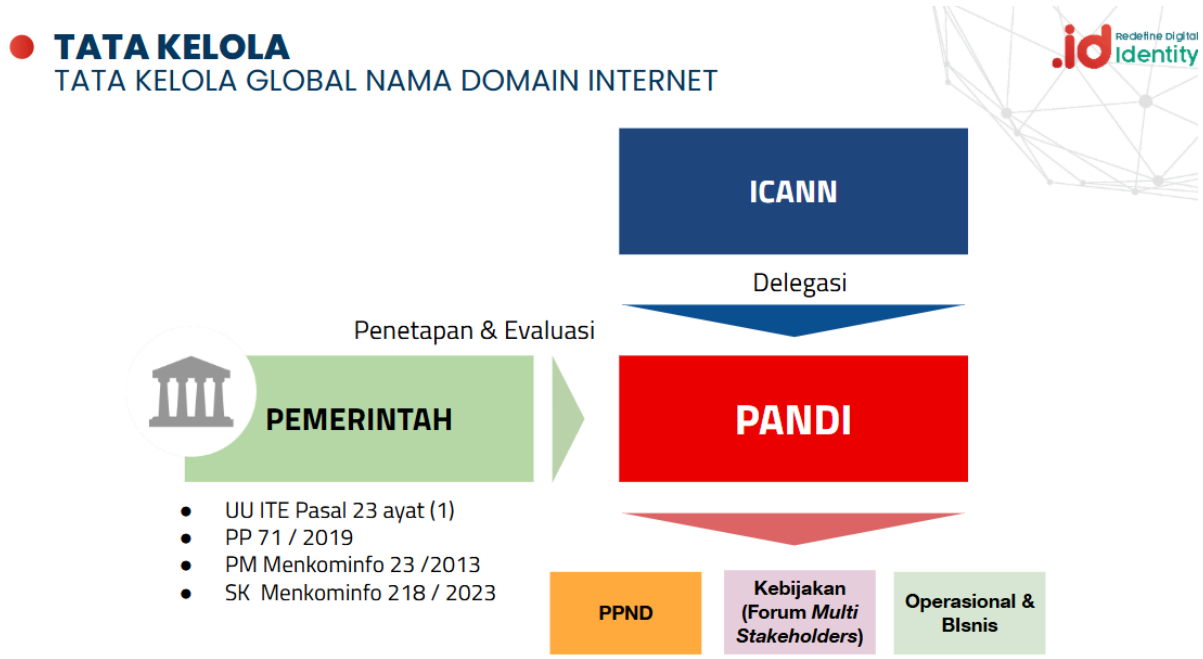
Tugas dan Kewajiban PANDI

Tugas

- Merumuskan kebijakan di bidang pengelolaan Nama Domain Tingkat Tinggi Indonesia
- Menyiapkan, mengoperasikan, dan memelihara infrastruktur yang dibutuhkan serta menyediakan sistem elektronik untuk pengelolaan Nama Domain Tingkat Tinggi Indonesia
- Menyelenggarakan pendaftaran Nama Domain Tingkat Tinggi Indonesia sesuai dengan ketentuan peraturan perundang-undangan, kepatutan yang berlaku dalam masyarakat, dan prinsip kehati-hatian
- Melaksanakan seleksi Registrar Nama Domain
- Memberikan peringatan kepada Registrar Nama Domain jika terindikasi melakukan pelanggaran
- Mencabut hak operasional Registrar Nama Domain jika terbukti melakukan pelanggaran, dan
- Melakukan pengawasan operasional dan teknis Registrar Nama Domain

Kewajiban

- Menjamin sistem elektronik Registri Nama Domain Indonesia beroperasi dengan baik, stabil, aman didukung dengan layanan yang dapat diandalkan
- Menempatkan pusat data dan pusat pemulihan bencana di wilayah Indonesia
- Melakukan pengawasan terhadap Registrar Nama Domain
- Memfasilitasi penyelesaian perselisihan Nama Domain Indonesia
- Melaporkan daftar Registrar Nama Domain kepada Menteri
- Mengikuti ketentuan pengelolaan Nama Domain internasional dan peraturan perundang-undangan
- Menyampaikan laporan berkala kepada Menteri sekurang-kurangnya 1 (satu) kali dalam setahun
- Membayar pungutan biaya Pengelolaan Nama Domain Indonesia



Tata Kelola Internet

PANDI berkomitmen penuh untuk terus berkontribusi aktif dalam tata kelola internet global dengan mendukung infrastruktur yang aman dan inovatif. Hal ini karena domain .id merupakan bagian penting dari ekosistem digital dunia yang inklusif dan berkelanjutan.

Di tahun 2024, PANDI menginisiasi berbagai wadah tata kelola internet nasional serta berpartisipasi aktif dalam forum tata kelola internet global, seperti:

- Forum Multi Stakeholder (FMS),
- PANDI Meeting 2024,
- ICANN 79 di Puerto Riko,
- ICANN 81 di Turki,
- APTLD 86 di Vietnam,
- APAC DNS Forum 2024 yang diselenggarakan oleh PANDI di Bali.

Selain APAC DNS Forum, PANDI juga menggelar program Indonesia Internet Governance Academy (IDIGA) yang bertujuan mencetak pemimpin muda di bidang tata kelola internet, baik di tingkat nasional maupun internasional.

Institut Teknologi Tangerang Selatan (ITTS) dan Penulis.

Para penulis adalah **dosen aktif di Institut Teknologi Tangerang Selatan (ITTS)**, sebuah **institusi pendidikan tinggi berbasis teknologi** yang berperan aktif dalam pengembangan **solusi digital** dan *kecerdasan buatan (AI)*. Tidak hanya mengajar, mereka juga merupakan **praktisi dan inovator** yang menghasilkan **karya-karya nyata di bidang teknologi mandiri**.

Berlokasi di **Komplek Komersial BSD, Serpong Utara, Tangerang Selatan**, **ITTS hadir sebagai pusat inovasi** yang mengintegrasikan **pembelajaran, penelitian, dan pengembangan teknologi secara menyeluruh**. Fokus utama ITTS mencakup bidang strategis seperti *AI, Machine Learning, Deep Learning*, jaringan, *cloud*, keamanan siber, pemrograman, sistem informasi, dan multimedia.

ITTS menawarkan lingkungan akademik yang dinamis dan kolaboratif. Setiap minggu, diselenggarakan berbagai *workshop*, *demo teknologi*, seminar, hingga *webinar*—mayoritas **gratis dan tersedia secara terbuka di YouTube**. Ini menciptakan **kultur belajar yang aktif dan relevan dengan perkembangan industri**.

Fasilitas ITTS menunjang kegiatan riset dan inovasi secara optimal. Lab komputer dilengkapi *GPU RTX 4060* untuk eksperimen *AI*, serta **laboratorium teknologi dengan puluhan server, perangkat *IoT*, sistem komunikasi radio, dan infrastruktur jaringan jarak jauh**. Dari ekosistem ini lahirlah **produk-produk mandiri** seperti **ChatGPT versi lokal** untuk institusi pendidikan dan **sistem perpustakaan digital yang dapat diakses tanpa koneksi internet**.

Daya saing ITTS juga tercermin dari karakter dosen dan mahasiswa yang tidak hanya akademik, tetapi juga **produktif secara praktis**. Mahasiswa didorong aktif menjadi **narasumber, penulis, dan peneliti** sejak dini. Banyak di antaranya telah menghasilkan **karya buku, artikel ilmiah**, serta **terlibat langsung dalam proyek-proyek teknologi**.

Kolaborasi strategis dengan berbagai mitra industri—seperti *IDCloudHost, Ubuntu, XecureIT, Dinas KOMINFO Tangerang Selatan*, berbagai *ISP*, dan komunitas IT nasional—menjadi **jembatan nyata antara kampus dan dunia profesional**. Hasilnya, **lebih dari 75% mahasiswa ITTS telah bekerja bahkan sebelum lulus**, dan **hampir seluruh lulusan langsung terserap industri**.

ITTS bukan hanya tempat belajar, tapi tempat mencipta. Dosen sebagai **penulis dan praktisi**, mahasiswa sebagai **peneliti dan kontributor**—semuanya berperan **membangun ekosistem teknologi yang mandiri dan berdaya saing tinggi**.

Institut Teknologi Tangerang Selatan (ITTS)
Komplek Komersial BSD
Jl. Raya Serpong No. Kav. 9, Lengkong Karya, Serpong Utara
Kota Tangerang Selatan, Banten 15331
📞 (+62) 877-7277-1775 📧 info@itts.ac.id 🌐 www.itts.ac.id