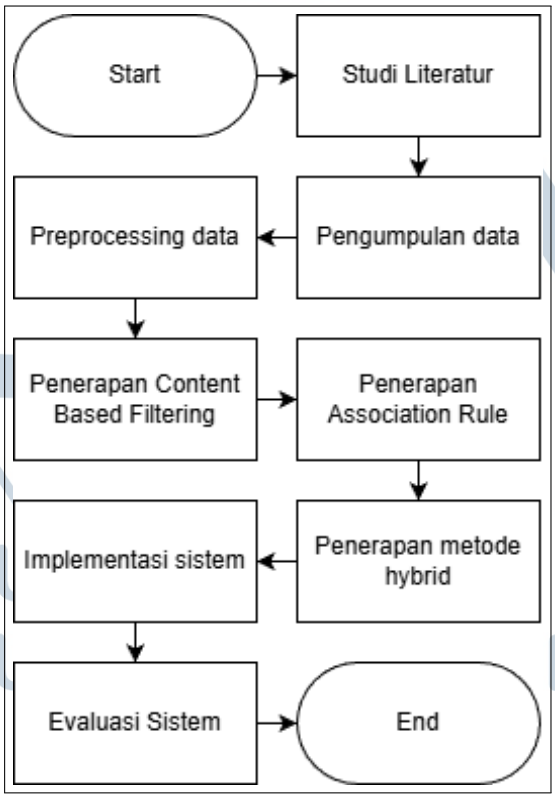
BAB 3
METODOLOGI PENELITIAN

Pada bagian ini dijelaskan tahapan-tahapan yang dilakukan dalam penyusunan dan pelaksanaan penelitian. Metodologi penelitian disusun secara sistematis untuk memastikan proses penelitian berjalan terarah dan sesuai dengan tujuan yang ingin dicapai, yaitu merancang sistem rekomendasi video game menggunakan metode Association Rule dan Content-Based Filtering.

Pada Gambar 3.1 ditunjukkan alur metodologi penelitian yang diawali dengan studi literatur untuk memahami konsep sistem rekomendasi serta metode yang digunakan. Selanjutnya dilakukan pengumpulan data dan preprocessing data untuk memastikan kualitas data. Data yang telah diproses kemudian digunakan pada dua pendekatan, yaitu Content-Based Filtering dan Association Rule menggunakan algoritma FP-Growth. Hasil dari kedua metode tersebut digabungkan menggunakan pendekatan hybrid dengan pemberian bobot pada masing-masing metode. Sistem kemudian diimplementasikan dan dievaluasi untuk mengukur kinerja rekomendasi yang dihasilkan.



Gambar 3.1. Flowchart metodologi penelitian

3.1 Studi Literatur

Sebelum memulai proses pengembangan aplikasi sistem rekomendasi video game, dilakukan terlebih dahulu studi literatur sebagai landasan teoritis dan metodologis penelitian. Studi literatur ini bertujuan untuk memperoleh pemahaman yang komprehensif mengenai konsep, metode, serta pendekatan yang telah diterapkan pada penelitian-penelitian sebelumnya yang berkaitan dengan sistem rekomendasi, khususnya pada domain video game.

Berdasarkan hasil studi literatur, ditemukan berbagai penelitian terdahulu yang mengembangkan sistem rekomendasi video game dengan menggunakan beragam metode. Salah satu penelitian membahas rekomendasi game edukasi dengan menggunakan metode *Item-Based Collaborative Filtering*. Metode ini memanfaatkan tingkat kesamaan antar item berdasarkan pola interaksi pengguna untuk menghasilkan rekomendasi yang relevan. Hasil penelitian menunjukkan bahwa sistem yang dibangun dinilai efektif oleh responden, terutama ketika data interaksi pengguna tersedia dalam jumlah yang memadai, meskipun masih menghadapi keterbatasan pada jumlah responden dan potensi permasalahan *cold-start*.

Penelitian selanjutnya mengimplementasikan sistem rekomendasi game pada platform digital Steam dengan pendekatan *Hybrid Filtering*, yang mengombinasikan metode *Neural Collaborative Filtering* (NCF) dan *Content-Based Filtering*. Penelitian ini menunjukkan bahwa model NCF mampu mencapai performa optimal berdasarkan nilai *loss* dan RMSE, serta metode *cosine similarity* berhasil memberikan rekomendasi yang relevan. Namun demikian, penelitian ini belum menjelaskan secara rinci mekanisme pembobotan dalam penggabungan kedua metode tersebut.

Penelitian lainnya berfokus pada sistem rekomendasi video game berbasis usia untuk mendukung pengawasan orang tua di platform Steam dengan menggunakan metode *Content-Based Filtering*. Sistem ini menghasilkan nilai evaluasi yang sangat tinggi, seperti precision dan recall, serta nilai kesalahan yang relatif rendah. Meskipun demikian, rekomendasi yang dihasilkan masih terbatas karena hanya mempertimbangkan kesesuaian usia dan kemiripan konten game tanpa melibatkan preferensi atau interaksi pengguna.

Selain itu, penelitian lainnya mengkaji penentuan rekomendasi pembelian video game dengan menggunakan metode *Simple Additive Weighting* (SAW). Metode ini digunakan untuk membantu pengambilan keputusan berdasarkan

beberapa kriteria penilaian yang telah ditentukan. Hasil penelitian menunjukkan bahwa SAW mampu mendukung proses pengambilan keputusan, namun sistem yang dibangun masih memiliki keterbatasan dalam aspek personalisasi serta belum disertai evaluasi kuantitatif yang memadai.

Penelitian terakhir menerapkan metode *Case Based Reasoning* (CBR) dalam pembuatan sistem rekomendasi judul gim. Sistem rekomendasi dihasilkan dengan menghitung tingkat kemiripan antara kasus baru dan kasus-kasus yang telah tersimpan sebelumnya. Meskipun sistem mampu memberikan rekomendasi berdasarkan kemiripan tertinggi, penelitian ini sangat bergantung pada ketersediaan basis kasus dan belum membahas permasalahan *cold-start* maupun evaluasi kuantitatif terhadap kualitas rekomendasi.

Berdasarkan ringkasan studi literatur pada Tabel 3.1, dapat disimpulkan bahwa berbagai metode sistem rekomendasi video game telah dikembangkan dengan keunggulan dan keterbatasannya masing-masing. Metode Collaborative Filtering dan turunannya mampu menghasilkan rekomendasi yang relevan ketika data interaksi pengguna tersedia dalam jumlah besar, namun masih menghadapi permasalahan cold-start dan ketergantungan pada data eksplisit pengguna. Sementara itu, Content-Based Filtering efektif dalam memanfaatkan atribut game dan mampu menghindari cold-start, tetapi tingkat personalisasi yang dihasilkan masih terbatas karena hanya bergantung pada kesamaan konten. Pendekatan lain seperti SAW dan CBR juga mampu mendukung pengambilan keputusan, namun umumnya belum dilengkapi dengan evaluasi kuantitatif yang komprehensif serta kurang mempertimbangkan pola perilaku pengguna secara implisit.

Selain itu, beberapa penelitian hibrida belum menjelaskan secara rinci mekanisme penggabungan metode maupun evaluasi kinerja sistem secara menyeluruh. Oleh karena itu, diperlukan penelitian baru yang mengombinasikan metode Association Rule dan Content-Based Filtering. Metode Association Rule digunakan untuk menangkap pola keterkaitan antar game berdasarkan data transaksi atau preferensi pengguna secara implisit, sedangkan Content-Based Filtering dimanfaatkan untuk mempertahankan relevansi rekomendasi berdasarkan karakteristik game. Kombinasi kedua metode ini diharapkan mampu menghasilkan sistem rekomendasi video game yang lebih akurat, relevan, dan memiliki tingkat personalisasi yang lebih baik dibandingkan penggunaan metode tunggal.

Tabel 3.1. Ringkasan studi literatur terkait metode rekomendasi

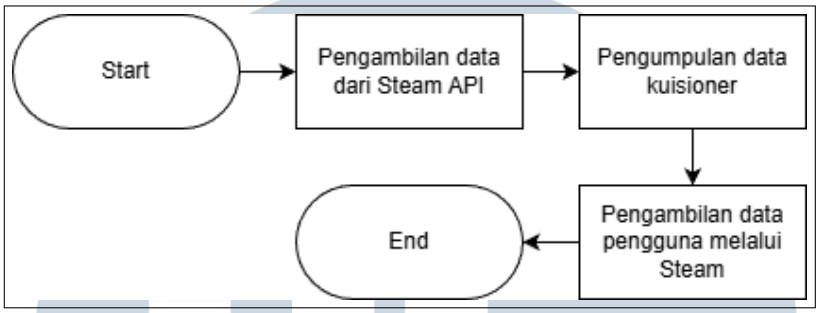
Judul Penelitian	Hasil Penelitian	Metode yang Digunakan	Celah Penelitian
Rekomendasi Game Edukasi Dengan Menggunakan Metode Item-Based Collaborative Filtering[10]	Sistem dinilai efektif, seluruh responden memberikan penilaian positif terhadap manfaat dan kegunaan rekomendasi	Item-Based Collaborative Filtering	Keterbatasan jumlah responden, penggunaan data interaksi eksplisit yang berpotensi menimbulkan masalah cold-start, serta belum diterapkannya pendekatan hybrid atau perbandingan dengan metode rekomendasi lain
Implementasi Sistem Rekomendasi Game Menggunakan Hybrid Filtering pada Platform Digital Steam[11]	Hasil pengujian menunjukkan model NCF mencapai performa optimal dengan nilai loss 0,3602 dan RMSE 0,2211 dan hasil metode cosine similarity berhasil memberikan rekomendasi yang relevan	Neural Collaborative Filtering (NCF)d dan Content-Based Filtering	Belum ada penjelasan mekanisme pembobotan dalam penggabungan hasil Neural Collaborative Filtering dan Content-Based Filtering, serta belum menyajikan hasil evaluasi kuantitatif terhadap kinerja metode hibrida secara keseluruhan
Sistem Rekomendasi Video Game Berbasis Usia untuk Pengawasan Orang Tua di Steam Menggunakan Content-Based Filtering[12]	Sistem menghasilkan nilai precision sebesar 0,98 dan recall sebesar 1,00 dan nilai Mean Absolute Error (MAE) yang diperoleh sebesar 0,469236, Mean Squared Error (MSE) sebesar 6,440935, dan Root Mean Squared Error (RMSE) sebesar 2,537900	Content-Based Filtering	Sistem rekomendasi hanya mengandalkan penyaringan usia dan kemiripan konten game tanpa mempertimbangkan preferensi atau interaksi pengguna, sehingga tingkat personalisasi rekomendasi masih terbatas
Menentukan Rekomendasi Pembelian Video Game Dengan Menggunakan Metode Simple Additive Weighting[13]	SAW mampu membantu pengambilan keputusan pembelian berdasarkan penilaian multi-kriteria	Simple Additive Weighting	Keterbatasan pada transparansi sumber data penilaian, rendahnya personalisasi rekomendasi, serta tidak adanya evaluasi kuantitatif dan perbandingan metode
Penerapan Case Based Reasoning (CBR) dalam Pembuatan Sistem Rekomendasi Judul Gim[14]	Sistem mampu memberikan rekomendasi judul gim berdasarkan nilai kemiripan tertinggi antara kasus baru dan kasus lama	Case Based Reasoning	Sistem sangat bergantung pada basis kasus yang tersedia, tidak membahas cold start, serta tidak disertai evaluasi kuantitatif seperti akurasi, precision, atau recall sehingga kualitas rekomendasi tidak dapat diukur secara objektif

3.2 Pengumpulan Data

Pada tahap ini dilakukan proses pengumpulan data yang menjadi dasar dalam pembangunan sistem rekomendasi video game. Data yang dikumpulkan mencakup informasi game serta preferensi pengguna yang diperoleh dari berbagai sumber. Proses pengumpulan data ini bertujuan untuk memastikan bahwa dataset yang digunakan bersifat relevan, akurat, dan representatif sehingga dapat mendukung proses analisis dan penerapan metode rekomendasi secara optimal.

Pada Gambar 3.2, pengumpulan data diawali dengan memperoleh data informasi game diperoleh melalui Steam Web API dengan mengambil sebanyak 258 judul game, yang mencakup atribut penting seperti nama game, genre, dan category. Data ini digunakan dalam proses Content-Based Filtering. Kedua, data preferensi pengguna dikumpulkan melalui penyebaran kuesioner menggunakan Google Form kepada sekitar 40 responden, yang berisi informasi mengenai game-game yang disukai atau pernah dimainkan oleh pengguna. Data ini dimanfaatkan sebagai data sekunder dalam pembentukan pola asosiasi pada metode Association Rule. Selain itu, pengambilan data juga dilakukan secara langsung melalui aplikasi Steam, yaitu dengan mengamati friend list serta profil pengguna yang bersifat publik (tidak ter-private) untuk memperoleh informasi tambahan mengenai daftar game yang dimiliki atau dimainkan. Data dari pengamatan ini digunakan untuk memperkaya

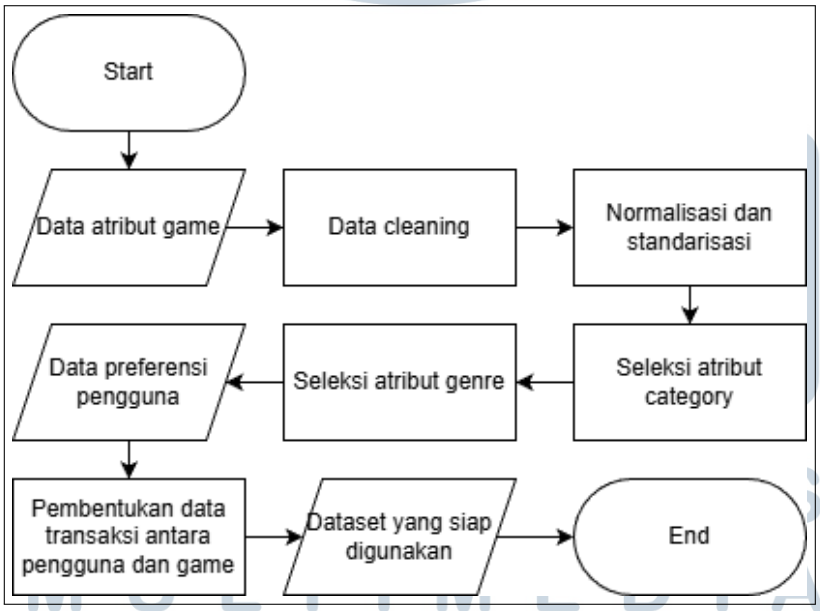
variasi preferensi pengguna dan mendukung proses analisis pola asosiasi dalam sistem rekomendasi.



Gambar 3.2. Flowchart pengumpulan data

3.3 Preprocessing Data

Pada tahap ini, data yang telah diperoleh dari proses pengumpulan data diolah terlebih dahulu agar layak dan sesuai untuk digunakan dalam proses analisis. Proses *preprocessing* dilakukan melalui beberapa langkah sebagai berikut yang dapat dilihat pada Gambar 3.3.



Gambar 3.3. Flowchart preprocesing data

Langkah pertama adalah pembersihan data (*data cleaning*). Pada tahap ini dilakukan verifikasi terhadap seluruh data game yang diambil dari Steam API untuk memastikan bahwa seluruh entri merupakan game *standalone*, bukan DLC

(*Downloadable Content*), bukan modifikasi (*mod*), dan bukan aplikasi tambahan seperti *Wallpaper Engine*, *Bongo Cat*, atau aplikasi non-game lainnya. Game yang tidak memenuhi kriteria tersebut dihapus atau diganti dengan data game lain agar dataset tetap konsisten dan relevan dengan tujuan penelitian.

Langkah berikutnya adalah normalisasi dan standarisasi atribut, seperti penulisan nama *genre* dan *category* agar seragam (misalnya huruf kecil seluruhnya dan penghapusan simbol yang tidak diperlukan). Setelah itu, data game yang telah bersih diubah ke dalam bentuk *feature matrix* berdasarkan *genre* dan *category* untuk digunakan dalam metode *Content-Based Filtering*.

Selain memastikan data game yang digunakan merupakan game *standalone* dan bukan DLC, *mod*, atau aplikasi tambahan, proses *preprocessing* juga dilakukan pada atribut *category*. Data *category* yang diperoleh dari Steam API mengandung berbagai jenis informasi, seperti fitur aksesibilitas, dukungan perangkat, mode permainan, layanan Steam, hingga fitur teknis lainnya. Tidak semua kategori tersebut relevan untuk proses *Content-Based Filtering*, karena metode ini membutuhkan atribut yang dapat menggambarkan karakteristik atau konten dari game itu sendiri.

Oleh karena itu, dilakukan seleksi kategori dengan hanya mengambil kategori yang relevan terhadap konten atau pengalaman bermain game, seperti:

- *Co-Op*, Mode permainan di mana dua atau lebih pemain bekerja sama untuk mencapai tujuan yang sama.
- *LAN PvP*, Mode *Player vs Player* yang dimainkan melalui jaringan lokal (*Local Area Network*).
- *MMO (Massively Multiplayer Online)*, Mode permainan daring yang memungkinkan sangat banyak pemain berpartisipasi secara bersamaan dalam satu dunia permainan yang sama.
- *Multiplayer*, Mode permainan yang memungkinkan lebih dari satu pemain bermain secara bersamaan, baik secara bekerja sama maupun saling bersaing.
- *Online Co-Op*, Mode kerja sama antar pemain yang dimainkan secara daring melalui koneksi internet.
- *Online PvP*, Mode *Player vs Player* yang dimainkan secara daring melawan pemain lain melalui koneksi internet.

- *PvP (Player vs Player)*, Mode permainan di mana pemain saling bertanding satu sama lain, bukan melawan sistem atau kecerdasan buatan.
- *Shared / Split Screen*, Mode *multiplayer* lokal di mana beberapa pemain bermain pada satu perangkat dengan tampilan layar yang dibagi.
- *Shared / Split Screen Co-Op*, Mode kerja sama lokal di satu perangkat dengan tampilan layar terbagi.
- *Shared / Split Screen PvP*, Mode *Player vs Player* lokal di satu perangkat dengan tampilan layar terbagi.
- *Single Player*, Mode permainan yang dimainkan oleh satu pemain tanpa melibatkan pemain lain.

Kategori yang tidak mencerminkan konten *gameplay* atau tidak berpengaruh terhadap kesamaan konten antar game—seperti *Steam Achievements*, *Steam Cloud*, *Family Sharing*, *Full Controller Support*, *In-App Purchases*, fitur *VR*, fitur aksesibilitas (*Adjustable Text Size, Color Alternatives, Captions Available*), atau layanan teknis lainnya dikeluarkan dari dataset karena tidak memberikan kontribusi signifikan dalam proses perhitungan kemiripan konten.

Dengan demikian, data *category* yang tersisa hanya berupa elemen-elemen yang mampu merepresentasikan gaya bermain (*playstyle*) atau tipe pengalaman game, sehingga lebih sesuai untuk digunakan dalam *Content-Based Filtering*.

Selain kategori permainan, atribut *genre* juga digunakan sebagai fitur utama dalam proses *Content-Based Filtering*. Genre berfungsi untuk merepresentasikan jenis permainan secara umum dan karakteristik gameplay utama dari setiap game. Berdasarkan data yang diperoleh dari Steam API dan hasil proses *preprocessing*, genre yang berhasil terambil dan digunakan dalam penelitian ini adalah sebagai berikut:

- *Action* Genre yang menekankan pada refleks, kecepatan, serta koordinasi tangan dan mata pemain, umumnya melibatkan pertarungan atau tantangan berbasis aksi secara real-time.
- *Adventure* Genre yang berfokus pada eksplorasi, alur cerita, dan pemecahan teka-teki, dengan penekanan pada pengalaman naratif.
- *Casual* Genre yang dirancang untuk dimainkan secara santai dengan mekanisme sederhana dan tingkat kesulitan yang relatif rendah.

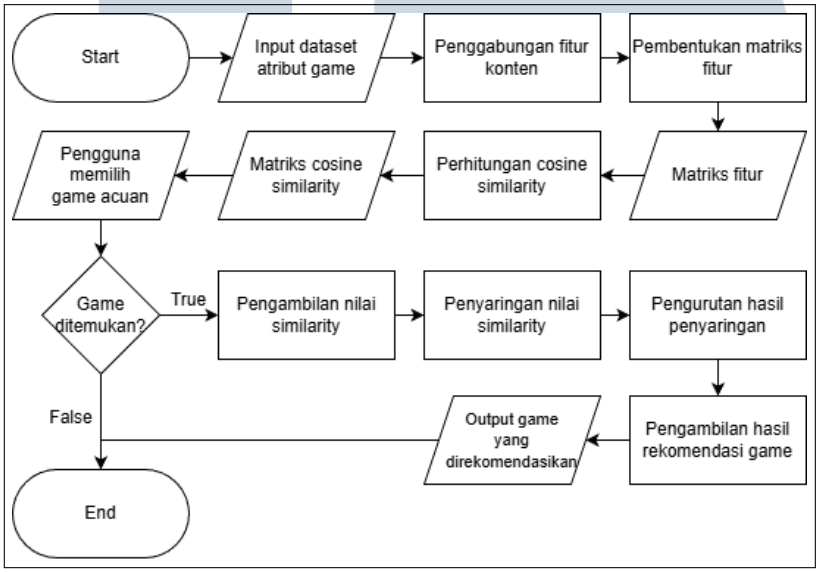
- *Early Access* Genre atau kategori yang menunjukkan bahwa game masih dalam tahap pengembangan dan dapat dimainkan oleh pengguna sebelum versi final dirilis.
- *Free To Play* Genre atau model permainan yang dapat dimainkan secara gratis, dengan kemungkinan adanya sistem monetisasi tambahan di dalam game.
- *Indie* Genre yang mencakup game yang dikembangkan oleh pengembang independen, umumnya dengan skala pengembangan yang lebih kecil dan konsep yang unik atau eksperimental.
- *Massively Multiplayer* Genre yang memungkinkan sejumlah besar pemain bermain secara bersamaan dalam satu lingkungan permainan yang sama.
- *Racing* Genre yang berfokus pada kompetisi balapan menggunakan kendaraan, dengan tujuan utama mencapai garis akhir dalam waktu tercepat.
- *RPG (Role-Playing Game)* Genre yang menekankan pada pengembangan karakter, sistem level, dan pengambilan keputusan yang memengaruhi alur permainan.
- *Simulation* Genre yang bertujuan untuk mensimulasikan aktivitas atau sistem dunia nyata secara realistis, seperti mengemudi, membangun kota, atau mengelola sumber daya.
- *Sports* Genre yang merepresentasikan cabang olahraga tertentu, baik secara realistis maupun dengan pendekatan arkade.
- *Strategy* Genre yang menekankan pada perencanaan, pengambilan keputusan, dan pengelolaan sumber daya untuk mencapai kemenangan.

Sementara itu, data preferensi pengguna dari Google Form diolah menjadi bentuk transaksi pengguna (*user–item transaction*), yaitu daftar game yang disukai atau dimainkan oleh setiap responden. Data ini kemudian disiapkan untuk penerapan metode *Association Rule* guna menemukan pola keterkaitan antar game.

3.4 Penerapan Content Based Filtering

Pada tahap ini, sistem rekomendasi mulai memanfaatkan karakteristik atau konten dari setiap game untuk menghasilkan rekomendasi yang relevan.

Pendekatan yang digunakan adalah *Content-Based Filtering*, yaitu metode yang bekerja dengan cara membandingkan kesamaan fitur antar game berdasarkan informasi seperti genre dan kategori. Dengan memanfaatkan deskripsi konten tersebut, sistem dapat mengidentifikasi game-game lain yang memiliki karakteristik serupa, sehingga rekomendasi yang diberikan menjadi lebih personal dan konsisten dengan preferensi pengguna. Penerapan metode ini dilakukan dalam berbagai tahap yang dapat dilihat pada Gambar 3.4.



Gambar 3.4. Flowchart penerapan Content Based Filtering

Penerapan metode *Content-Based Filtering* pada sistem rekomendasi video game. Proses diawali dengan memasukkan dataset game yang berisi berbagai atribut atau fitur konten. Selanjutnya, seluruh fitur konten tersebut digabungkan untuk membentuk representasi data yang digunakan dalam proses analisis.

Tahap berikutnya adalah perhitungan tingkat kemiripan antar game menggunakan metode *Cosine Similarity*. Setelah itu, pengguna menentukan game acuan sebagai dasar rekomendasi, kemudian sistem mengambil nilai kemiripan yang relevan dan melakukan penyaringan berdasarkan nilai similarity tertentu. Hasil yang telah disaring selanjutnya diurutkan berdasarkan tingkat kemiripan dari yang tertinggi hingga terendah.

Pada tahap akhir, sistem melakukan pemilihan Top-*N* rekomendasi untuk menghasilkan output berupa daftar game yang direkomendasikan kepada pengguna. Proses ini berakhir setelah rekomendasi berhasil ditampilkan.

Pada potongan Kode 3.1, sistem melakukan penggabungan dua atribut utama

yang merepresentasikan konten game, yaitu *genres* dan *categories*. Kedua atribut tersebut disatukan ke dalam satu kolom baru bernama `combined_features`. Proses ini bertujuan untuk menyediakan deskripsi tekstual yang lebih komprehensif bagi setiap game sebelum dilakukan ekstraksi fitur. Selain itu, fungsi `fillna("")` digunakan untuk memastikan tidak ada nilai kosong yang dapat mengganggu proses pembentukan fitur.

```
df["combined_features"] = (
    df["genres"].fillna("") + " " +
    df["categories"].fillna("")
)
```

Kode 3.1: Kode penggabungan atribut genre dan kategori sebagai fitur konten game

Sebagai ilustrasi, game Apex Legends memiliki atribut:

- *Genres*:
  "*Action, Adventure, Free To Play*"
- *Categories*:
  "*Multi-player, PvP, Online PvP, Co-op, Online Co-op, Steam Achievements, Full controller support, Steam Trading Cards, In-App Purchases, Chat Speech-to-text, Playable without Timed Input*"

Setelah melalui proses penggabungan pada kode di atas, kedua atribut tersebut membentuk satu nilai string baru pada kolom `combined_features` sebagai berikut:

- `combined_features`:
  "*Action, Adventure, Free To Play, Multi-player, PvP, Online PvP, Co-op, Online Co-op, Steam Achievements, Full controller support, Steam Trading Cards, In-App Purchases, Chat Speech-to-text, Playable without Timed Input*"

Dengan demikian, setiap game memiliki deskripsi tekstual yang lebih informatif yang mencerminkan karakteristik kontennya. Kolom `combined_features` inilah yang selanjutnya digunakan sebagai dasar dalam proses Content-Based Filtering untuk menghitung tingkat kemiripan antar game menggunakan ukuran seperti cosine similarity.

Pada potongan Kode 3.2, sistem memanfaatkan *CountVectorizer* untuk mengubah data teks pada kolom `combined_features` menjadi representasi numerik berbentuk matriks fitur.

```
vectorizer = CountVectorizer(stop_words="english")
feature_matrix = vectorizer.fit_transform(df["combined_features"])

cosine_sim = cosine_similarity(feature_matrix, feature_matrix)
```

Kode 3.2: Kode transformasi fitur konten menjadi matriks vektor dan perhitungan cosine similarity

Pada Tabel 3.2 ditampilkan contoh data pada kolom `combined_features` untuk dua game, yaitu *Apex Legends* dan *Counter-Strike 2*. *Apex Legends* merupakan game *first-person shooter* (FPS) dengan genre *battle royale* yang menekankan permainan tim serta kemampuan karakter yang beragam, sedangkan *Counter-Strike 2* adalah game FPS kompetitif yang berfokus pada pertarungan taktis antara dua tim dengan mekanisme permainan yang lebih sederhana namun strategis. Kolom `combined_features` merupakan hasil penggabungan atribut *genres* dan *categories*, sehingga menghasilkan representasi teks yang lebih lengkap mengenai karakteristik konten setiap game. Informasi ini menjadi dasar dalam proses ekstraksi fitur pada metode *Content-Based Filtering*. Dengan menggabungkan kedua atribut tersebut, sistem dapat menangkap lebih banyak kata kunci yang relevan sehingga meningkatkan kualitas perhitungan kemiripan antar game.

Tabel 3.2. Contoh data *combined_features* untuk dua game

Game	**combined_features**
Apex Legends	Action Adventure Free To Play Multi-player PvP Online PvP Co-op Online Co-op Steam Achievements Full controller support Steam Trading Cards In-App Purchases Chat Speech-to-text Playable without Timed Input
Counter-Strike 2	Action Free To Play Multi-player PvP Online PvP Co-op Online Co-op Steam Achievements Competitive Cross-Platform Multiplayer Valve Anti-Cheat

Selanjutnya, Tabel 3.3 menunjukkan hasil tokenisasi dari proses CountVectorizer, di mana setiap kata unik yang muncul pada `combined_features` dijadikan fitur, dan masing-masing game direpresentasikan sebagai nilai frekuensi kemunculan kata tersebut. Sebagai contoh, kata "action", "free", "multi-player", dan "pvp" muncul pada kedua game, sehingga memiliki nilai 1 untuk masing-masing game. Sebaliknya, kata seperti "adventure" hanya muncul pada Apex Legends, sedangkan kata "competitive" dan "anti-cheat" hanya muncul pada Counter-Strike 2. Perbedaan pola kemunculan kata inilah yang membentuk vektor

fitur bagi setiap game dan menjadi dasar penghitungannya melalui metode cosine similarity. Dengan demikian, kedua tabel tersebut menggambarkan bagaimana data teks diolah menjadi representasi numerik yang dapat digunakan untuk mengukur tingkat kemiripan antar game dalam sistem rekomendasi berbasis konten.

Tabel 3.3. Contoh hasil tokenisasi dan pembentukan matriks fitur

Kata	**Apex Legends**	**Counter-Strike 2**	**Valorant**
action	1	1	1
adventure	1	0	0
free	1	1	1
multi-player	1	1	1
pvp	1	1	1
co-op	1	1	0
competitive	0	1	1
anti-cheat	0	1	1

Penerapan `stop_words="english"` digunakan untuk menghilangkan kata-kata umum yang tidak memiliki makna penting dalam proses pemodelan. Setelah fitur terbentuk, sistem menghitung tingkat kemiripan antar game menggunakan metode *cosine similarity*. Hasil perhitungan ini berupa matriks kesamaan yang menunjukkan seberapa mirip satu game dengan game lainnya berdasarkan fitur kontennya.

Sebagai contoh, dilakukan perhitungan *cosine similarity* antara game *Apex Legends*,*Counter-Strike 2*, dan *Valorant* berdasarkan delapan fitur konten, yaitu *action*, *adventure*, *free*, *multi-player*, *pvp*, *co-op*, *competitive*, dan *anti-cheat*. Representasi vektor fitur untuk masing-masing game ditunjukkan sebagai berikut:

$$\vec{A} = [1,1,1,1,1,1,0,0] \quad \text{(Apex Legends)}$$

$$\vec{B} = [1,0,1,1,1,1,1,1] \quad \text{(Counter-Strike 2)}$$

$$\vec{C} = [1,0,1,1,1,0,1,1] \quad \text{(Valorant)}$$

Perhitungan *cosine similarity* dilakukan menggunakan persamaan berikut:

$$\text{Cosine Similarity} = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \times \|\vec{y}\|}$$

Apex Legends dan Counter-Strike 2

Nilai *dot product* diperoleh sebagai berikut:

$$\vec{A} \cdot \vec{B} = 5$$

Panjang masing-masing vektor adalah:

$$\|\vec{A}\| = \sqrt{6}, \quad \|\vec{B}\| = \sqrt{7}$$

Sehingga nilai *cosine similarity* adalah:

$$\text{Cosine}(A,B) = \frac{5}{\sqrt{6} \times \sqrt{7}} = \frac{5}{\sqrt{42}} \approx 0.772$$

Apex Legends dan Valorant

Nilai *dot product*:

$$\vec{A} \cdot \vec{C} = 4$$

Panjang vektor:

$$\|\vec{A}\| = \sqrt{6}, \quad \|\vec{C}\| = \sqrt{6}$$

Nilai *cosine similarity*:

$$\text{Cosine}(A,C) = \frac{4}{6} \approx 0.667$$

Counter-Strike 2 dan Valorant

Nilai *dot product*:

$$\vec{B} \cdot \vec{C} = 6$$

Panjang vektor:

$$\|\vec{B}\| = \sqrt{7}, \quad \|\vec{C}\| = \sqrt{6}$$

Nilai *cosine similarity*:

$$\text{Cosine}(B,C) = \frac{6}{\sqrt{7} \times \sqrt{6}} = \frac{6}{\sqrt{42}} \approx 0.926$$

Berdasarkan hasil perhitungan tersebut, pasangan game *Counter-Strike 2* dan *Valorant* memiliki tingkat kemiripan tertinggi, sedangkan *Apex Legends* dan *Valorant* memiliki tingkat kemiripan yang lebih rendah dibandingkan pasangan lainnya.

Matriks pada Tabel 3.4 menunjukkan tingkat kemiripan antar game yang dihitung menggunakan metode cosine similarity. Setiap nilai merepresentasikan seberapa mirip dua game berdasarkan fitur kontennya, di mana nilai 1 menunjukkan kemiripan sempurna dan nilai mendekati 0 menunjukkan kemiripan rendah. Nilai pada diagonal selalu 1 karena setiap game dibandingkan dengan dirinya sendiri. Dari matriks tersebut, terlihat bahwa beberapa game memiliki tingkat kemiripan yang tinggi, sehingga dapat digunakan sebagai dasar untuk memberikan rekomendasi pada tahap berikutnya.

Tabel 3.4. Contoh matriks cosine similarity antar game

Game	Apex Legends	Counter-Strike 2	Valorant
Apex Legends	1.00	0.77	0.67
Counter-Strike 2	0.77	1.00	0.93
Valorant	0.67	0.93	1.00

Potongan Kode 3.3 mendefinisikan fungsi `fetchRecommendedGamesCF()` yang bertugas menghasilkan rekomendasi berbasis konten. Fungsi ini pertama-tama memeriksa apakah nama game yang dicari tersedia dalam dataset. Jika tidak ditemukan, fungsi akan mengembalikan pesan kesalahan. Jika tersedia, sistem menentukan indeks game tersebut dan mengambil nilai *cosine similarity* terhadap game lain. Pada tahap selanjutnya, diterapkan proses penyaringan nilai kemiripan menggunakan baris kode `sim_scores = [s for s in sim_scores if s[1] >= 0.5]`. Nilai ambang batas 0.5 digunakan sebagai nilai dasar (*baseline*) untuk membedakan antara game yang memiliki kemiripan bermakna dan game dengan kemiripan rendah. Pemilihan nilai ini bertujuan untuk menjaga keseimbangan antara relevansi rekomendasi dan jumlah item yang direkomendasikan. Secara konseptual, nilai cosine similarity berada pada rentang 0 hingga 1, di mana nilai yang semakin mendekati 1 menunjukkan tingkat kemiripan yang semakin tinggi. Nilai 0.5 dipilih karena merepresentasikan tingkat kemiripan menengah, yang menunjukkan bahwa dua game memiliki cukup banyak kesamaan atribut konten tanpa harus identik. Dengan menggunakan ambang batas ini, sistem dapat menghindari rekomendasi game yang hanya memiliki kesamaan minimal, sekaligus tetap mempertahankan variasi dalam hasil rekomendasi. Game dengan nilai cosine similarity di bawah 0.5 dianggap memiliki kesamaan konten yang kurang signifikan, sehingga tidak disertakan dalam proses rekomendasi. Setelah proses penyaringan, daftar game diurutkan berdasarkan nilai kemiripan tertinggi, dan sepuluh game

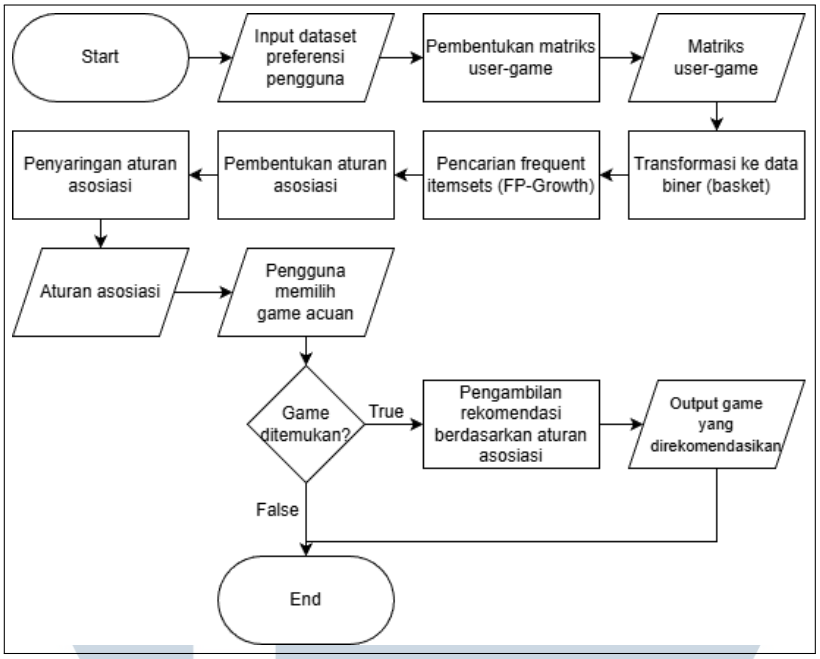
teratas dipilih sebagai hasil rekomendasi. Output akhir berupa daftar nama game yang memiliki tingkat kemiripan konten paling relevan dengan game yang dicari oleh pengguna

```
def fetchRecommendedGamesCF(game: str):
    if game not in df["name"].values:
        return {"error": f"'{game}' not found in the dataset."}

    idx = df[df["name"] == game].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))

    sim_scores = [s for s in sim_scores if s[1] >= 0.5]

    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=
    True)[1:6]

    recommended_indices = [i[0] for i in sim_scores]
    recommended_games = df.iloc[recommended_indices]["name"].
    tolist()
    return recommended_games
```

Kode 3.3: Fungsi pengambilan rekomendasi game berdasarkan cosine similarity

3.5 Penerapan Association Rule

Sebelum memasuki proses implementasi, metode *Association Rule Mining* digunakan dalam sistem rekomendasi untuk menemukan pola hubungan antar game berdasarkan perilaku pengguna. Berbeda dengan *Content-Based Filtering* yang berfokus pada kemiripan konten, metode ini menganalisis kecenderungan pengguna dalam memainkan beberapa game secara bersamaan. Dengan memanfaatkan pola ko-occurence tersebut, sistem mampu mengidentifikasi game yang memiliki keterkaitan kuat berdasarkan data historis interaksi pengguna, sehingga rekomendasi yang dihasilkan lebih mencerminkan kebiasaan alami pengguna dalam memilih game. Penerapan metode Association Rule dibagi menjadi beberapa tahapan yang dapat dilihat pada pada Gambar 3.5.



Gambar 3.5. Flowchart penerapan Association Rule

Flowchart tersebut menggambarkan alur penerapan metode *Association Rule Mining* dalam sistem rekomendasi video game. Proses dimulai dengan memasukkan data preferensi pengguna yang merepresentasikan interaksi pengguna terhadap game. Data tersebut kemudian diolah dengan membentuk *user-game matrix* untuk menggambarkan hubungan antara pengguna dan game.

Selanjutnya, data ditransformasikan ke dalam bentuk data biner atau *basket* sebagai persiapan untuk proses pencarian *frequent itemsets*. Berdasarkan itemsets yang terbentuk, sistem melakukan pembentukan aturan asosiasi yang menggambarkan keterkaitan antar game. Aturan asosiasi yang dihasilkan kemudian diseleksi untuk memperoleh aturan yang relevan sesuai dengan kriteria yang telah ditentukan.

Aturan yang telah diseleksi selanjutnya dipetakan ke dalam struktur rekomendasi. Setelah pengguna menentukan game acuan, sistem mengambil rekomendasi berdasarkan aturan asosiasi yang tersedia. Tahap akhir dari proses ini menghasilkan output berupa daftar game rekomendasi yang ditampilkan kepada pengguna, kemudian sistem mengakhiri proses.

Pada potongan Kode 3.4 digunakan untuk membentuk *user–game matrix* dari data riwayat permainan pengguna. Proses ini dilakukan dengan mengelompokkan data berdasarkan `user_id` dan `game_name`, kemudian menghitung jumlah kemunculan setiap game yang dimainkan oleh masing-masing

pengguna. Fungsi `unstack()` digunakan untuk mengubah data yang semula berbentuk deretan baris menjadi format matriks dengan kolom merepresentasikan nama game dan baris merepresentasikan pengguna. Hasil akhirnya adalah sebuah matriks yang menunjukkan frekuensi setiap game dimainkan oleh tiap pengguna, yang nantinya digunakan sebagai dasar dalam penerapan metode Association Rule.

```
user_game_matrix = (
    df_user.groupby(["user_id", "game_name"])["game_name"]
    .count()
    .unstack()
    .reset_index()
)
```

Kode 3.4: Kode pembentukan user-game matrix dari riwayat permainan pengguna

Tabel 3.5 menampilkan contoh data awal berupa riwayat permainan pengguna. Setiap baris merepresentasikan satu interaksi, yaitu seorang pengguna memainkan sebuah game tertentu. Misalnya, pengguna U1 memainkan dua game yaitu *Apex Legends* dan *Valorant*, sedangkan pengguna U2 memainkan *Apex Legends* serta dua kali memainkan *Counter-Strike 2*. Format data seperti ini masih bersifat vertikal dan belum dapat digunakan langsung dalam proses pencarian pola asosiasi, sehingga perlu diubah menjadi struktur matriks yang merepresentasikan hubungan pengguna terhadap seluruh game.

Tabel 3.5. Contoh data awal riwayat permainan pengguna

user_id	**game_name**
U1	Apex Legends
U1	Valorant
U2	Apex Legends
U2	Counter-Strike 2
U3	Valorant

Tabel 3.6 menunjukkan hasil transformasi data awal menjadi *user–game matrix* menggunakan operasi `groupby` dan `unstack`. Pada matriks ini, setiap baris mewakili seorang pengguna dan setiap kolom mewakili sebuah game, dengan nilai berupa jumlah kemunculan pengguna memainkan game tersebut. Sebagai contoh, pengguna U2 memiliki nilai 2 pada kolom *CS2*, yang berarti ia memainkan game tersebut sebanyak dua kali. Struktur matriks ini memudahkan proses selanjutnya dalam pembentukan pola asosiasi, karena memungkinkan sistem menganalisis game mana saja yang cenderung dimainkan bersama oleh pengguna.

Tabel 3.6. Hasil pembentukan user–game matrix

user_id	Apex Legends	Valorant	CS2
U1	1	1	0
U2	1	0	1
U3	0	1	0

Pada Kode 3.5, kode tersebut digunakan untuk memastikan bahwa seluruh kolom pada *user–game matrix* memiliki tipe data numerik. Proses konversi dilakukan menggunakan fungsi `pd.to_numeric()` dengan opsi `errors="coerce"` sehingga nilai yang tidak dapat dikonversi akan diganti menjadi `NaN`. Selanjutnya, nilai `NaN` tersebut diisi dengan angka nol menggunakan `fillna(0)`. Terakhir, kolom `user_id` dijadikan indeks sehingga matriks memiliki struktur yang sesuai untuk proses analisis lebih lanjut pada metode Association Rule.

```
for col in user_game_matrix.columns[1:]:
    user_game_matrix[col] = pd.to_numeric(user_game_matrix[col],
    errors="coerce")

user_game_matrix = user_game_matrix.fillna(0).set_index("user_id")
```

Kode 3.5: Kode normalisasi tipe data dan penyesuaian struktur user-game matrix

Kode 3.6 berfungsi untuk mengubah *user–game matrix* menjadi bentuk data biner yang menunjukkan apakah seorang pengguna memainkan suatu permainan atau tidak. Proses ini dilakukan dengan membandingkan setiap nilai pada matriks dengan angka nol (`> 0`). Jika nilai lebih besar dari nol maka akan dikonversi menjadi `True`, sedangkan nilai nol akan menjadi `False`. Struktur biner ini diperlukan sebagai input untuk algoritma FP-Growth karena metode tersebut bekerja berdasarkan representasi *transaction–item*, bukan jumlah frekuensi.

```
basket = user_game_matrix > 0
```

Kode 3.6: Kode transformasi user-game matrix menjadi data basket biner

Tabel 3.7 menunjukkan hasil konversi *user–game matrix* menjadi bentuk biner yang digunakan pada proses *market basket analysis*. Setiap nilai pada tabel merepresentasikan apakah seorang pengguna pernah memainkan suatu game. Nilai `True` berarti pengguna memainkan game tersebut setidaknya satu kali, sedangkan `False` menunjukkan bahwa game tersebut tidak pernah dimainkan oleh pengguna. Representasi biner ini diperlukan sebagai masukan utama bagi algoritma FP-Growth untuk menemukan pola keterkaitan antar item (game) berdasarkan riwayat permainan pengguna.

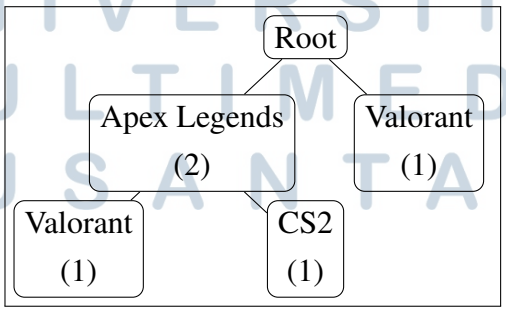
Tabel 3.7. Contoh matriks basket setelah konversi menjadi nilai biner

user_id	Apex Legends	Valorant	CS2
U1	True	True	False
U2	True	False	True
U3	False	True	False

Sebelum proses pencarian *frequent itemsets* dilakukan, algoritma FP-Growth secara konseptual membangun struktur data yang disebut *Frequent Pattern Tree* (FP-Tree). FP-Tree merupakan representasi terkompresi dari data transaksi yang menyimpan item-item yang sering muncul beserta frekuensinya dalam bentuk struktur pohon.

Pada penelitian ini, pembentukan FP-Tree tidak dituliskan secara eksplisit dalam kode program karena telah ditangani secara internal oleh fungsi `fpgrowth()` dari pustaka yang digunakan. Namun demikian, secara konseptual proses ini tetap berlangsung, dimulai dengan penyusunan transaksi pengguna berdasarkan urutan frekuensi item, kemudian membangun simpul pohon yang mewakili kemunculan game dalam setiap transaksi.

Gambar 3.6 menunjukkan ilustrasi sederhana struktur *FP-Tree* yang dibentuk berdasarkan data basket pengguna. Node akar (*Root*) merepresentasikan awal transaksi. Game *Apex Legends* muncul sebagai node utama karena memiliki frekuensi tertinggi dan menjadi prefix pada beberapa transaksi. Cabang *Apex Legends* $\rightarrow$ *Valorant* terbentuk dari transaksi pengguna U1, sedangkan cabang *Apex Legends* $\rightarrow$ *CS2* berasal dari transaksi pengguna U2. Sementara itu, transaksi pengguna U3 yang hanya memainkan *Valorant* membentuk cabang terpisah langsung dari node akar. Struktur FP-Tree ini memungkinkan algoritma FP-Growth untuk menelusuri pola keterkaitan antar game secara efisien tanpa perlu membangkitkan kandidat itemset secara eksplisit.



Gambar 3.6. Ilustrasi sederhana FP-Tree berdasarkan data basket pengguna

Pada Kode 3.7, diterapkan algoritma *FP-Growth* untuk mencari *frequent itemsets*, yaitu kombinasi game yang sering dimainkan secara bersamaan oleh pengguna. Algoritma ini dipilih karena mampu mengekstraksi pola asosiasi secara efisien tanpa melakukan pembangkitan kandidat itemset, sehingga sesuai untuk dataset dengan jumlah item yang besar.

```
freq_items = fpgrowth(basket, min_support=0.033, use_colnames=True
    )
```

Kode 3.7: Kode pencarian frequent itemsets menggunakan algoritma FP-Growth

Parameter `min_support=0.033` digunakan sebagai ambang minimum kemunculan suatu kombinasi item, yang berarti suatu *itemset* harus muncul setidaknya pada 3,3% dari seluruh transaksi agar dianggap *frequent*. Dataset yang digunakan dalam penelitian ini terdiri dari sekitar 90 transaksi dengan 258 item unik, sehingga memiliki karakteristik data yang jarang (*sparse*). Pada kondisi tersebut, penggunaan nilai *minimum support* yang terlalu tinggi berpotensi menghilangkan pola asosiasi yang bermakna karena sebagian besar kombinasi item hanya muncul pada sebagian kecil transaksi.

Nilai `min_support=0.033` dipilih agar ambang kemunculan *itemset* dapat dibulatkan secara praktis menjadi minimal 3 transaksi, sehingga setiap pola yang dihasilkan tetap memiliki makna statistik dan tidak hanya muncul secara kebetulan. Selain itu, penelitian oleh Ogedengbe et al.[15] menunjukkan bahwa pada dataset dunia nyata dengan jumlah item yang lebih sedikit, nilai *minimum support* optimal berada pada kisaran 0.057. Mengingat jumlah item unik pada penelitian ini jauh lebih besar, penggunaan ambang yang lebih rendah, yaitu sekitar 3%, dinilai lebih representatif untuk menangkap pola asosiasi yang relevan tanpa menghasilkan aturan yang berlebihan.

Penggunaan opsi `use_colnames=True` memastikan bahwa hasil *frequent itemsets* ditampilkan dalam bentuk nama game, bukan indeks numerik, sehingga memudahkan interpretasi dan analisis lanjutan. Himpunan *frequent itemsets* yang diperoleh selanjutnya digunakan sebagai dasar dalam pembentukan aturan asosiasi pada tahap berikutnya.

Potongan Kode 3.8 menggunakan fungsi `association_rules()` dari pustaka *mlxtend* untuk membentuk aturan asosiasi berdasarkan *frequent itemset* yang dihasilkan oleh algoritma FP-Growth. Pada proses ini digunakan parameter `metric="lift"` dengan nilai `min_threshold=1`, yang bertujuan untuk menyaring aturan asosiasi yang memiliki hubungan positif dan signifikan antara *antecedent* dan *consequent*.

```
all_ar_rules = association_rules(
    freq_items, metric="conf", min_threshold=0.3
)
```

Kode 3.8: Kode pembentukan aturan asosiasi berdasarkan metrik lift

Penentuan nilai minimum confidence dalam penelitian ini dilakukan melalui pengujian beberapa nilai confidence yang berbeda untuk mengamati perubahan jumlah aturan asosiasi, kualitas pola hubungan antar item, serta relevansi aturan yang dihasilkan. Nilai confidence tidak dapat ditetapkan terlalu rendah karena berpotensi menghasilkan banyak aturan asosiasi yang lemah dan bersifat noise, sehingga menurunkan kualitas rekomendasi. Sebaliknya, nilai confidence yang terlalu tinggi dapat mengeliminasi aturan yang masih signifikan dan menyebabkan jumlah pola yang ditemukan menjadi sangat terbatas. Hal ini sejalan dengan penelitian sebelumnya yang menyatakan bahwa perubahan nilai minimum support dan minimum confidence berpengaruh terhadap jumlah aturan yang dihasilkan, struktur pola asosiasi, serta kualitas dan peringkat rekomendasi solusi yang diperoleh [16].

Berdasarkan hasil analisis tersebut, nilai minimum confidence sebesar 0.3 dipilih sebagai threshold yang paling sesuai. Nilai ini dipilih karena mampu memberikan keseimbangan antara jumlah aturan asosiasi yang dihasilkan dan kekuatan hubungan antar item. Selain itu, pada nilai confidence di bawah 0.3, nilai lift relatif tidak mengalami perubahan yang signifikan sehingga penurunan threshold hanya meningkatkan jumlah aturan tanpa peningkatan kualitas pola yang berarti. Sementara itu, pada nilai confidence di atas 0.3, jumlah aturan yang dihasilkan menurun secara signifikan sehingga sebagian pola yang masih relevan ikut tereliminasi. Oleh karena itu, nilai confidence sebesar 0.3 dipandang sebagai nilai yang optimal untuk menghasilkan aturan yang stabil, representatif, dan relevan dalam proses rekomendasi.

Fungsi `association_rules()` menghasilkan aturan asosiasi dalam bentuk pasangan *antecedent* dan *consequent*, beserta nilai metrik pendukung seperti *support*, *confidence*, dan *lift*. Aturan asosiasi yang memenuhi kriteria tersebut selanjutnya dimanfaatkan sebagai dasar dalam menghasilkan rekomendasi game berdasarkan pola keterkaitan item dan kesamaan perilaku pemain.

Tabel 3.8 menampilkan contoh aturan asosiasi yang dihasilkan dari algoritma FP-Growth berdasarkan data interaksi pengguna dengan tiga game, yaitu Apex Legends, Valorant, dan Counter-Strike 2 (CS2). Setiap baris menunjukkan hubungan antara antecedent (game yang sudah dimainkan pengguna)

dan consequent (game lain yang berasosiasi dengannya), disertai nilai support, confidence, dan lift sebagai indikator kekuatan asosiasi. Sebagai contoh, aturan Apex Legends → CS2 memiliki nilai lift sebesar 1,50, yang menunjukkan hubungan positif dan mengindikasikan bahwa pemain Apex Legends memiliki kecenderungan lebih tinggi untuk memainkan CS2 dibandingkan peluang acak. Sementara itu, aturan Apex Legends → Valorant memiliki nilai lift 0,75, yang berarti asosiasinya relatif lemah. Tabel ini secara keseluruhan menggambarkan pola ko-occurence antar game dalam riwayat pengguna, yang selanjutnya digunakan untuk membentuk rekomendasi berbasis Association Rule Mining.

Tabel 3.8. Contoh aturan asosiasi yang dihasilkan dari FP-Growth

Antecedent	**Consequent**	**Support**	**Confidence**	**Lift**
Apex Legends	Valorant	0.33	0.50	0.75
Valorant	Apex Legends	0.33	0.50	0.75
Apex Legends	CS2	0.33	0.50	1.50
CS2	Apex Legends	0.33	1	1.50

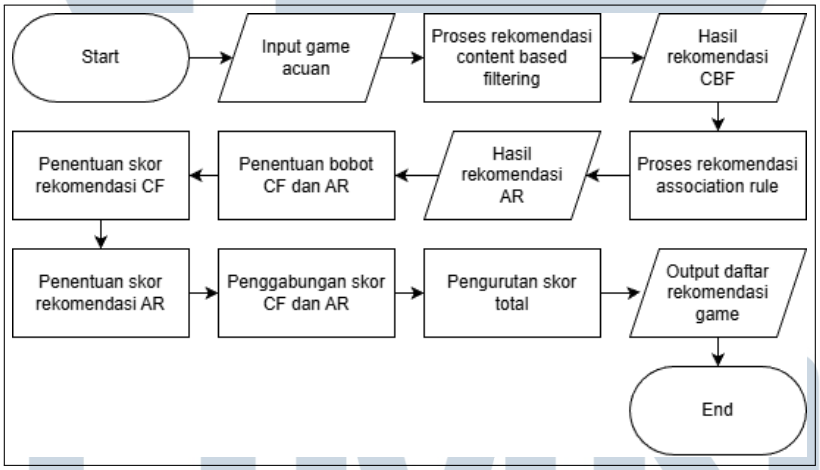
Potongan Kode pada Listing 3.9 merupakan fungsi yang digunakan untuk menghasilkan rekomendasi game berdasarkan model Association Rule Mining yang telah dibentuk sebelumnya. Fungsi ini menerima sebuah parameter berupa nama game yang dijadikan titik acuan. Pada tahap awal, fungsi melakukan validasi untuk memastikan bahwa game tersebut terdapat pada struktur pemetaan `ar_map`, yaitu struktur yang memuat daftar konsekuen hasil pembentukan aturan asosiasi. Apabila game tidak ditemukan dalam pemetaan, fungsi akan mengembalikan pesan bahwa tidak terdapat rekomendasi yang dapat diberikan. Sebaliknya, apabila game tersebut memiliki aturan asosiasi yang relevan, fungsi akan mengembalikan hingga lima game pertama yang muncul sebagai konsekuen dari aturan-aturan tersebut. Dengan demikian, fungsi ini berperan sebagai mekanisme pengambilan rekomendasi yang didasarkan pada pola keterkaitan antar game yang diperoleh dari proses analisis FP-Growth, sehingga memungkinkan sistem memberikan saran permainan yang berpotensi diminati oleh pengguna.

```
1 def fetchRecommendedGamesAR(game: str):
2     if game not in ar_map:
3         return {"message": f"No AR recommendations found for '{
    game}'."}
4     return ar_map[game][:5]
```

Kode 3.9: Fungsi pengambilan rekomendasi game berbasis aturan asosiasi

3.6 Penerapan Metode Hybrid Content-Based Filtering dan Association Rule

Pada tahap ini, sistem rekomendasi menggabungkan dua pendekatan utama, yaitu Content-Based Filtering (CF) dan Association Rule (AR), untuk menghasilkan rekomendasi permainan yang lebih akurat dan relevan. Metode hybrid ini dirancang agar mampu memanfaatkan kekuatan masing-masing pendekatan, di mana CF bekerja berdasarkan kemiripan karakteristik antar permainan, sedangkan AR mengidentifikasi pola keterkaitan antar permainan berdasarkan perilaku pengguna sebelumnya. Kombinasi keduanya diharapkan dapat menghasilkan rekomendasi yang lebih komprehensif dan mampu mengatasi kelemahan dari masing-masing metode ketika digunakan secara terpisah. Proses penerapan penggabungan metode *Content Based Filtering* dan *Association Rule* dapat dilihat pada gambar Gambar 3.7.



Gambar 3.7. Flowchart penerapan metode hybrid

Alur penerapan metode *Hybrid* pada sistem rekomendasi video game. Proses diawali dengan input game acuan yang dipilih oleh pengguna. Berdasarkan game acuan tersebut, sistem secara paralel menghasilkan rekomendasi menggunakan dua pendekatan, yaitu *Content-Based Filtering* (CBF) dan *Association Rule* (AR).

Hasil rekomendasi dari masing-masing metode kemudian diperoleh dalam bentuk daftar rekomendasi CBF dan rekomendasi AR. Selanjutnya, dilakukan penentuan bobot untuk masing-masing metode sebagai dasar dalam proses penggabungan hasil rekomendasi. Berdasarkan bobot tersebut, sistem melakukan perhitungan skor rekomendasi untuk setiap game.

Tahap berikutnya adalah penggabungan skor dari kedua metode untuk

menghasilkan skor akhir. Skor yang telah digabungkan kemudian diurutkan berdasarkan nilai tertinggi hingga terendah. Pada tahap akhir, sistem menghasilkan output berupa daftar rekomendasi game yang paling relevan bagi pengguna, kemudian proses diakhiri.

Proses awal metode hybrid diawali dengan memperoleh hasil rekomendasi dengan menggabungkan pendekatan Content-Based Filtering (CF) dan Association Rule (AR). yang dapat dilihat pada kode 3.10. Pertama, sistem memanggil fungsi `fetchRecommendedGamesCF(game)` untuk menghasilkan daftar rekomendasi berbasis kemiripan konten. Jika fungsi tersebut mengembalikan objek bertipe `dict`, maka hal tersebut menandakan bahwa terjadi kesalahan atau data yang diminta tidak tersedia, sehingga proses dihentikan dan pesan kesalahan dikembalikan kepada pengguna. Selanjutnya, sistem memanggil fungsi `fetchRecommendedGamesAR(game)` untuk mengambil rekomendasi berdasarkan aturan asosiasi. Apabila hasil yang diperoleh berupa `dict` atau bernilai `None`, maka hasil AR dianggap tidak valid dan digantikan dengan daftar kosong. Mekanisme ini memastikan bahwa proses hybrid tetap dapat dilanjutkan meskipun salah satu metode tidak menghasilkan rekomendasi yang valid.

```
cf_results = fetchRecommendedGamesCF(game)
if isinstance(cf_results, dict):
    return cf_results

ar_results = fetchRecommendedGamesAR(game)
if isinstance(ar_results, dict) or ar_results is None:
    ar_results = []
```

Kode 3.10: Penerapan kode pengambilan rekomendasi awal pada metode hybrid

Kemudian sistem akan mendefinisikan fungsi `calculate_weighted`, yang bertujuan untuk menggabungkan rekomendasi dari metode Content-Based Filtering (CF) dan Association Rule (AR) melalui mekanisme pembobotan seperti pada kode 3.11. Fungsi ini menggunakan dua parameter, yaitu *cf_weight* dan *ar_weight*, yang merepresentasikan bobot kontribusi masing-masing metode dalam perhitungan skor akhir. Setiap item dalam daftar rekomendasi CF dan AR diberikan skor berdasarkan posisinya (*rank*); semakin tinggi posisinya, semakin besar skor yang diberikan. Skor tersebut dihitung menggunakan rumus $(11 - rank) \times weight$, kemudian dijumlahkan ke dalam variabel `score_dict`. Setelah semua skor digabungkan, sistem melakukan pengurutan berdasarkan nilai skor secara menurun, dan lima item dengan skor tertinggi dikembalikan sebagai hasil rekomendasi akhir. Pendekatan ini memastikan bahwa metode dengan bobot lebih tinggi memiliki pengaruh lebih

besar terhadap urutan rekomendasi yang dihasilkan.

```
def calculate_weighted(cf_weight, ar_weight):
    score_dict = {}

    for rank, item in enumerate(cf_results, start=1):
        score = (11 - rank) * cf_weight
        score_dict[item] = score_dict.get(item, 0) + score

    for rank, item in enumerate(ar_results, start=1):
        score = (11 - rank) * ar_weight
        score_dict[item] = score_dict.get(item, 0) + score

    combined_sorted = sorted(score_dict.items(), key=lambda x: x
    [1], reverse=True)
    return [item[0] for item in combined_sorted][:5]
```

Kode 3.11: Penerapan kode perhitungan dan penggabungan skor pada metode hybrid

Berikut merupakan contoh proses perhitungan skor gabungan antara metode Content-Based Filtering (CF) dan Association Rule (AR) menggunakan pendekatan pembobotan. Pada contoh ini digunakan lima rekomendasi pertama dari masing-masing metode.

Daftar rekomendasi CF:

1. Apex Legends
2. Valorant
3. CS2
4. Overwatch 2
5. Battlefield V

Daftar rekomendasi AR:

1. Dota 2
2. PUBG
3. Apex Legends
4. Warframe
5. Rainbow Six Siege

Bobot yang digunakan:

$$cf\_weight = 0.5, \qquad ar\_weight = 0.5$$

1. Perhitungan Skor Content-Based Filtering

Pada metode *Content-Based Filtering*, sistem menghasilkan daftar rekomendasi game berdasarkan tingkat kemiripan atribut antara game yang pernah dimainkan pengguna dengan game lainnya. Hasil rekomendasi tersebut kemudian diurutkan berdasarkan tingkat relevansi. Untuk memberikan nilai kuantitatif pada setiap rekomendasi, dilakukan perhitungan skor dengan mempertimbangkan posisi peringkat game dan bobot metode CBF yang telah ditentukan. Perhitungan skor ini bertujuan untuk merepresentasikan kontribusi metode CBF dalam sistem rekomendasi hybrid. Perhitungan skor ini dilakukan dengan rumus:

$$score = (11 - rank) \times cf\_weight$$

Pada Tabel 3.9 menunjukkan hasil perhitungan skor rekomendasi menggunakan metode *Content-Based Filtering*. Skor dihitung dengan mengalikan bobot CF sebesar 0.5 dengan nilai peringkat terbalik $(11 - rank)$, sehingga item dengan peringkat lebih tinggi memperoleh skor yang lebih besar. Hasil ini mencerminkan tingkat kesesuaian game yang direkomendasikan berdasarkan kemiripan fitur dengan game yang sebelumnya disukai pengguna.

Tabel 3.9. Perhitungan skor rekomendasi CF

Rank	Game	Perhitungan	Skor
1	Apex Legends	$(11-1) \times 0.5$	5.0
2	Valorant	$(11-2) \times 0.5$	4.5
3	CS2	$(11-3) \times 0.5$	4.0
4	Overwatch 2	$(11-4) \times 0.5$	3.5
5	Battlefield V	$(11-5) \times 0.5$	3.0

2. Perhitungan Skor Association Rule

Pada metode *Association Rule*, sistem menghasilkan rekomendasi berdasarkan pola keterkaitan antar game yang sering muncul secara bersamaan dalam histori pengguna. Game yang memiliki hubungan asosiasi yang kuat akan

memperoleh peringkat lebih tinggi. Selanjutnya, setiap rekomendasi diberikan skor numerik berdasarkan peringkatnya dan dikalikan dengan bobot metode Association Rule. Proses ini bertujuan untuk mengukur kontribusi metode AR dalam membentuk rekomendasi akhir pada sistem hybrid. Rumus perhitungan skor:

$$score = (11 - rank) \times ar\_weight$$

Tabel 3.10 menyajikan hasil perhitungan skor rekomendasi menggunakan metode *Association Rule*. Skor dihitung menggunakan rumus yang sama, dengan bobot AR sebesar 0.5. Metode ini menilai relevansi game berdasarkan pola asosiasi yang sering muncul pada data histori kepemilikan atau interaksi pengguna, sehingga mampu menangkap hubungan implisit antar game.

Tabel 3.10. Perhitungan skor rekomendasi AR

Rank	Game	Perhitungan	Skor
1	Dota 2	$(11-1) \times 0.5$	5.0
2	PUBG	$(11-2) \times 0.5$	4.5
3	Apex Legends	$(11-3) \times 0.5$	4.0
4	Warframe	$(11-4) \times 0.5$	3.5
5	Rainbow Six Siege	$(11-5) \times 0.5$	3.0

3. Penggabungan Skor CF dan AR

Setelah skor dari masing-masing metode diperoleh, tahap selanjutnya adalah menggabungkan skor dari *Content-Based Filtering* dan *Association Rule*. Penggabungan ini dilakukan dengan menjumlahkan skor dari kedua metode untuk setiap game yang direkomendasikan. Hasil penggabungan skor tersebut kemudian digunakan sebagai dasar dalam menentukan peringkat akhir rekomendasi.

Tabel 3.11 penggabungan skor menunjukkan hasil integrasi antara skor *Content-Based Filtering* dan *Association Rule*. Jika suatu game hanya muncul pada salah satu metode, maka skor dari metode lainnya dianggap nol. Total skor diperoleh dengan menjumlahkan skor CF dan skor AR, sehingga game yang direkomendasikan oleh kedua metode akan memiliki nilai total yang lebih tinggi.

Tabel 3.11. Penggabungan skor CF dan AR

Game	Skor CF	Skor AR	Total Skor
Apex Legends	5.0	4.0	9.0
Valorant	4.5	–	4.5
CS2	4.0	–	4.0
Overwatch 2	3.5	–	3.5
Battlefield V	3.0	–	3.0
Dota 2	–	5.0	5.0
PUBG	–	4.5	4.5
Warframe	–	3.5	3.5
Rainbow Six Siege	–	3.0	3.0

4. Hasil Akhir Rekomendasi Hybrid

Tahap ini menyajikan hasil akhir dari sistem rekomendasi hybrid yang diperoleh berdasarkan penggabungan skor *Content-Based Filtering* dan *Association Rule*. Peringkat rekomendasi ditentukan berdasarkan skor total tertinggi yang merepresentasikan tingkat relevansi game terhadap preferensi pengguna. Daftar berikut menunjukkan urutan game yang direkomendasikan sebagai hasil akhir dari proses pengolahan data dan perhitungan skor.
Peringkat akhir berdasarkan skor total tertinggi:

1. Apex Legends (9.0)
2. Dota 2 (5.0)
3. Valorant (4.5)
4. PUBG (4.5)
5. CS2 (4.0)

Berdasarkan hasil perhitungan dan pengurutan skor total, diperoleh lima game dengan tingkat relevansi tertinggi sebagai rekomendasi akhir, yaitu Apex Legends, Dota 2, Valorant, PUBG, dan CS2. Game Apex Legends menempati peringkat teratas karena memperoleh kontribusi skor dari kedua metode, sedangkan game lainnya direkomendasikan berdasarkan dominasi skor dari salah satu metode. Hasil ini menunjukkan bahwa pendekatan hybrid mampu mengombinasikan

keunggulan masing-masing metode dalam menghasilkan rekomendasi yang lebih komprehensif.

Pada kode 3.12 menunjukkan penerapan tiga skenario pembobotan yang berbeda dalam proses penggabungan rekomendasi. Skenario pertama menggunakan kombinasi bobot 0,25:0,75, di mana metode Association Rule (AR) memiliki pengaruh yang lebih dominan dibandingkan dengan Content-Based Filtering (CF). Skenario kedua menggunakan bobot seimbang 0,50:0,50, sehingga kedua metode memberikan kontribusi yang sama besar terhadap hasil rekomendasi. Sementara itu, skenario ketiga menggunakan kombinasi bobot 0,75:0,25, yang memberikan porsi dominan kepada metode CF. Variasi pembobotan tersebut dilakukan untuk mengevaluasi sejauh mana kontribusi relatif antara metode CF dan AR memengaruhi kualitas rekomendasi yang dihasilkan.

```
result_25_75 = calculate_weighted(0.25, 0.75)
result_50_50 = calculate_weighted(0.50, 0.50)
result_75_25 = calculate_weighted(0.75, 0.25)
```

Kode 3.12: Penerapan kode pembobotan metode pada metode hybrid

Melalui perbandingan hasil rekomendasi dari ketiga skenario tersebut, sistem dapat mengidentifikasi konfigurasi bobot yang menghasilkan rekomendasi paling relevan dan konsisten. Proses ini penting dalam penelitian karena pemilihan bobot yang tepat berperan langsung terhadap performa metode hybrid. Dengan demikian, pengujian ini memungkinkan peneliti untuk menentukan bobot optimal yang mampu memaksimalkan akurasi rekomendasi berdasarkan karakteristik data serta pola preferensi pengguna.

3.7 Implementasi Sistem

Bab ini menjelaskan tahapan implementasi sistem rekomendasi game yang dikembangkan pada penelitian ini. Sistem terdiri dari dua komponen utama, yaitu API berbasis Python yang menangani seluruh proses komputasi rekomendasi serta website berbasis PHP yang berfungsi sebagai antarmuka pengguna. Kedua komponen tersebut saling terhubung untuk menghasilkan rekomendasi game yang ditampilkan secara interaktif melalui halaman web.

3.7.1 Implementasi API Sistem Rekomendasi

API dibangun menggunakan bahasa pemrograman Python dan berperan sebagai pusat pengolahan data serta perhitungan rekomendasi. Seluruh proses komputasi, mulai dari Content-Based Filtering (CF), Association Rule (AR), hingga metode hybrid CF–AR, dilakukan pada sisi API sehingga website tidak melakukan proses komputasi berat.

API mengimplementasikan beberapa fungsi utama sebagai berikut:

1. **Pemuatan Data**
   API memuat dataset game yang berisi informasi deskripsi, genre, dan relasi antar game. Dataset ini menjadi dasar dalam perhitungan kemiripan serta pembentukan aturan asosiasi.

2. **Content-Based Filtering (CF)**
   API menghitung kemiripan antar game berdasarkan fitur konten, seperti genre atau deskripsi. Proses ini mencakup ekstraksi fitur, transformasi teks, dan perhitungan kemiripan menggunakan metode tertentu.

3. **Association Rule (AR)**
   API membangkitkan aturan asosiasi menggunakan pola hubungan antar game yang berasal dari data historis. Aturan tersebut digunakan untuk memberikan rekomendasi berdasarkan *co-occurrence* antar game.

4. **Hybrid CF–AR**
   API menggabungkan hasil rekomendasi CF dan AR menggunakan pembobotan tertentu. Tiga konfigurasi bobot yang dihasilkan adalah:

   $$(cf\_weight, ar\_weight) = (0.25, 0.75),\ (0.50, 0.50),\ (0.75, 0.25)$$

   Pembobotan ini bertujuan untuk mengetahui kombinasi yang paling efektif dalam menghasilkan rekomendasi.

5. **Penyediaan Endpoint API**
   API menyediakan endpoint untuk mengambil daftar game, detail game, serta rekomendasi AR, CF, dan hybrid. Setiap endpoint mengembalikan data dalam format JSON sehingga dapat diproses oleh website.

3.7.2 Implementasi Website Berbasis PHP

Website dikembangkan menggunakan PHP dan berfungsi sebagai antarmuka pengguna untuk menampilkan daftar game serta hasil rekomendasi. Website tidak melakukan komputasi apa pun, melainkan hanya mengambil data dari API dan menyajikannya kepada pengguna.

Website terdiri dari beberapa halaman utama, yaitu:

A Halaman Utama

Halaman utama menampilkan daftar game yang dapat dijelajahi oleh pengguna. Fitur yang tersedia meliputi:

- **Daftar game** dalam bentuk tampilan grid atau kartu.
- **Search bar** untuk mencari game berdasarkan kata kunci.
- **Filter genre** yang memungkinkan pengguna memfilter daftar game sesuai kategori yang diinginkan.

Ketika pengguna melakukan pencarian atau filter, website mengirimkan permintaan ke API dan menampilkan hasilnya secara dinamis.

B Halaman Detail Game

Ketika pengguna memilih sebuah game, website menampilkan halaman detail yang berisi:

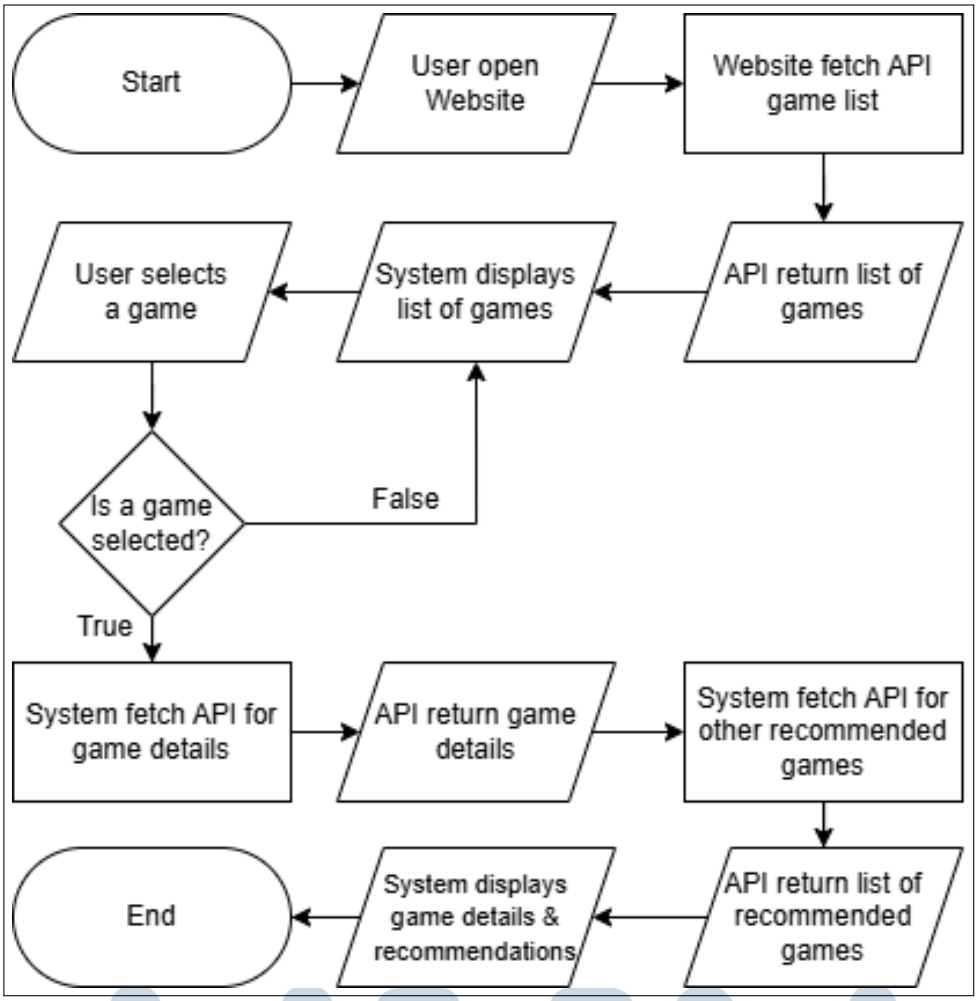
- Gambar, deskripsi singkat, dan tautan menuju halaman resmi game.
- Lima jenis hasil rekomendasi yang disajikan dalam bentuk *tab*, yaitu:
    1. Rekomendasi Association Rule (AR)
    2. Rekomendasi Content-Based Filtering (CF)
    3. Hybrid CF–AR (bobot 25% : 75%)
    4. Hybrid CF–AR (bobot 50% : 50%)
    5. Hybrid CF–AR (bobot 75% : 25%)

Website mengambil hasil rekomendasi dari API menggunakan permintaan HTTP dan menampilkan daftar game rekomendasi pada setiap *tab*.

3.7.3 Alur Kerja Sistem

Pada tahap implementasi sistem, diperlukan pemahaman yang jelas mengenai alur kerja dari aplikasi yang dibangun. Alur kerja ini menggambarkan proses komunikasi antara pengguna, website berbasis PHP, dan API berbasis Python yang menangani seluruh proses perhitungan sistem rekomendasi. Penyajian alur kerja dalam bentuk flowchart bertujuan untuk memberikan gambaran menyeluruh mengenai bagaimana data diproses mulai dari pengguna mengakses aplikasi hingga sistem menghasilkan rekomendasi permainan yang relevan.

Flowchart pada Gambar 3.8 menunjukkan bahwa proses dimulai ketika pengguna membuka halaman utama website. Website kemudian mengirim permintaan (fetch) ke API untuk memperoleh daftar seluruh permainan yang tersedia. Setelah API mengembalikan data tersebut, sistem menampilkan daftar permainan pada halaman utama. Ketika pengguna memilih salah satu permainan, website kembali melakukan dua permintaan API secara paralel: permintaan pertama untuk mengambil detail permainan dan permintaan kedua untuk menghitung serta mengambil daftar rekomendasi permainan berdasarkan lima metode (Association Rule, Content-Based Filtering, serta tiga metode hybrid dengan variasi bobot). Seluruh data yang diterima dari API kemudian ditampilkan pada halaman detail permainan, sehingga pengguna dapat melihat informasi permainan serta rekomendasi yang telah dihitung oleh sistem.



Gambar 3.8. Flowchart alur kerja sistem