LAPORAN TUGAS BESAR

IF2110 Algoritma dan Pemrograman 2

CREDIT

Dipersiapkan oleh:

13524001	Samuelson D Tanuraharja
13524019	Kevin Wirya Valerian
13524023	Jonathan Kris Wicaksono
13524026	Made Branenda Jordhy
13524053	Muhammad Haris Putra Sulastianto

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Daftar Isi

1 Ringkasan

Pada Tugas Besar IF2110 Algoritma dan Pemrograman 2, kami diminta membangun Groddit, sebuah aplikasi forum sosial berbasis CLI yang memungkinkan pengguna melakukan autentikasi, membuat post, berkomentar, memberikan vote, mengikuti pengguna lain, hingga menyimpan dan memuat seluruh state aplikasi. Sistem ini dikembangkan dengan bahasa C dan memanfaatkan berbagai ADT yang telah dipelajari, mencakup ADT Sederhana, ADT List Berkait, Mesin Karakter dan Mesin Kata, Tree, dan Graph.

Secara garis besar, program terdiri dari beberapa fitur utama, yaitu

1. Manajemen pengguna dengan autentikasi
2. Post dan Subgroddit untuk membuat dan mengelola forum
3. Komentar dan Reply untuk memberikan reaksi pada hal-hal yang diunggah pada forum
4. Sistem *upvote* dan *downvote* yang memengaruhi karma pengguna
5. Relasi Sosial yang memungkinkan *follow* ataupun *unfollow* antarpengguna
6. Pemuatan dan penyimpanan seluruh konfigurasi program ke dalam file eksternal

, serta beberapa fitur tambahan apabila dikerjakan.

Laporan ini terdiri atas beberapa bagian. Pada Bab 1, laporan ini akan memberikan gambaran umum soal apa yang dikerjakan dalam tugas besar kali ini. Kemudian, pada Bab 2, laporan ini akan menjelaskan soal spesifikasi-spesifikasi fitur tambahan yang belum rinci dideskripsikan. Pada Bab 3, laporan berisi penjelasan mengenai ADT yang kelompok kami gunakan untuk mengatasi persoalan pada tugas besar ini. Pada Bab 4, program utama menjelaskan alur kerja keseluruhan sistem mulai dari inisialisasi, pemanggilan fitur-fitur utama, hingga program berhenti. Bab 5 menjelaskan algoritma-algoritma khusus yang dianggap unik atau penting dalam implementasi sistem. Setelah itu, Bab 6 berisi kumpulan data uji beserta penjelasan fitur yang diuji dan hasil yang diharapkan dari setiap pengujian. Pada Bab 7, bagian ini menjelaskan skenario pengujian untuk seluruh fitur yang ada, termasuk tujuan, langkah-langkah, data *input*, dan hasil yang diharapkan. Pada Bab 8 dan 9, laporan ini menjelaskan sola pembagian kerja dalam kelompok kami dan lampiran yang relevan, secara berturut-turut.

Secara keseluruhan, aplikasi Credit yang kami implementasikan berhasil memenuhi spesifikasi inti Tugas Besar. Implementasi yang dilakukan mampu memodelkan interaksi dalam platform diskusi dengan berbagai operasi dasar yang melibatkan struktur data yang diajarkan. Selain itu, seluruh modul terintegrasi dalam alur program utama yang konsisten dan dapat menjalankan fungsi sesuai kebutuhan spesifikasi dan menangani seluruh kasus. Hasil akhir menunjukkan bahwa Credit dapat berjalan sesuai tujuan, stabil, dan mampu memproses seluruh data secara tepat dan efisien.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Penjelasan Tambahan Fitur Wajib

2.1.1. Comment and Reply - Cascading Delete

Spesifikasi hanya menyebutkan bahwa penghapusan komentar dan post bersifat cascading, tetapi tidak menjelaskan mekanismenya secara detail. Dalam tugas besar groddit ini, kelompok kami menggunakan struktur komentar berupa tree dengan representasi first-child next-sibling sehingga proses penghapusan dilakukan secara rekursif dari node yang yang dihapus. Ketika sebuah komentar dihapus, seluruh reply di bawahnya (child dan sibling) akan ikut terhapus dalam satu operasi traversal. Lalu untuk penghapusan post, kami melakukan cascade delete yang lebih luas, jadi tidak hanya menghapus semua komentar dalam post tersebut, tetapi juga menghapus seluruh record voting yang terkait dengan post dan komentarnya untuk menjaga konsistensi data.

2.1.2. Detail Penomoran ID

Pada data terdapat format ID, namun tidak menjelaskan bagaimana pengelolaan ID berjalan. Kelompok kami memutuskan untuk menggunakan dua pendekatan yang berbeda. Pertama, untuk POST ID, kami menggunakan algoritma MEX (Minimum Excluded) untuk recycling ID yang terhapus. Jadi jika ada post e.g. P001, P003, maka post baru akan mendapat ID P002 jadinya mengisi gap yang ada.

2.1.3. Save & Load

Dokumen menyebutkan bahwa input dapat menggunakan tanda kutip atau tidak, tetapi tidak menjelaskan secara eksplisit apakah pada fitur Save tanda kutip diletakkan pada atribut mana saja dalam data yang diberikan. Setelah observasi data sample, kelompok kami mengambil keputusan bahwa hanya kolom yang berpotensi mengandung delimiter seperti title dan content yang bisa mengandung koma, newline, atau karakter spesial yang diberikan tanda kutip di awal dan di akhir. Jadi field lain seperti misal ID, timestamp, dan counter numerik disimpan tanpa quotes. Sistem kami juga menangani escaped quotes di dalam content.

2.2 Spesifikasi Fitur Bonus

2.2.1. Feed (Heap)

Fitur Feed di Groddit dirancang untuk menampilkan kumpulan post yang relevan bagi pengguna berdasarkan akun-akun yang ia follow. Untuk mengatur urutan tampilan, kami menggunakan struktur data Heap (khususnya Max Heap) dengan key berupa timestamp pembuatan post (created_at). Ketika pengguna memanggil perintah feed, sistem terlebih dahulu membangun daftar kandidat post dengan cara:

1. Mengidentifikasi user lain yang di-follow oleh pengguna saat ini melalui graph sosial.

2. Memindai seluruh post di struktur global
3. Memasukkan hanya post milik akun-akun yang di-follow ke dalam array kandidat.

Setiap kandidat kemudian dimasukkan ke dalam Max Heap, di mana key-nya adalah nilai waktu (timestamp) post. Pada mode LATEST, ekstraksi dari heap dilakukan secara langsung sehingga post dengan timestamp terbesar (paling baru) muncul lebih dulu. Pada mode NEWEST, hasil ekstraksi dibalik urutannya sehingga post yang lebih lama tampil terlebih dahulu. Parameter LIMIT (jika diberikan) diterapkan dengan membatasi jumlah elemen yang di-pop dari heap, sehingga proses dapat dihentikan tepat setelah sejumlah LIMIT post berhasil diambil, tanpa perlu memproses seluruh kandidat.

2.2.2. Trending Analysis

Fitur Trending Analysis bertujuan mengidentifikasi kata-kata yang paling sering muncul dalam ekosistem diskusi subgroddit berdasarkan judul dan isi post maupun komentar. Prosesnya diawali dengan tokenisasi seluruh teks menggunakan mesin kata yang sudah ada, sehingga setiap kata dapat diproses sebagai token yang terpisah.

Selanjutnya, untuk menjaga relevansi waktu, setiap post atau komentar yang dianalisis dicek timestamp-nya terhadap batas waktu tertentu (misalnya beberapa hari atau jam terakhir). Hanya konten yang berada dalam jendela waktu ini yang disertakan dalam perhitungan tren. Kata-kata yang lolos kemudian dimasukkan ke dalam struktur data penampung (list atau array dinamis) yang menyimpan pasangan (kata, frekuensi).

Untuk mengurangi noise, hanya token yang dianggap sebagai kata valid (berisi huruf alfabet) yang dihitung: simbol, angka murni, atau karakter non-alfabet dapat diabaikan. Setelah seluruh data terkumpul, struktur frekuensi ini diurutkan berdasarkan jumlah kemunculan sehingga menghasilkan daftar kata yang paling sering muncul pada periode yang dianalisis, yang kemudian dapat ditampilkan sebagai hasil trending.

2.2.3. Advanced Search (Trie)

Fitur Advanced Search memanfaatkan struktur data Trie untuk mendukung pencarian berbasis prefix secara efisien. Dalam implementasinya, terdapat beberapa Trie terpisah yang masing-masing menyimpan daftar username, judul post, dan nama subgroddit. Setiap kali data dimuat atau saat ada entitas baru yang dibuat (misalnya user atau post baru), string yang relevan akan dimasukkan ke Trie karakter demi karakter. Dengan demikian, ketika pengguna memasukkan sebuah prefix, sistem hanya perlu menelusuri Trie hingga node yang merepresentasikan prefix tersebut, yang dapat dicapai dalam kompleksitas O(k), di mana k adalah panjang prefix. Dari node prefix ini, sistem kemudian melakukan traversal (misalnya preorder) untuk mengunjungi semua cabang di bawahnya dan mengumpulkan seluruh string yang memiliki awalan tersebut. Hasil traversal ini dikonversi menjadi daftar kandidat (misalnya daftar user, daftar post, atau daftar subgroddit) yang cocok dengan prefix yang diminta, lalu

ditampilkan ke pengguna dalam format yang terstruktur. Pendekatan ini jauh lebih efisien dibandingkan harus melakukan pencarian linear ke seluruh daftar setiap kali pengguna mengetikkan query.

2.2.4. Following Recommendation (BFS Graph)

Fitur Following Recommendation dibangun di atas struktur Graph berorientasi user, dengan representasi adjacency list. Setiap user direpresentasikan sebagai sebuah simpul (vertex), sedangkan hubungan follow direpresentasikan sebagai sisi berarah dari user A ke user B jika A mengikuti B.

Untuk menghasilkan rekomendasi akun yang mungkin menarik bagi pengguna saat ini, sistem melakukan traversal BFS (Breadth-First Search) mulai dari simpul current user hingga kedalaman tertentu (misalnya 2–3 level). Ide utamanya adalah mencari friends-of-friends: user yang tidak langsung di-follow oleh pengguna, tetapi berada "dekat" dalam graph sosial.

Selama atau setelah BFS, sistem menghitung jumlah mutual connections untuk setiap kandidat (yaitu berapa banyak akun yang di-follow bersama oleh pengguna dan kandidat tersebut). Nilai mutual ini digunakan sebagai skor prioritas kandidat. Setelah semua skor dihitung, daftar kandidat diurutkan berdasarkan skor mutual secara menurun, lalu dipotong hingga sejumlah maksimum untuk ditampilkan sebagai rekomendasi Who to follow yang ringkas namun relevan.

2.2.5. Content Moderation

Untuk menjaga kualitas konten, Credit menyertakan fitur Content Moderation berbasis daftar kata terlarang (*blacklist*) yang didefinisikan pada file konfigurasi. Saat pengguna menulis post atau komentar, input teks tersebut diproses melalui mesin kata sehingga terpecah menjadi token kata.

Setiap token kemudian dibandingkan dengan daftar kata terlarang yang telah dimuat ke memori pada saat inisialisasi aplikasi. Jika ditemukan satu atau lebih token yang cocok dengan *blacklist*, sistem akan menolak post atau komentar tersebut. Penolakan disertai dengan pesan yang menjelaskan bahwa konten mengandung kata terlarang, sehingga pengguna memahami alasan penolakan dan dapat menyesuaikan kembali isi tulisannya. Dengan cara ini, kebijakan moderasi dapat diubah secara terpusat hanya dengan memodifikasi daftar kata terlarang di konfigurasi, tanpa perlu mengubah kode program.

2.2.6. Data Security (Hashing + Enkripsi)

Dari sisi keamanan, sistem tidak pernah menyimpan password dalam bentuk plaintext. Sebaliknya, password yang dimasukkan pengguna akan diolah menggunakan fungsi hash sehingga yang tersimpan di struktur data dan file konfigurasi hanyalah nilai hash-nya saja. Pada saat login, input password pengguna kembali di-hash, dan hasilnya dibandingkan dengan hash yang tersimpan; autentikasi hanya berhasil jika kedua nilai hash tersebut sama.

Selain hashing password, Credit juga menyediakan mode keamanan di mana seluruh file konfigurasi dapat disimpan dalam bentuk terenkripsi. Ketika mode ini aktif, proses save akan menulis data ke disk dalam bentuk terenkripsi, dan proses load akan secara otomatis mendeteksi format terenkripsi tersebut untuk kemudian melakukan dekripsi sebelum parsing lebih lanjut. Mode keamanan ini dikendalikan oleh sebuah flag global yang bertahan selama eksekusi program, sehingga perilaku baca/tulis terhadap file konfigurasi akan konsisten sesuai pengaturan yang dipilih di awal.

2.2.7. Kreativitas

Program menambahkan elemen estetika seperti banner ASCII, warna terminal, dan efek loading. Fitur interaktif tambahan juga disertakan untuk meningkatkan pengalaman pengguna.

3 Struktur Data (ADT)

3.1 ADT Sederhana

src/adt/ADTSederhana/ADTSederhana.h

```
type User             : < user_id     : Word,
                          username    : Word,
                          password    : Word,
                          karma       : integer,
                          created_at  : time_t >

type SubGroddit       : < subgroddit_id   : Word,
                          name            : Word >

type Post             : < post_id         : Word,
                          subgroddit_id   : Word,
                          author_id       : Word,
                          title           : Word,
                          content         : Word,
                          created_at      : time_t,
                          upvotes         : integer,
                          downvotes       : integer >

type Comment          : < comment_id        : integer,
                          post_id                 : Word,
                          author_id               : Word,
                          parent_comment_id : integer,
                          content                 : Word,
                          upvotes                 : integer,
                          downvotes               : integer >

type Voting           : < user_id     : Word,
                          target_type : Word,
                          target_id   : Word,
                          vote_type   : Word >

type Social           : < user_id         : Word,
                          following_id    : Word >
```

src/utils/GlobalData/GlobalData.h

```
USERS                 : array[0..USER_CAPACITY] of User
USER_CAPACITY         : integer
USER_COUNT            : integer

SUBGRODDITS         : List    of Subgroddit { Linked List }
SUBGRODDIT_COUNT    : integer

POSTS                 : List  of Post       { Linked List }
POST_COUNT            : integer

COMMENTS            : array[0..COMMENT_CAPACITY] of Comment
COMMENT_CAPACITY    : integer
COMMENT_COUNT       : integer

VOTINGS               : array[0..VOTING_CAPACITY] of Voting
VOTING_CAPACITY       : integer
```

```
VOTING_COUNT          : integer

SOCIALS               : array[0..SOCIAL_CAPACITY] of Social
SOCIAL_CAPACITY       : integer
SOCIAL_COUNT          : integer
```

Dalam pengerjaan tugas besar ini, berikut adalah struktur ADT sederhana yang digunakan. Kelompok kami menggunakan 6 jenis ADT sederhana berbeda dalam menyimpan data yang berhasil di-load dari file csv. ADT sederhana ini akan disimpan dalam bentuk list dinamis secara eksplisit agar memudahkan pengerjaan, tetapi khusus untuk kumpulan ADT SubGroddit dan Post, kelompok kami memilih untuk menggunakan linked list karena banyaknya SubGroddit dan Post dapat bertambah secara signifikan seiring berjalannya waktu. Dalam konteks ini, kami memilih untuk menggunakan tipe data Word daripada string atau array of character karena kami ingin memaksimalkan penggunaan tipe data yang telah kami buat sebelumnya.

3.2 List Berkait

src/adt/ListBerkait/ListBerkait.h

```
type ElementType      : < TYPE_USER,
                          TYPE_POST,
                          TYPE_COMMENT,
                          TYPE_SUBGRODDIT >

type ListElement      : < type : ElementType,
                          data : < user       : User,
                                   post       : Post,
                                   comment    : Comment,
                                   subgroddit : SubGroddit  > >

type Node             : < element : ListElement
                          next    : pointer to Node >

type List             : < head    : pointer to Node >

{ KONSTRUKTOR ELEMENT }
function MakeUserElement(u : User) → ListElement
function MakePostElement(p : Post) → ListElement
function MakeCommentElement(c : Comment) → ListElement
function MakeSubgrodditElement(s : SubGroddit) → ListElement

{ NODE / LIST CREATION }
function NewNode(elem : ListElement) → pointer to Node
{ Menghasilkan node baru, NIL jika alokasi gagal }

procedure CreateList(L : pointer to List)
{ Inisialisasi list kosong }

function IsEmptyList(L : List) → boolean
{ true jika L.head = NIL }

function ListLength(L : List) → integer
{ Menghasilkan jumlah elemen list }
```

```
{ OPERASI INSERT }
procedure InsertFirstList(L : pointer to List, elem : ListElement)
procedure InsertLastList(L : pointer to List, elem : ListElement)
procedure InsertAtList(L : pointer to List, elem : ListElement, idx : integer)

{ OPERASI DELETE }
procedure DeleteFirstList(L : pointer to List, out : pointer to ListElement)
procedure DeleteLastList(L : pointer to List, out : pointer to ListElement)
procedure DeleteAtList(L : pointer to List, idx : integer, out : pointer to ListElement)

{ AKSES ELEMEN }
function GetElementAt(L : List, idx : integer, out : pointer to ListElement) → boolean

function SetElementAt(L : pointer to List, idx : integer, elem : ListElement) → boolean
{ Mengganti elemen pada index tertentu }

{ OPERASI SEARCH }
function indexOfList(L : List, target : ListElement, cmp : function pointer) → integer
{ Mencari index element dengan comparator, return -1 jika ga ketemu }

function findElement(L : List, target : ListElement, cmp : function pointer) → pointer to
ListElement
{ Mencari elemen dan return pointer, NULL jika tidak ketemu }

{ FILTER & COUNT }
function FilterList(L : List, predicate : function pointer, context : pointer) → List
{ Membuat list baru berisi element yang memenuhi predicate }

function countIf(L : List, predicate : function pointer, context : pointer) → integer
{ Menghitung jumlah elemen yang memenuhi predicate }

{ OPERASI SORTING }
procedure sortList(L : pointer to List, cmp : function pointer)
{ Mengurutkan list menggunakan comparator (in-place) }

function sortedCopyList(L : List, cmp : function pointer) → List
{ Membuat copy list yang sudah diurut }

{ OPERASI UTILITY }
function copyList(L : List) → List
procedure reverseList(L : pointer to List)
procedure reversedCopyList(L : List) → List
procedure displayList(L : List, print : function pointer)
function concatList(L1, L2: list) → List
procedure freeList(L : pointer to List)
```

ADT ListBerkait merupakan implementasi *linked list* yang kami rancang untuk menyimpan berbagai tipe data secara dinamis dalam satu struktur. Struktur datanya terdiri dari Node yang saling terhubung melalui pointer, di mana setiap node menyimpan ListElement yang dapat berupa User, Post, Comment, atau SubGroddit menggunakan mekanisme *tagged union*. ADT ini kami rancang agar bisa menyelesaikan persoalan penyimpanan koleksi data heterogen yang ukurannya tidak dapat diprediksi di awal, seperti daftar posts yang terus bertambah atau subgroddits yang dibuat secara dinamis. Pemilihan *linked list* dipilih karena kami rasa bisa memberikan fleksibilitas dalam operasi insert dan delete tanpa memerlukan realokasi memori

seperti pada array dinamis serta memungkinkan pertumbuhan struktur data sesuai kebutuhan runtime.

Implementasi ListBerkait menyediakan berbagai operasi tinggi seperti filtering, sorting, dan searching yang sangat berguna untuk fitur-fitur kompleks dalam tugas besar ini. Operasi seperti `filterList()` memungkinkan ekstraksi subset data berdasarkan kriteria tertentu (misalnya posts dalam subgroddit tertentu), sementara `sortList()` mendukung pengurutan berdasarkan berbagai params seperti upvotes atau timestamp. ADT ini juga menggunakan pendekatan functional programming dengan fallback functions sehingga logika dapat dipisahkan dari implementasi struktur data. Hal ini terbukti sangat efektif dalam mendukung fitur-fitur seperti VIEW_SUBGRODDIT yang memerlukan filtering dan sorting posts, TRENDING yang menganalisis posts dalam rentang waktu tertentu, dan SHOW_FEED yang menampilkan posts daari pengguna yang diikuti.

3.3 Mesin Karakter dan Mesin Kata

src/adt/MesinKarakter/MesinKarakter.h

```
constant CHAR_MARK           : ';'   : character
constant BLANK               : ' '   : character
constant NEWLINE             : '\n'  : character

constant MODE_INPUT          : 0     : integer
constant MODE_CSV_FILE       : 1     : integer
constant MODE_CONF_FILE      : 2     : integer
constant MODE_CSV_BUFFER     : 3     : integer
constant MODE_JSON_FILE      : 4     : integer

currentChar : character,
EOP         : boolean,
pita        : FILE*,
MODE        : integer,

procedure STARTINPUT()
{ Memulai pembacaan karakter dari stdin }

procedure ADVINPUT()
{ Mengambil karakter berikutnya dari stdin }

procedure STARTCSV(filename : string)
{ Memulai pembacaan karakter dari file CSV }

procedure ADVCSV()
{ Mengambil karakter berikutnya dalam mode CSV }

procedure STARTCONF(filename : string)
{ Memulai pembacaan karakter dari file konfigurasi }

procedure ADVCONF()
{ Mengambil karakter berikutnya dalam mode CONF }

procedure STARTCSV_BUFFER(buffer : uint8_t*, len : integer)
{ Memulai pembacaan CSV dari buffer memory }
```

```
procedure ADVCSV_BUFFER()
{ Mengambil karakter berikutnya dari buffer CSV }

jsonChar    : character
JSON_EOF    : boolean

procedure STARTJSON(filename : string)
{ Memulai pembacaan karakter dari file JSON }

procedure ADVJSON()
{ Mengambil karakter berikutnya dalam mode JSON }

procedure IgnoreWSJSON()
{ Mengabaikan whitespace (spasi, newline, tab) dalam JSON }

function CopyJSONString(out : string, maxLen : integer) → integer
{ Menyalin isi string JSON ke buffer out hingga maxLen }

procedure CLOSEPITA()
{ Menutup pita pembacaan karakter }

procedure IgnoreBlanks()
{ Mengabaikan karakter blank ' ' }

procedure IgnoreNewline()
{ Mengabaikan karakter newline '\n' }
```

src/adt/MesinKata/MesinKata.h

```
constant NMax : 200 : integer

type Word : <
    TabWord : array[0..NMax-1] of character,
    Length  : integer >

type MesinKata : <
    currentWord   : Word,
    EndWordInput  : boolean,
    EndWordCSV    : boolean,
    EndWordCONF   : boolean >

procedure STARTWORD_INPUT()
{ Memulai pembacaan kata dari input standar }

procedure ADVWORD_INPUT()
{ Mengambil kata berikutnya dari input standar }

procedure CopyWordInput()
{ Menyalin karakter pada pita menjadi currentWord pada mode input }

procedure STARTWORD_CSV(filename : string)
{ Memulai pembacaan field dari file CSV }

procedure STARTWORD_CSV_BUFFER(buffer : integer, len : integer)
{ Memulai pembacaan CSV dari buffer memory }

procedure ADVWORD_CSV()
```

```
{ Mengambil field berikutnya dalam mode CSV }

procedure CopyFieldCSV()
{ Menyalin satu field CSV ke currentWord }

procedure STARTWORD_CONF(filename : string)
{ Memulai pembacaan token dari file konfigurasi }

procedure ADVWORD_CONF()
{ Mengambil token berikutnya dalam mode CONF }

procedure CopyTokenCONF()
{ Menyalin token CONF ke currentWord }

procedure IgnoreSpacesCONF()
{ Mengabaikan spasi berlebih dalam file CONF }
```

Mesin karakter dan Mesin kata merupakan dua ADT yang digunakan untuk melakukan pemrosesan teks secara terstruktur dalam program ini. Mesin karakter bekerja sebagai pembaca aliran karakter dari berbagai sumber, seperti stdin, file CSV, file konfigurasi, buffer memori, dan file JSON. Struktur datanya menggunakan sebuah pita pembacaan karakter, penunjuk karakter aktif (*cc*), serta penanda akhir pembacaan atau MARK. ADT ini menyelesaikan persoalan bagaimana membaca masukan secara sekuensial dan konsisten tanpa bergantung pada mekanisme I/O bawaan bahasa C yang *low-level*. Pemilihan model mesin karakter dipilih karena memberikan kontrol penuh terhadap *parsing* manual format teks seperti CSV, JSON, maupun file konfigurasi sederhana.

Sementara itu, Mesin kata memanfaatkan Mesin Karakter sebagai dasar untuk membangun abstraksi tingkat lebih tinggi, yaitu pembacaan satuan “kata” atau umum disebut sebagai *token*. MesinKata menggunakan struktur data Word yang menyimpan array karakter serta panjang kata sehingga memungkinkan pengelolaan token secara eksplisit. ADT ini menyelesaikan persoalan ekstraksi token dari berbagai format input pula. Hal ini mencegah ketergantungan kami pada fungsi string library yang kurang fleksibel.

3.4 Tree

src/adt/PostTree/PostTree.h

```
type AddrComment      : pointer to CommentNode
type AddrChild        : pointer to ChildNode

type ChildNode        : < info : AddrComment,
                          next : AddrChild >

type CommentChildrenList : < head : AddrChild >

type CommentNode      : < data     : Comment,
                          children : CommentChildrenList >

type PostTree         : < root     : Post,
                          children : CommentChildrenList >

procedure CreatePostTree(input/output T : pointer to PostTree, input p : Post)
{ I.S. T sembarang, p terdefinisi
```

```
  F.S. T.root berisi salinan data Post p, T.children diinisialisasi sebagai
       list kosong (head = NIL). }

function NewCommentNode(c : Comment) → AddrComment
{ Mengalokasikan node komentar baru dengan data c dan children kosong. Mengembalikan alamat
node baru, atau NIL jika alokasi gagal. }

procedure AddCommentToRoot(input/output T : pointer to PostTree, input c : Comment)
{ I.S. T terdefinisi sebagai PostTree
  F.S. Node komentar baru berisi c ditambahkan sebagai anak langsung
       dari root (level-1) pada akhir list T.children. }

procedure AddCommentToNode(input parent : AddrComment, input c : Comment)
{ I.S. parent adalah node komentar yang valid
  F.S. Node komentar baru berisi c ditambahkan sebagai anak dari parent
       pada akhir list parent.children. }

procedure AddComment(input/output T : pointer to PostTree, input parent_comment_id :
integer, input c : Comment)
{ I.S. T terdefinisi
  F.S. Jika parent_comment_id = -1, komentar c ditambahkan sebagai anak root.
       Jika parent_comment_id != -1, mencari node dengan ID tersebut menggunakan
       DFS, lalu menambahkan c sebagai anaknya.
       Jika parent tidak ditemukan, menampilkan pesan kesalahan. }

function FindCommentInTree(T : pointer to PostTree, comment_id : integer) → AddrComment
{ Mencari node komentar dengan comment_id tertentu menggunakan DFS. Mengembalikan pointer ke
node jika ditemukan, NIL jika tidak ada. }

procedure DeleteCommentSubtree(input/output T : pointer to PostTree, input comment_id :
integer)
{ I.S. T terdefinisi, comment_id adalah ID komentar yang akan dihapus
  F.S. Node komentar dengan comment_id beserta seluruh subtree anaknya
       dihapus dari tree dan memorinya dibebaskan secara rekursif.
       Jika comment_id tidak ditemukan, tree tidak berubah. }

procedure PrintPostTree(input T : pointer to PostTree)
{ I.S. T terdefinisi
  F.S. Seluruh komentar pada tree ditampilkan ke layar dengan indentasi
       sesuai kedalaman masing-masing node (traversal preorder). }
```

ADT PostTree merepresentasikan struktur hierarki komentar pada sebuah Post menggunakan model first-child next-sibling. Setiap Post menjadi root, dan komentar disimpan dalam linked list anak. Struktur ini memungkinkan kedalaman komentar tak terbatas dan mendukung operasi cascading delete yang menghapus seluruh subtree secara rekursif. Pemilihan DFS untuk pencarian komentar memberikan efisiensi karena traversal dilakukan secara depth-first sesuai dengan struktur hierarki komentar.

3.5 Trie

src/adt/Trie/Trie.h

```
type TrieNode         : < isEnd : boolean,
                          index : integer,
                          next  : array[0..94] of pointer to TrieNode >
{ isEnd menandai akhir key, index menyimpan indeks keberadaan (-1 jika tidak ada), next
adalah array pointer anak untuk karakter ASCII 32..126. }
```

```
type Trie             : < root : pointer to TrieNode >

procedure createTrie(input/output T : pointer to Trie)
{ I.S. T sembarang
  F.S. T.root menunjuk ke node baru yang kosong dengan isEnd = false,
       index = -1, dan seluruh elemen next = NIL. }

procedure freeTrie(input/output T : pointer to Trie)
{ I.S. T terdefinisi
  F.S. Seluruh node dalam Trie dibebaskan dari memori secara rekursif,
       T.root diset menjadi NIL. }

procedure trieInsert(input/output T : pointer to Trie, input key : string, input index :
integer)
{ I.S. T terdefinisi, key adalah C-string null-terminated
  F.S. Key dimasukkan ke dalam Trie karakter per karakter.
       Node akhir ditandai isEnd = true dengan index = parameter index.
       Karakter di luar rentang ASCII 32..126 diabaikan. }


function trieSearchPrefix(T : pointer to Trie, prefix : string, outCount : pointer to
integer) → pointer to array of integer
{ Mencari semua entitas yang memiliki prefix tertentu. Mengembalikan array dinamis berisi
index entitas (panjang = *outCount). Mengembalikan NIL jika prefix tidak ditemukan atau
alokasi gagal. Caller bertanggung jawab membebaskan array hasil dengan free(). }

procedure wordToLowercaseString(output out : string, input w : Word)
{ I.S. w adalah Word terdefinisi, out adalah buffer berukuran minimal NMax+1
  F.S. out berisi representasi lowercase dari w, null-terminated. }

procedure trieInsertWordLower(input/output T : pointer to Trie,
                              input w : Word,
                              input index : integer)
{ I.S. T terdefinisi
  F.S. Word w dikonversi ke lowercase lalu dimasukkan ke Trie
       dengan index yang diberikan. }

function trieSearchPrefixWordLower(T : pointer to Trie, prefixWord : Word,
                                   outCount : pointer to integer)
        → pointer to array of integer
{ Mencari entitas dengan prefix sesuai Word (case-insensitive). Word dikonversi ke lowercase
sebelum pencarian. Mengembalikan array dinamis berisi index hasil, NIL jika tidak ada. }
```

ADT Trie (prefix tree) digunakan untuk pencarian berbasis prefix yang efisien dengan kompleksitas O(k) di mana k adalah panjang prefix. Setiap node menyimpan array 95 pointer anak untuk mendukung karakter ASCII printable (32..126). Trie digunakan dalam fitur Advanced Search untuk mencari username, judul post, dan nama subgroddit secara case-insensitive. Fungsi helper berbasis Word memudahkan integrasi dengan ADT MesinKata yang sudah ada.

3.6 Graph

src/adt/Graph/Graph.h

```
type AdjNode : < v : integer,
```

```
                next : pointer to AdjNode >

type Graph : < nVertex : integer,
               adj : array[0..nVertex-1] of pointer to AdjNode >

constant IDX_UNDEF : integer = -1

procedure createGraph(input/output G : Graph, input nVertex : integer)
{ I.S. G sembarang
  F.S. Jika nVertex > 0 dan alokasi berhasil, G berisi graf berarah dengan
       nVertex buah simpul bernomor 0..nVertex-1 dan semua adj[i] = NIL.
       Jika nVertex ≤ 0 atau alokasi gagal, G menjadi graf kosong dengan
       nVertex = 0 dan adj = NIL. }

procedure deallocateGraph(input/output G : Graph)
{ I.S. G terdefinisi
  F.S. Seluruh node adjacency pada G dan array adj dibebaskan dari memori
       sehingga G menjadi graf kosong dengan nVertex = 0 dan adj = NIL. }

function isValidVertex(G : Graph, v : integer) → boolean
{ Mengirimkan true jika 0 ≤ v < G.nVertex, false jika di luar rentang tersebut. }

function isAdjacent(G : Graph, u : integer, v : integer) → boolean
{ Mengirimkan true jika terdapat edge berarah (u → v) pada graf, dan false
  jika tidak ada edge tersebut atau jika u/v bukan vertex yang valid. }

function outDegree(G : Graph, u : integer) → integer
{ Menghasilkan banyaknya edge keluar dari vertex u (derajat keluar).
  Jika u tidak valid, hasil fungsi adalah 0. }

function inDegree(G : Graph, v : integer) → integer
{ Menghasilkan banyaknya edge masuk ke vertex v (derajat masuk).
  Jika v tidak valid, hasil fungsi adalah 0. }

procedure addEdge(input/output G : Graph, input u : integer, input v : integer)
{ I.S. G terdefinisi
  F.S. Jika u dan v valid, u ≠ v, dan edge (u → v) belum ada, maka edge baru
       ditambahkan ke adjacency list milik u.
       Jika salah satu syarat tidak terpenuhi, graf tidak berubah. }

procedure removeEdge(input/output G : Graph, input u : integer, input v : integer)
{ I.S. G terdefinisi
  F.S. Jika terdapat edge (u → v) dan u serta v valid, edge tersebut dihapus
       dari adjacency list milik u.
       Jika edge tidak ada atau u/v tidak valid, graf tidak berubah. }

procedure addVertex(input/output G : Graph)
{ I.S. G terdefinisi
  F.S. nVertex bertambah satu dan elemen baru adj[nVertex-1] diinisialisasi
       dengan NIL. Jika realokasi array adj gagal, G tetap tidak berubah. }

procedure printAdjacencyGraph(input G : Graph)
{ Menuliskan representasi adjacency list ke layar dalam format:
    u: v1 v2 v3 ...
  untuk setiap vertex u, terutama untuk keperluan debugging. }
```

ADT Graph merepresentasikan relasi sosial antar pengguna dalam bentuk graf berarah dengan struktur adjacency list. Setiap vertex direpresentasikan sebagai indeks integer yang dihubungkan dengan entri pada struktur data global USERS, sehingga edge (u → v) dapat diinterpretasikan sebagai pengguna dengan indeks u melakukan follow terhadap pengguna dengan indeks v. Pemilihan adjacency list memungkinkan penyimpanan relasi yang relatif jarang (sparse) secara efisien tanpa menghabiskan memori seperti pada matriks ketetanggaan.

Fungsi isValidVertex, isAdjacent, outDegree, dan inDegree menyediakan operasi query dasar yang diperlukan untuk memeriksa keberadaan simpul maupun hubungan antar pengguna. Prosedur addEdge dan removeEdge menangani perubahan hubungan sosial secara lokal pada adjacency list tanpa perlu memodifikasi keseluruhan struktur, sehingga operasi follow dan unfollow dapat dieksekusi dengan kompleksitas yang wajar. Prosedur addVertex digunakan ketika terjadi penambahan pengguna baru ke dalam sistem sehingga jumlah simpul graf bertambah dan adjacency list diperluas secara dinamis. Terakhir, printAdjacencyGraph berfungsi sebagai alat bantu debugging untuk memverifikasi isi graf, misalnya ketika menguji fitur rekomendasi teman berbasis BFS yang berjalan di atas struktur ADT ini.

3.7 Heap

src/adt/Heap/Heap.h

```
type HeapElType : < key : integer,
                    dataIdx : integer >
{ key menyimpan nilai prioritas, misalnya timestamp post atau skor numerik
  lain, sedangkan dataIdx menyimpan indeks ke struktur data lain
  seperti array atau list Post. }

type Heap : < buffer : array[0..capacity-1] of HeapElType,
              size : integer,
              capacity : integer,
              isMax : boolean >
{ Jika isMax = true, heap berperilaku sebagai max-heap (key terbesar di root).
  Jika isMax = false, heap berperilaku sebagai min-heap (key terkecil di root). }

procedure createHeap(input/output H : Heap, input initialCap : integer, input isMax :
boolean)
{ I.S. H sembarang
  F.S. Jika initialCap > 0 dan alokasi berhasil, H berisi heap kosong dengan:
         buffer teralokasi berkapasitas initialCap,
         size = 0,
         capacity = initialCap,
         isMax sesuai nilai parameter.
       Jika initialCap ≤ 0 atau alokasi gagal, H menjadi heap kosong tanpa
       buffer dengan buffer = NIL, size = 0, dan capacity = 0. }

procedure deallocateHeap(input/output H : Heap)
{ I.S. H terdefinisi
  F.S. Memori buffer heap dilepas dari memori sehingga H menjadi heap kosong
       tanpa buffer dengan buffer = NIL, size = 0, dan capacity = 0. }

function isHeapEmpty(H : Heap) → boolean
{ Mengirimkan true jika H.size = 0, false jika tidak. }

function heapSize(H : Heap) → integer
{ Menghasilkan banyaknya elemen yang tersimpan pada heap, yaitu H.size. }
```

```
function heapCapacity(H : Heap) → integer
{ Menghasilkan kapasitas buffer heap, yaitu H.capacity. }

function heapPush(H : pointer to Heap, x : HeapElType) → boolean
{ I.S. H terdefinisi
  F.S. Elemen x disisipkan ke dalam heap. Jika buffer penuh, kapasitas
       dapat dinaikkan melalui realokasi (misalnya dikalikan dua).
       Jika realokasi dan penyisipan sukses, H.size bertambah satu dan
       properti heap (max-heap atau min-heap sesuai isMax) tetap terjaga,
       fungsi mengirimkan true.
       Jika realokasi gagal, H tidak berubah dan fungsi mengirimkan false. }

function heapTop(H : Heap, out : pointer to HeapElType) → boolean
{ I.S. H terdefinisi, mungkin kosong
  F.S. Jika H tidak kosong, nilai elemen root (elemen dengan prioritas
       tertinggi sesuai mode heap) disalin ke lokasi yang ditunjuk oleh out,
       dan fungsi mengirimkan true.
       Jika H kosong, out tidak diubah dan fungsi mengirimkan false. }

function heapPop(H : pointer to Heap, out : pointer to HeapElType) → boolean
{ I.S. H terdefinisi, mungkin kosong
  F.S. Jika H tidak kosong, elemen root (prioritas tertinggi) dihapus dari heap,
       nilainya disalin ke lokasi yang ditunjuk oleh out, H.size berkurang
       satu, dan properti heap tetap terjaga, fungsi mengirimkan true.
       Jika H kosong, heap tidak berubah, out tidak diubah, dan fungsi
       mengirimkan false. }
```

ADT Heap dimodelkan sebagai binary heap yang direalisasikan di atas array kontigu dan berfungsi sebagai priority queue generik. Setiap elemen menyimpan pasangan (key, dataIdx), sehingga struktur ini dapat digunakan untuk mengurutkan entitas lain secara tidak langsung menggunakan indeks ke struktur data eksternal, misalnya daftar post. Parameter isMax memungkinkan heap dikonfigurasi sebagai max-heap untuk mengekstrak elemen dengan prioritas tertinggi terlebih dahulu, atau sebagai min-heap untuk kasus lain yang membutuhkan prioritas terendah di depan.

Fungsi isHeapEmpty, heapSize, dan heapCapacity menyediakan operasi dasar untuk memeriksa keadaan heap dan kapasitasnya, yang diperlukan sebelum melakukan operasi mutasi. Prosedur createHeap dan deallocateHeap mengelola siklus hidup buffer internal sehingga alokasi dan pelepasan memori dilakukan secara terkontrol. Operasi utama berupa heapPush, heapTop, dan heapPop dimanfaatkan untuk menyisipkan elemen baru, melihat elemen berprioritas tertinggi, serta mengeluarkannya dari struktur tanpa melanggar properti heap. Dalam konteks fitur Feed, heap berperan sebagai mesin pengurut yang efisien untuk menyajikan daftar post berdasarkan timestamp atau skor relevansi sehingga urutan tampilannya selalu konsisten dengan kriteria prioritas yang ditentukan program.

4 Program Utama

4.1 Inisialisasi

`src/utils/Initialize/Initialize.h`

```
USE MesinKata, ADTSederhana, Load, Helper, GlobalData, User, Kreativitas, Security,
ContentModeration

procedure buildPath(output out : string,
                    input folder : string,
                    input file : string)
{ I.S. folder dan file adalah string yang valid
  F.S. out berisi path lengkap hasil konkatenasi "config/folder/file".

  Contoh:
    buildPath(out, "config-1", "user.csv")
    → out = "config/config-1/user.csv" }


function isFolderValid(folder : string) → integer
{ Memvalidasi apakah folder konfigurasi tersedia dan dapat diakses.
  Mengembalikan 1 jika folder valid, 0 jika tidak valid.

  Validasi:
    - Folder harus ada di direktori config/
    - Semua file yang diperlukan harus ada (user.csv, post.csv, dll) }

procedure initialize()
{ I.S. Sistem dalam keadaan awal, belum ada data ter-load
  F.S. Sistem siap digunakan dengan data ter-load dari folder konfigurasi.
       Global arrays (USERS, POSTS, COMMENTS, SUBGRODDITS, SOCIALS, VOTINGS)
       terisi dengan data dari file CSV. Social graph dan content moderation
       ter-inisialisasi. Welcome screen ditampilkan.

  Algoritma:
    1. Tampilkan startup banner dengan ASCII art
    2. Spinner animation "Starting up system..."
    3. Minta user input folder konfigurasi
    4. Validasi folder dengan isFolderValid()
    5. Build paths untuk semua file (user, post, comment, dll)
    6. Load security configuration (password hashing, file encryption)
    7. Load data dari 6 file CSV:
       a. loadComments() - [1/7] Loading comments data...
       b. loadPosts()    - [2/7] Loading posts data...
       c. loadUsers()    - [3/7] Loading users data...
       d. loadSubgroddits() - [4/7] Loading subgroddits data...
       e. loadSocial()   - [5/7] Loading social connections...
       f. loadVoting()   - [6/7] Loading voting records...
    8. Build social graph untuk friend recommendation - [7/7]
    9. Initialize content moderation (blacklist words)
    10. Tampilkan statistik data ter-load
    11. Pause "Press ENTER to continue..."
    12. Panggil printWelcomeScreen()

  Error Handling:
    - Folder tidak valid: tampilkan error dan minta input ulang
    - File tidak dapat dibuka: tampilkan peringatan
    - Data corrupt: skip dan lanjutkan loading }

procedure printWelcomeScreen()
{ I.S. Sistem sudah ter-inisialisasi dengan data
  F.S. Welcome screen ditampilkan dengan informasi quick start.
```

```
  Tampilan:
    - Judul "GRODDIT"
    - Tagline platform
    - Quick start guide (REGISTER, LOGIN, HELP)
    - System status dengan jumlah users, posts, subgroddits }
```

Fitur Initialize berfungsi sebagai bootstrap layer yang mempersiapkan seluruh infrastruktur data groddit. Modul ini menggunakan dynamic arrays untuk semua entities (USERS, POSTS, COMMENTS, SUBGRODDITS, SOCIALS, VOTINGS) yang dialokasikan di memori heap dengan kapasitas initial dan dapat di-extend. Struktur graph untuk social network di-build setelah loading data social untuk mendukung BFS based friend recommendation. Desain ini juga melakukan lazy initialization untuk content moderation dengan load blacklist words hanya jika file tersedia. Path building menggunakan string concatenation untuk fleksibilitas multi environment (berbagai folder config).

Fitur ini menyelesaikan persoalan mengenai pemuatan semua state dari CSV files ke memori. Pemilihan prosedur initialize() sebagai single entry point bisa membuat semua data terload sepenuhnya sebelum aplikasi kami siap digunakan. Fungsi isFolderValid() memastikan data integrity dengan validasi awal sebelum loading dimulai. Prosedur buildPath() memisahkan concern path management untuk portability lintas OS.

4.2 Pengguna

src/utils/User/User.h

```
USE MesinKata, GlobalData, Helper, Security, Kreativitas

function findIdByUsername(username : string) → integer
{ Mencari index user di array USERS berdasarkan username.
  Mengembalikan index jika ditemukan, -1 jika tidak ada.

  Algoritma:
    Iterasi linear pada USERS[0..USER_COUNT-1]
    Return index i jika USERS[i].username = username }

function findUsernameById(id : integer) → string
{ Menghasilkan username dari USERS berdasarkan index.
  Mengembalikan pointer ke username string jika index valid,
  NULL jika index di luar range. }

function findUserIndexById(userId : string) → integer
{ Mencari index user di USERS berdasarkan user_id (e.g., "USER001").
  Mengembalikan index jika ditemukan, -1 jika tidak ada.

  Algoritma:
    Iterasi linear pada USERS[0..USER_COUNT-1]
    Return index i jika USERS[i].user_id = userId }

procedure generateUserID(output id : string, input num : integer)
{ I.S. num adalah nomor urut user baru
  F.S. id berisi user_id dalam format "USERXXX" dengan zero-padding.
```

```
  Contoh:
    generateUserID(id, 1)   → id = "USER001"
    generateUserID(id, 42)  → id = "USER042"
    generateUserID(id, 999) → id = "USER999" }

procedure registerUser()
{ I.S. Sistem siap menerima pendaftaran user baru
  F.S. User baru ditambahkan ke USERS array dengan username dan password
       yang ter-hash (jika security enabled). CURRENT_USER tetap NULL.

  Proses:
    1. Cek apakah user sudah login (tidak boleh register saat login)
    2. Input username dari mesin kata
    3. Validasi username belum terdaftar (findIdByUsername)
    4. Input password dari mesin kata
    5. Validasi panjang password (8-20 karakter)
    6. Generate user_id baru dengan generateUserID
    7. Hash password jika security password enabled
    8. Inisialisasi karma = 0, created_at = current time
    9. Tambahkan ke USERS array dengan ensureCapacity
    10. Tampilkan account summary (ID, username, karma, join date)

  Validasi:
    - User belum login
    - Username unik (tidak duplikat)
    - Password 8-20 karakter

  Error Messages:
    - "Already logged in" jika CURRENT_USER != NULL
    - "Username already exists" jika username duplikat
    - "Password must be 8-20 characters" jika tidak valid }


procedure loginUser()
{ I.S. User belum login, sistem siap menerima kredensial
  F.S. Jika kredensial valid, CURRENT_USER diset ke pointer user yang login.
       Session aktif dan user dapat mengakses fitur yang memerlukan autentikasi.

  Proses:
    1. Cek apakah sudah ada user login (CURRENT_USER != NULL)
    2. Input username dari mesin kata
    3. Cari user dengan findIdByUsername
    4. Input password dari mesin kata
    5. Verifikasi password:
       - Jika security enabled: hash input dan compare
       - Jika tidak: compare plaintext
    6. Set CURRENT_USER = &USERS[userIndex]
    7. Tampilkan welcome message dengan account details

  Validasi:
    - Belum ada user login
    - Username terdaftar
    - Password cocok

  Error Messages:
    - "Already logged in as <username>" jika sudah login
    - "User not found" jika username tidak ada
    - "Incorrect password" jika password salah }
```

```
procedure logoutUser()
{ I.S. User sudah login (CURRENT_USER != NULL)
  F.S. CURRENT_USER diset NULL, session berakhir.
       User tidak dapat lagi mengakses fitur yang memerlukan autentikasi.

  Proses:
    1. Cek apakah user sudah login
    2. Simpan username untuk farewell message
    3. Set CURRENT_USER = NULL
    4. Tampilkan logout success message

  Validasi:
    - User harus sudah login

  Error Messages:
    - "No user is logged in" jika CURRENT_USER = NULL }
```

User management menggunakan global pointer CURRENT_USER untuk menyimpan session state user yang sedang login menunjuk ke entry di array USERS. Approach ini memungkinkan akses ke data user aktif tanpa perlu pencarian berulang. Array USERS sendiri adalah dynamic array yang di-manage dengan ensureCapacity() untuk menampung user baru tanpa batasan ukuran fixed. Setiap struct User menyimpan user_id dengan format "USERXXX", username, password, karma, dan created_at. Sistem ini menggunakan linear search untuk pencarian by username/ID dengan asumsi bahwa jumlah user tidak terlalu besar.

Fitur ini menyelesaikan persoalan identity management dan access control dalam aplikasi kami, Groddit. Desain fungsi findIdByUsername dan findUserIndexById sebagai helper terpisah mendukung reusability, digunakan tidak hanya di autentikasi tetapi juga di fitur lain seperti profile, follow, dan voting. Prosedur generateUserID mengimplementasikan auto increment ID dengan zero handling untuk konsistensi format. Command handlers didesain sebagai state machine yang saling eksklusif dengan validasi ketat untuk mencegah race condition (tidak bisa register saat login, tidak bisa login dua kali). Integrasi modul Security memungkinkan transparent password hashing, modul User tidak perlu tahu detail algoritma hash, cukup panggil fungsi hash jika security enabled.

4.3 Profil

src/utils/Profil/Profil.h

```
USE ADTSederhana, Graph, Boolean, GlobalData, Helper, User, Kreativitas

function countUserPosts(userIndex : integer) → integer
{ Menghitung jumlah post yang dibuat oleh user.
  Mengembalikan total post milik user pada index userIndex.

  Algoritma:
    Iterasi linked list POSTS, count post dengan author_id = USERS[userIndex].user_id }

function countUserComments(userIndex : integer) → integer
{ Menghitung jumlah comment yang dibuat oleh user.
  Mengembalikan total comment milik user pada index userIndex.
```

```
  Algoritma:
    Iterasi array COMMENTS, count comment dengan author_id = USERS[userIndex].user_id }

function countUserFollowers(userIndex : integer) → integer
{ Menghitung jumlah followers user.
  Mengembalikan total user yang mem-follow user pada index userIndex.

  Algoritma:
    Iterasi SOCIALS, count entry dimana following_id = USERS[userIndex].user_id }

function countUserFollowing(userIndex : integer) → integer
{ Menghitung jumlah user yang di-follow.
  Mengembalikan total user yang di-follow oleh user pada index userIndex.

  Algoritma:
    Iterasi SOCIALS, count entry dimana user_id = USERS[userIndex].user_id }

function isCurrentUserFollowing(currentIndex : integer,
                                targetIndex : integer) → boolean
{ Mengecek apakah current user sedang mem-follow target user.
  Mengembalikan true jika ada relasi follow, false jika tidak.

  Prasyarat: currentIndex dan targetIndex valid

  Algoritma:
    Cari di SOCIALS entry dengan user_id = USERS[currentIndex].user_id
    dan following_id = USERS[targetIndex].user_id }

function computeUserKarma(userIndex : integer) → integer
{ Menghitung total karma user dari semua post dan comment.
  Karma = (total upvotes) - (total downvotes) dari semua konten user.

  Algoritma:
    1. Iterasi POSTS milik user, sum (upvotes - downvotes)
    2. Iterasi COMMENTS milik user, sum (upvotes - downvotes)
    3. Return total karma }

procedure findLatestPostsForUser(input userIndex : integer,
                                 output indices : array[0..2] of integer,
                                 output foundCount : integer)
{ I.S. userIndex adalah index user yang valid
  F.S. indices berisi index 3 post terbaru milik user (sorted by created_at desc).
       foundCount berisi jumlah post yang ditemukan (0-3).
       Jika user punya < 3 post, sisanya diisi -1.

  Algoritma:
    1. Inisialisasi indices dengan -1
    2. Untuk k = 0 sampai 2:
       a. Cari post dengan created_at terbesar yang belum dipilih
       b. Simpan index post ke indices[k]
       c. Break jika tidak ada post lagi
    3. Set foundCount sesuai jumlah post ditemukan }

procedure getSubgrodditName(input subId : Word, output out : string)
{ I.S. subId adalah subgroddit_id yang akan dicari
  F.S. out berisi nama subgroddit jika ditemukan, atau subgroddit_id
       jika tidak ditemukan.
```

```
  Algoritma:
    1. Iterasi linked list SUBGRODDITS
    2. Jika ketemu subgroddit dengan subgroddit_id = subId:
       Copy name ke out
    3. Jika tidak ketemu: Copy subId ke out (default) }

procedure showUserProfile(input username : string)
{ I.S. Sistem siap menampilkan profil user
  F.S. Profil lengkap user ditampilkan dengan format TUI professional,
       berisi account info, statistics, connection status, dan recent posts.

  Validasi:
    - User harus sudah login
    - Username harus terdaftar di sistem

  Proses:
    1. Cek user sudah login
    2. Cari user dengan findIdByUsername
    3. Clear screen dan tampilkan breadcrumb
    4. Spinner animation "Loading profile data"
    5. Hitung statistik dengan helper functions:
       - countUserPosts
       - countUserComments
       - countUserFollowers
       - countUserFollowing
       - Ambil karma langsung dari USERS[idx].karma
    7. Tampilkan success message }
```

Fitur Profile Management menggunakan aggregation pattern data profil tidak disimpan secara redundan melainkan dihitung dari berbagai sumber, POSTS (linked list), COMMENTS (array), SOCIALS (array), dan USERS (array). Approach ini menjamin data consistency karena statistik selalu reflect keadaan terkini tanpa perlu sinkronisasi manual. Fungsi findLatesPostsForUser() menggunakan selection sort pattern untuk menemukan top 3 post terbaru dengan menyimpan index sementara, menghindari modifikasi data asli. Karma diambil langsung dari field USERS[].karma yang di-maintain secara atomik saat voting terjadi. Status following menggunakan lookup pattern pada array SOCIALS dengan kompleksitas O(n).

Fitur ini menyelesaikan persoalaan user insight dan social transparency dengan menyediakan dashboard komprehensif tentang aktivitas dan pengaruh user dalam platform. Desain fungsi terpisah (countUserPosts, countUserComments, dll) mendukung modularity dan testability, setiap metric dapat di-test dan debug secara independen. Fungsi isCurrentUserFollowing() memungkinkan contextual display, profil menunjukkan status relasi antara viewer dan target user. Prosedur getSubgrodditName() sebagai mapper antara ID dan display name meningkatkan readibility output. showUserProfile sebagai orchestrator mengintegrasikan semua helper functions dengan error handling komprehensif dan progressive disclosure.

4.4 Post

`src/utils/Post/Post.h`

```
USE PostTree, GlobalData, Helper, User, Profile, ContentModeration, Kreativitas

function findSubgrodditIndexByName(subName : string) → integer
{ Mencari index subgroddit di linked list SUBGRODDITS berdasarkan nama.
  Mengembalikan index jika ditemukan, -1 jika tidak ada.

  Algoritma:
    Iterasi linked list SUBGRODDITS, return index saat name = subName }

function findPostIndexById(postId : string) → integer
{ Mencari index post di linked list POSTS berdasarkan post_id.
  Mengembalikan index jika ditemukan, -1 jika tidak ada.

  Algoritma:
    Iterasi linked list POSTS, return index saat post_id = postId }

function getPostById(postId : string) → pointer to Post
{ Mencari dan mengembalikan pointer ke Post berdasarkan post_id.
  Mengembalikan pointer ke Post jika ditemukan, NULL jika tidak ada.

  Algoritma:
    Iterasi linked list POSTS, return pointer saat post_id = postId }

function getPostAuthorId(postId : string) → string
{ Mencari author_id dari sebuah Post.
  Mengembalikan string author_id (static buffer) jika ditemukan,
  NULL jika post tidak ada.

  Algoritma:
    1. Panggil getPostById untuk mendapat pointer post
    2. Jika post != NULL, convert author_id dari Word ke string
    3. Return string atau NULL }

procedure generatePostID(output id : string, input num : integer)
{ I.S. num adalah nomor urut post baru
  F.S. id berisi post_id dalam format "PXYZ" dengan zero-padding (3 digit).

  Contoh:
    generatePostID(id, 1)   → id = "P001"
    generatePostID(id, 42)  → id = "P042"
    generatePostID(id, 999) → id = "P999" }

function getPostNumberFromWord(postId : Word) → integer
{ Mengekstrak nomor dari post_id berformat "PXYZ".
  Mengembalikan nilai integer dari 3 digit terakhir.
  Mengembalikan 0 jika format tidak valid.

  Contoh:
    "P001" → 1
    "P042" → 42
    "P999" → 999 }

function getNextPostNumberMex() → integer
{ Menghitung MEX (Minimum Excluded) dari nomor post yang sudah ada.
  MEX adalah bilangan positif terkecil yang belum digunakan.

  Algoritma:
    1. Buat array used[0..maxCheck] untuk tracking nomor terpakai
```

```
    2. Iterasi semua post, tandai used[num] = 1
    3. Cari bilangan terkecil n dimana used[n] = 0
    4. Return n sebagai MEX

  Contoh:
    Post: P001, P002, P004 → MEX = 3
    Post: P001, P003       → MEX = 2 }

procedure commandPost()
{ I.S. Sistem siap menerima input untuk membuat post baru
  F.S. Post baru ditambahkan ke linked list POSTS dengan content moderation.
       Post tervalidasi untuk inappropriate content sebelum disimpan.

  Validasi:
    - User harus sudah login
    - Subgroddit harus valid (dipilih dari daftar)
    - Title dan content tidak boleh kosong
    - Content tidak mengandung blacklisted words

  Proses:
    1. Cek user sudah login
    2. Clear screen dan breadcrumb
    3. Tampilkan daftar subgroddits untuk dipilih
    4. Input nomor subgroddit pilihan
    5. Validasi subgroddit exists
    6. Input title post
    7. Input content post (multi-line)
    8. Content moderation check dengan spinnerAnimation
    9. Generate post_id dengan MEX algorithm
    10. Alokasi dynamic memory untuk content
    11. Inisialisasi post fields:
        - post_id, subgroddit_id, author_id
        - title, content, created_at (current time)
        - upvotes = 0, downvotes = 0
    12. Insert post ke linked list POSTS (insertLastList)
    13. Tampilkan success box dengan post details

  Error Messages:
    - "Not logged in" jika belum login
    - "Invalid subgroddit selection" jika pilihan tidak valid
    - "Content contains inappropriate words" jika gagal moderasi }

procedure commandViewPost()
{ I.S. Sistem siap menampilkan detail post dengan comment tree
  F.S. Post beserta semua comments ditampilkan dalam format hierarki.
       Comments diorganisir dalam tree structure berdasarkan parent.

  Validasi:
    - User harus sudah login
    - Post harus exist

  Proses:
    1. Cek user sudah login
    2. Input post_id dari mesin kata
    3. Cari post dengan getPostById
    4. Clear screen dan breadcrumb dengan subgroddit context
    5. Tampilkan header dengan post_id
    6. Section POST DETAILS:
```

```
       - Subgroddit name, title, author, timestamp
    7. Section CONTENT:
       - Post body text
    8. Section VOTING:
       - Upvotes (green), Downvotes (red)
    9. Section COMMENTS:
       - Build comment tree dengan buildCommentTree()
       - Display tree dengan printCommentTree()
    10. Tampilkan success message

  Error Messages:
    - "Not logged in" jika belum login
    - "Post not found" jika post_id tidak ada }

procedure commandDeletePost()
{ I.S. User sudah login dan akan menghapus post miliknya
  F.S. Post dihapus dari POSTS linked list jika validasi sukses.
       Semua voting terkait post juga dihapus (cascade delete).

  Validasi:
    - User harus sudah login
    - Post harus exist
    - User harus pemilik post (authorization check)

  Proses:
    1. Cek user sudah login
    2. Clear screen dan breadcrumb
    3. Input post_id
    4. Cari post dengan getPostById
    5. Validasi post exists
    6. Spinner animation "Validating post..."
    7. Cek authorization: author_id == current user_id
    8. Tampilkan konfirmasi dengan detail post:
       - Post ID, Title, Created date
       - Warning message (irreversible action)
    9. Input konfirmasi (Y/N)
    10. Jika Y:
        a. Cascade delete: deleteVotingsByTarget(postId, "POST")
        b. Find node di linked list POSTS
        c. Hapus node dengan list operation
        d. Free dynamic memory untuk content
        e. Spinner animation "Deleting post..."
        f. Tampilkan success box dengan deletion summary
    11. Jika N: Cancel deletion

  Error Messages:
    - "Not logged in" jika belum login
    - "Post not found" jika post_id tidak ada
    - "Unauthorized" jika bukan pemilik post
    - "Deletion cancelled" jika user pilih No }
```

Fitur Post menggunakan linked list (POSTS.head) untuk menyimpan semua post dalam sistem. Setiap node berisi ListElement dengan type TYPE_POST dan data Post yang mencakup post_id, subgroddit_id, author_id, title, content (dynamic allocation), timestamps, dan voting counters. Desain linked list memungkinkan dynamic growth tanpa realokasi array dan efficient

insertion di akhir list (O(1) dengan tail pointer). Post ID menggunakan minimum excluded algorithm untuk auto increment dengan gap filling, jadi jika P001, P003 ada, post baru akan dapat ID P002. Content disimpan dengan dynamic string allocation untuk menghemat memori pada post pendek. Comment tree untuk view post dibuild menggunakan struktur PostTree (n-ary tree) berdasarkan relasi parent_comment_id.

Fitur ini menyelesaikan persoalan content discovery sebagai core functionality platform groddit. Desain fungsi getPostById() dan getPostAuthorId() sebagai accessor functions mendukung encapsulation, jadi modul lain tidak perlu tahu detail struktur linked list. Fungsi getNextPostNumberMex() mengimplementasikan ID recycling untuk efisiensi namespace sehingga ID post yang dihapus dapat digunakan kembali. Command commandPost() terintegrasi dengan content moderation untuk filtering inappropriate content sebelum publish, jadi menjamin groddit tetap aman. commandViewPost() menggunakan tree traversal untuk menampilkan comment hierarchy dengan visual indentation. commandDeletePost() mengimplementasikan cascade delete dengan memanggil deleteVotingsByTarget() untuk menjaga referential integrity, jadi semua voting yang terkait post yang dihapus juga akan ikut terhapus.

4.5 Subgroddit

src/utils/Subgroddit/Subgroddit.h

```
USE MesinKata, GlobalData, Helper, Post, Kreativitas

procedure createSubgroddit()
{ I.S. Sistem siap menerima input untuk membuat subgroddit baru
  F.S. Subgroddit baru ditambahkan ke linked list SUBGRODDITS dengan
       nama yang unique dan format yang valid.

  Validasi:
    - Nama harus dimulai dengan prefix "r/"
    - Nama minimal 3 karakter (r/ + 1 karakter)
    - Nama harus unique (tidak duplikat)

  Proses:
    1. Clear screen dan breadcrumb
    2. Tampilkan header "CREATE SUBGRODDIT"
    3. Tampilkan format guidance (r/programming, r/gaming)
    4. Input nama subgroddit dari mesin kata
    5. Validasi format (harus diawali "r/")
    6. Spinner animation "Checking name availability"
    7. Validasi nama belum digunakan (findSubgrodditIndexByName)
    8. Generate subgroddit_id dengan format "SXXX" (S001, S002, ...)
    9. Inisialisasi struct SubGroddit:
       - subgroddit_id
       - name (dari input)
    10. Insert ke linked list SUBGRODDITS (insertLastList)
    11. Increment SUBGRODDIT_COUNT
    12. Spinner animation "Creating subgroddit"
    13. Tampilkan success box dengan Name dan ID

  Format ID:
    S001, S002, ..., S999 (sequential based on count)

  Error Messages:
    - "Invalid input" jika nama kosong
```

```
    - "Invalid format" jika tidak diawali r/
    - "Name already exists" jika nama duplikat

  Contoh:
    Input: r/programming
    Output: Subgroddit ID = S042, Name = r/programming }

procedure viewSubgroddit()
{ I.S. Sistem siap menampilkan posts dalam subgroddit dengan sorting
  F.S. Daftar posts dalam subgroddit ditampilkan terurut sesuai mode
       (HOT/NEW) dan order (INCR/DECR).

  Format Command:
    VIEW_SUBGRODDIT <name> <mode> <order>;

    <name>  : nama subgroddit (e.g., r/programming)
    <mode>  : HOT (by upvotes) atau NEW (by timestamp)
    <order> : INCR (ascending) atau DECR (descending)

  Validasi:
    - Jumlah parameter harus tepat 3
    - Mode harus "HOT" atau "NEW"
    - Order harus "INCR" atau "DECR"
    - Subgroddit harus exists

  Proses:
    1. Parse 3 parameter dari mesin kata
    2. Validasi parameter lengkap dan valid
    3. Cari subgroddit dengan findSubgrodditIndexByName
    4. Collect posts yang subgroddit_id-nya match:
       a. Alokasi dynamic array PostWithSort
       b. Iterasi linked list POSTS
       c. Filter post dengan compareWord(post.subgroddit_id, target)
       d. Tentukan sortValue:
          - HOT mode: sortValue = post.upvotes
          - NEW mode: sortValue = post.created_at
       e. Expand array jika capacity penuh (realloc)
    5. Sort posts dengan Bubble Sort:
       a. Compare berdasarkan sortValue
       b. DECR: descending (terbesar ke terkecil)
       c. INCR: ascending (terkecil ke terbesar)
    6. Clear screen dan breadcrumb dinamis
    7. Tampilkan header dengan subgroddit name dan mode icon
    8. Display sorting info (mode + order + arrow)
    9. Jika postCount = 0:
       - Warning "No posts in this subgroddit"
    10. Jika postCount > 0:
       - Display numbered list posts:
         * Post number, ID, title
         * Upvotes (green ↑), downvotes (red ↓)
         * Author username dengan @ prefix
         * Created timestamp
    11. Free dynamic array

  Error Messages:
    - "Invalid command format" jika parameter kurang/lebih
    - "Invalid MODE" jika mode bukan HOT/NEW
    - "Invalid ORDER" jika order bukan INCR/DECR
```

```
    - "Subgroddit not found" jika nama tidak ada }
```

Fitur Subgroddit menggunakan linked list (SUBGRODDITS.head) untuk menyimpan kategori platform. Setiap subgroddit memiliki subgroddit_id dengan format SXXX dan format name “r/topic”. Untuk sorting posts, kami menggunakan temporary dynamic array PostWithSort yang menggabungkan struct Post dengan sortValue integer, nilai ini bisa upvotes (HOT mode) atau timestamp (NEW mode). Pemilihan dynamic array untuk sorting memungkinkan flexible capacity dengan saat jumlah post bertambah. Setelah sorting selesai, array langsung di-free untuk menghemat memori.

Fitur ini menyelesaikan persoalan content organization dan content discovery dalam platform groddit ini. createSubgroddit() mengimplementasikan namespace validation dengan enforcing prefix “r/” dan uniqueness check, mencegah adanya konflik nama. Penggunaan sequetial ID generation (S001, S002, …) lebih sederhana dibanding MEX algorithm karena subgroddit sendiri memang jarang dihapus. viewSubgroddit() menyediakan flexible content browsing dengan dual sorting modes yaitu HOT (popularity based) untuk menemukan konten trending, dan NEW (time based) untk menemukan konten terbaru. Pilihan INCR/DECR memberikan user control atas urutan tampilan. Bubble Sort dipilih untuk sorting karena simplicity dan ukuran dataset post per subgroddit biasanya kecil (< 100) sehingga O(N^2) complexity acceptable. Filetring posts dengan compareWord(post.subgroddit_id, target) menggunakan lazy evaluation, yaitu hanya post yang relevan yang di-collect ke array.

4.6 Comment and Reply

src/utils/Comment/Comment.h

```
USE GlobalData, Helper, User, Profil, Voting, ContentModeration

function getCommentById(postId : string, commentId : integer) → CommentPtr
{
I.S. Array COMMENTS terdefinisi dan COMMENT_COUNT bernilai benar.
F.S. Mengembalikan pointer ke Comment yang cocok dengan postId dan commentId;
bila tidak ada, mengembalikan NULL.
}

function getCommentAuthorId(postId : string, commentId : integer) → string
{
I.S. Struktur komentar sudah terdefinisi.
F.S. Mengembalikan author_id dari komentar yang sesuai,
atau NULL jika komentar tidak ditemukan.
}

function getCommentPtr(commentId : integer, postIdWord : Word) → CommentPtr
{
I.S. COMMENT array terdefinisi, postIdWord berisi ID post dalam bentuk Word.
F.S. Mengembalikan pointer ke Comment bila ditemukan, NULL bila tidak ada.
}

procedure deleteCommentAtIndex(input index : integer)
{
I.S. index valid, 0 ≤ index < COMMENT_COUNT.
F.S. COMMENTS[index] dihapus, elemen setelahnya digeser ke kiri,
```

```
COMMENT_COUNT berkurang satu.
}

procedure deleteCommentRecursive(input commentId : integer, input postId : string)
{
I.S. commentId dan postId valid atau mungkin tidak ditemukan.
F.S. Jika komentar ditemukan:
• komentar tersebut dihapus,
• seluruh reply yang memiliki parentId = commentId ikut dihapus,
• proses dilakukan rekursif untuk setiap turunannya.
Jika tidak ditemukan, tidak ada perubahan pada data.
}

procedure commandAddComment()
{
I.S. Input pengguna sudah siap dibaca oleh MesinKata.
Post dan user yang dituju harus valid.
F.S. Komentar baru atau reply ditambahkan ke COMMENTS.
COMMENT_COUNT bertambah satu.
Jika input tidak valid, komentar tidak ditambahkan.
}

procedure commandDeleteComment()
{
I.S. Input pengguna mengandung perintah DELETE_COMMENT, postId, commentId.
F.S. Komentar dan seluruh reply di bawahnya dihapus menggunakan
deleteCommentRecursive.
COMMENT_COUNT berkurang sesuai jumlah komentar yang terhapus.
Jika komentar tidak ditemukan, tidak ada perubahan.
}

function generateNewCommentId() → integer
{
I.S. COMMENTS terdefinisi dan bisa saja kosong.
F.S. Mengembalikan nilai (maksimum comment_id saat ini + 1).
Jika belum ada komentar sama sekali, ID pertama adalah 1.
}
```

Fitur Comment and Reply direpresentasikan menggunakan ADT Tree dengan model *first-child next-sibling*. Seluruh komentar disimpan dalam array global, tetapi hubungan antar komentar dibentuk melalui pasangan comment_id dan parent_comment_id sehingga secara logis membentuk struktur pohon. Setiap komentar utama berperan sebagai root dari sebuah subtree, sementara reply menjadi node anak yang dapat memiliki turunan sendiri. Pendekatan ini memungkinkan struktur komentar bertingkat yang fleksibel, sekaligus tetap efisien untuk disimpan dan diproses.

Masalah utama yang diselesaikan modul ini adalah penelusuran komentar, penambahan komentar baru pada posisi yang benar dalam pohon, dan terutama penghapusan komentar beserta seluruh reply di bawahnya. Penghapusan ini bersifat *cascading delete*, artinya ketika satu komentar dihapus, seluruh subtree yang berada di bawah komentar tersebut juga harus dihapus agar tidak menyisakan node tanpa induk. Tanpa model pohon, proses seperti ini tidak dapat dilakukan secara konsisten.

Fungsi dan prosedur yang ada dirancang untuk mendukung kebutuhan tersebut. Fungsi getCommentById, getCommentPtr, dan getCommentAuthorId menyediakan mekanisme

pencarian node dalam struktur pohon berdasarkan kombinasi post_id dan comment_id. Prosedur deleteCommentAtIndex menangani penghapusan fisik elemen dari array, sementara deleteCommentRecursive menghapus seluruh subtree secara rekursif sesuai karakteristik *cascading delete*. Prosedur commandAddComment dan commandDeleteComment menjadi pengendali alur operasi dari sisi pengguna, termasuk validasi input dan pemanggilan fungsi struktural yang sesuai. Terakhir, generateNewCommentId memastikan setiap node memiliki identitas unik sehingga proses pencarian, traversal, dan *cascading delete* dapat dijalankan dengan deterministik. Dengan kombinasi ini, modul Comment and Reply mampu mempertahankan integritas struktur pohon dan mendukung seluruh operasi yang diperlukan.

4.7 Voting

src/utils/Voting/Voting.h

```
USE ADTSederhana, ListBerkait, MesinKata, GlobalData, Helper, Post, Comment, User

function findVotingIndex(userId : string,
                         targetId : string,
                         targetType : string) → integer
{ Mencari index voting di array global VOTINGS.
  Mengembalikan index jika ditemukan, -1 jika tidak ada. }

  Algoritma:
    Iterasi linear pada VOTINGS[0..VOTING_COUNT-1]
    Return index i jika match userId, targetId, dan targetType }


procedure addNewVote(input userId : string,
                     input targetId : string,
                     input targetType : string,
                     input voteType : string)
{ I.S. VOTINGS array terdefinisi, voting baru akan ditambahkan
  F.S. Voting baru ditambahkan ke VOTINGS array dengan kapasitas
       yang di-expand jika diperlukan (ensureCapacity).

  Proses:
    1. Cek kapasitas array, expand jika penuh
    2. Alokasi entry baru di VOTINGS[VOTING_COUNT]
    3. Copy userId, targetId, targetType, voteType
    4. Increment VOTING_COUNT }


procedure updateVote(input votingIdx : integer, input newVoteType : string)
{ I.S. votingIdx valid (0 <= votingIdx < VOTING_COUNT)
  F.S. Vote type pada VOTINGS[votingIdx] diubah menjadi newVoteType.
       Digunakan untuk toggle antara UPVOTE dan DOWNVOTE. }

procedure removeVote(input votingIdx : integer)
{ I.S. votingIdx valid dan voting akan dihapus
  F.S. Voting pada index votingIdx dihapus dan array di-compact
       dengan menggeser elemen setelahnya ke kiri.
       VOTING_COUNT berkurang satu.}

  Algoritma:
    1. Shift semua elemen [votingIdx+1..VOTING_COUNT-1] ke kiri
    2. Decrement VOTING_COUNT }
```

```
procedure updatePostVotes(input postId : string,
                          input upvoteDelta : integer,
                          input downvoteDelta : integer)
{ I.S. postId adalah ID post yang valid
  F.S. Counter upvotes dan downvotes pada post di-update sesuai delta.
       upvoteDelta/downvoteDelta dapat bernilai +1, -1, atau 0. }

  Proses:
    1. Cari post dengan postId di array POSTS
    2. Update post.upvotes += upvoteDelta
    3. Update post.downvotes += downvoteDelta }

procedure updateCommentVotes(input postId : string,
                             input commentId : integer,
                             input upvoteDelta : integer,
                             input downvoteDelta : integer)
{ I.S. postId dan commentId adalah ID yang valid
  F.S. Counter upvotes dan downvotes pada comment di-update sesuai delta.

  Proses:
    1. Cari comment dengan postId dan commentId di COMMENTS
    2. Update comment.upvotes += upvoteDelta
    3. Update comment.downvotes += downvoteDelta }

procedure updateUserKarma(input userId : string, input karmaDelta : integer)
{ I.S. userId adalah ID user yang valid
  F.S. Karma user di-update sesuai karmaDelta (+1/-1/+2/-2).
       Karma tidak boleh negatif (minimum 0).

  Proses:
    1. Cari user dengan userId di array USERS
    2. Update user.karma += karmaDelta
    3. Jika karma < 0, set karma = 0 }

procedure deleteVotingsByTarget(input targetId : string, input targetType : string)
{ I.S. targetId dan targetType terdefinisi
  F.S. Semua voting yang terkait dengan target dihapus dari VOTINGS.
       Digunakan saat post atau comment dihapus (cascade delete).

  Algoritma:
    1. Iterasi VOTINGS dari belakang ke depan
    2. Hapus voting jika match targetId dan targetType
    3. Compact array setelah penghapusan }

procedure commentIdToString(input commentId : integer, output out : string)
{ I.S. commentId adalah integer valid
  F.S. out berisi representasi string dari commentId.
       Digunakan untuk matching dengan target_id di Voting. }

{ command handlers }
procedure commandUpvotePost()
{ I.S. User sudah login, sistem siap menerima input
  F.S. User memberikan upvote pada post yang dipilih.
       Jika sudah pernah downvote, toggle menjadi upvote.
       Update counter post, karma author, dan VOTINGS array.
```

```
  Validasi:
    - User harus login
    - Post harus exist
    - Tidak bisa vote post sendiri

  Skenario:
    1. Belum pernah vote: tambah upvote baru
    2. Sudah upvote: informasi "already upvoted"
    3. Sudah downvote: ubah downvote → upvote }

procedure commandDownvotePost()
{ I.S. User sudah login, sistem siap menerima input
  F.S. User memberikan downvote pada post yang dipilih.
       Jika sudah pernah upvote, toggle menjadi downvote.
       Update counter post, karma author, dan VOTINGS array.

  Validasi sama seperti commandUpvotePost.

  Skenario:
    1. Belum pernah vote: tambah downvote baru
    2. Sudah downvote: informasi "already downvoted"
    3. Sudah upvote: ubah upvote → downvote }

procedure commandUndoVotePost()
{ I.S. User sudah login dan pernah melakukan vote pada post
  F.S. Vote user pada post dihapus dari sistem.
       Update counter post, karma author, dan VOTINGS array.

  Validasi:
    - User harus login
    - Post harus exist
    - Harus ada vote sebelumnya

  Proses:
    1. Cari voting yang sesuai
    2. Hapus dari VOTINGS array
    3. Revert counter post
    4. Revert karma author }

procedure commandUpvoteComment()
{ I.S. User sudah login, sistem siap menerima input
  F.S. User memberikan upvote pada comment yang dipilih.
       Logika sama dengan commandUpvotePost namun untuk comment.
       Memerlukan input postId dan commentId.

  Validasi:
    - User harus login
    - Post dan comment harus exist
    - Tidak bisa vote comment sendiri }

procedure commandDownvoteComment()
{ I.S. User sudah login, sistem siap menerima input
  F.S. User memberikan downvote pada comment yang dipilih.
       Logika sama dengan commandDownvotePost namun untuk comment.
       Memerlukan input postId dan commentId. }

procedure commandUndoVoteComment()
{ I.S. User sudah login dan pernah vote pada comment
```

```
  F.S. Vote user pada comment dihapus dari sistem.
       Logika sama dengan commandUndoVotePost namun untuk comment.
       Update counter comment dan karma author. }
```

Fitur Voting menggunakan array dinamis global (VOTING) untuk menyimpan seluruh record voting dalam sistem kami. Setiap entry Voting berisi user_id, target_type (POST/COMMENT), target_id, dan vote_type (UPVOTE/DOWNVOTE). Struktur ini memungkinkan pencarian voting dengan kompleksitas O(n) dan efisien karena jumlah voting per user terbatas. Selain itu, desain kami juga terintegrasi dengan array global seperti misalnya POSTS, COMMENTS, dan USERS untuk update counter dan karma secara sinkron. Pemilihan array dinamis dengan ensureCapacity memastikan fleksibilitas penambahan data tanpa batasan, sementara operasi compact array pada penghapusan tetap menjaga efisiensi memori.

Fitur ini menyelesaikan persoalan engagement tracking dan pengukuran content quality melalui mekanisme voting yang istilahnya demokratis seperti Reddit. Pemilihan fungsi-fungsi helper seperti findVotingIndex, updatePostVotes, updateCommentVotes, updateUserKarma, memisahkan concern antara pencarian, update counter, dan update karma untuk modularitas dan kemudahan testing. Fungsi deleteVotingsByTarget mengimplementasikan cascade delete untuk menjaga integritas referensial saat konten dihapus. Command handlers kami desain sebagai prosedur terpisah yang menangani transisi state dengan validasi sehingga bisa mencegah self voting atau double voting.

4.8 Social

src/utils/Social/Social.h

```
USE Graph, User, GlobalData

procedure socialFollowUser(input targetUsername : string)
{ Melakukan FOLLOW dari current user ke user dengan username targetUsername.
  I.S.  Sistem dapat dalam keadaan sudah login atau belum.
  F.S.  Jika belum login:
          - Menuliskan pesan kesalahan bahwa pengguna harus login.
        Jika user target dengan username targetUsername tidak ditemukan:
          - Menuliskan pesan kesalahan user tidak ditemukan.
        Jika targetUsername sama dengan username current user:
          - Menuliskan pesan kesalahan bahwa user tidak dapat
            mengikuti dirinya sendiri.
        Jika current user sudah mengikuti target:
          - Menuliskan pesan informasi bahwa hubungan follow sudah ada.
        Jika seluruh validasi lolos:
          - Menambahkan edge (currentUserIdx → targetIdx) ke SOCIAL_GRAPH.
          - Menambahkan pasangan (user_id, following_id) baru ke array SOCIALS.
          - Menuliskan pesan keberhasilan FOLLOW. }

procedure socialUnfollowUser(input targetUsername : string)
{ Melakukan UNFOLLOW dari current user ke user dengan username targetUsername.
  I.S.  Sistem dapat dalam keadaan sudah login atau belum.
  F.S.  Jika belum login:
          - Menuliskan pesan kesalahan bahwa pengguna harus login.
        Jika user target tidak ditemukan:
          - Menuliskan pesan kesalahan user tidak ditemukan.
```

```
        Jika belum ada edge (currentUserIdx → targetIdx) pada SOCIAL_GRAPH:
          - Menuliskan pesan bahwa current user belum mengikuti target.
        Jika seluruh validasi lolos:
          - Menghapus edge (currentUserIdx → targetIdx) dari SOCIAL_GRAPH.
          - Menghapus entri terkait dari array SOCIALS.
          - Menuliskan pesan keberhasilan UNFOLLOW. }

procedure socialShowFollowing(input username : string)
{ Menampilkan daftar akun yang di-follow oleh user dengan username tertentu.
  I.S.  username mungkin berupa string kosong.
  F.S.  Jika username adalah string kosong:
          - Jika belum login, menuliskan pesan kesalahan bahwa pengguna
            harus login dan prosedur berhenti.
          - Jika sudah login, target user adalah current user.
        Jika username tidak kosong:
          - Jika user dengan username tersebut tidak ditemukan,
            menuliskan pesan kesalahan dan prosedur berhenti.
        Jika indeks target user tidak merupakan vertex valid pada SOCIAL_GRAPH:
          - Menuliskan pesan kesalahan data sosial dan prosedur berhenti.
        Selain itu:
          - Menuliskan judul "Daftar akun yang diikuti oleh <username_target>:".
          - Jika adjacency list SOCIAL_GRAPH pada vertex target kosong,
            menuliskan "(Tidak mengikuti siapa pun)".
          - Jika tidak kosong, menelusuri seluruh tetangga v dari vertex target
            dan untuk setiap v menuliskan "- <username_v>". }

procedure socialShowFollowers(input username : string)
{ Menampilkan daftar akun yang menjadi follower user dengan username tertentu.
  I.S.  username mungkin berupa string kosong.
  F.S.  Jika username adalah string kosong:
          - Jika belum login, menuliskan pesan kesalahan bahwa pengguna
            harus login dan prosedur berhenti.
          - Jika sudah login, target user adalah current user.
        Jika username tidak kosong:
          - Jika user dengan username tersebut tidak ditemukan,
            menuliskan pesan kesalahan dan prosedur berhenti.
        Jika indeks target user tidak merupakan vertex valid pada SOCIAL_GRAPH:
          - Menuliskan pesan kesalahan data sosial dan prosedur berhenti.
        Selain itu:
          - Menuliskan judul "Daftar akun yang mengikuti <username_target>:".
          - Melakukan iterasi untuk setiap vertex u pada SOCIAL_GRAPH dan
            menelusuri adjacency list-nya.
          - Jika ditemukan edge (u → target), menuliskan "- <username_u>"
            dan menandai bahwa setidaknya ada satu follower.
          - Jika setelah seluruh iterasi tidak ditemukan follower sama sekali,
            menuliskan "(Belum ada yang mengikuti akun ini)". }
```

Fitur Social mengelola seluruh relasi follow dan daftar hubungan sosial antar pengguna di dalam aplikasi. Informasi hubungan follow disimpan secara terstruktur dalam dua lapisan. Pertama, struktur SOCIAL_GRAPH yang direpresentasikan sebagai graf berarah dengan adjacency list menyimpan hubungan follow dalam bentuk edge (u → v) antara indeks pengguna pada array global USERS. Kedua, array dinamis SOCIALS menyimpan pasangan user_id dan following_id dalam bentuk ADT sederhana Social yang mudah di-*parse* ke berkas konfigurasi.

Pendekatan dua lapis ini dipilih agar traversal dan algoritma graf dapat bekerja efisien di atas struktur Graph, sementara penyimpanan ke berkas tetap sederhana karena cukup menulis seluruh isi array SOCIALS.

Prosedur socialFollowUser dan socialUnfollowUser dirancang sebagai *command handler* yang menangani seluruh validasi yang diperlukan sebelum mengubah struktur data. Validasi yang dilakukan mencakup pengecekan autentikasi, keberadaan user target, larangan mengikuti diri sendiri, serta deteksi apakah hubungan follow sudah ada atau belum. Setelah seluruh syarat terpenuhi, prosedur ini memanggil primitif graf seperti addEdge dan removeEdge pada SOCIAL_GRAPH serta memperbarui array SOCIALS agar konsisten. Pemisahan tanggung jawab ini membuat logika fitur sosial tetap modular sehingga ADT Graph bertanggung jawab terhadap representasi relasi, sedangkan fitur Social bertanggung jawab terhadap aturan bisnis dan pesan yang ditampilkan kepada pengguna.

Sementara itu, socialShowFollowing dan socialShowFollowers menyelesaikan persoalan visualisasi jaringan sosial dari sudut pandang pengguna tertentu. Kedua prosedur ini mengizinkan pemanggilan baik untuk current user maupun user lain dengan parameter username. Ketika username kosong, prosedur melakukan validasi login untuk mencegah akses data sosial tanpa autentikasi. Untuk menampilkan following, prosedur cukup menelusuri adjacency list dari vertex target pada SOCIAL_GRAPH. Untuk menampilkan followers, prosedur melakukan pemindaian menyeluruh terhadap seluruh vertex dan memeriksa apakah terdapat edge menuju target. Dalam kedua kasus tersebut, hasilnya selalu disajikan dalam bentuk daftar teks yang ramah pengguna, dengan penanganan eksplisit untuk keadaan kosong seperti tidak memiliki following atau belum mempunyai follower sama sekali. Desain fungsi-fungsi ini memastikan bahwa semua skenario sosial, mulai dari follow, unfollow, hingga melihat hubungan sosial, tertangani secara konsisten di atas satu sumber kebenaran yang sama, yaitu SOCIAL_GRAPH dan SOCIALS.

4.9 Save & Load

src/utils/Save/Save.h

```
Fungsi dan Prosedur fitur Save dalam Notasi Algoritmik
```

src/utils/Load/Load.h

```
Fungsi dan Prosedur fitur Load dalam Notasi Algoritmik
```

src/utils/LoadCommand/LoadCommand.h

```
Fungsi dan Prosedur fitur Load dalam Notasi Algoritmik
```

Penjelasan mengenai sketsa struktur data, persoalan apa yang diselesaikan, dan alasan pemilihan fungsi/prosedur yang ada pada fitur Inisialisasi.

4.10 Feed

src/utils/Feed/Feed.h

```
USE Heap, ListBerkait, MesinKata, GlobalData, Helper, User, Profil, Kreativitas

procedure commandShowFeed()
```

```
{ Menjalankan fitur utama Feed yang menampilkan daftar post
  dari akun yang diikuti oleh pengguna saat ini.

  I.S.  Sistem dapat dalam keadaan sudah login atau belum,
        pita kata berada setelah token SHOW_FEED.
  F.S.  Sistem menampilkan daftar post pada layar sesuai mode dan LIMIT
        yang diberikan, atau menampilkan pesan kesalahan yang sesuai.

  Langkah umum:
  1. Jika pengguna belum login:
       - Membersihkan sisa token pada pita input.
       - Menuliskan pesan bahwa pengguna harus login terlebih dahulu.
       - Prosedur berhenti.
  2. Membaca token berikutnya sebagai mode:
       - Jika tidak ada token, menuliskan pesan format salah.
       - Jika token bukan "LATEST" maupun "NEWEST":
           * Membersihkan sisa token pada pita input.
           * Menuliskan pesan bahwa mode tidak dikenal.
           * Prosedur berhenti.
       - Jika token = "LATEST", set isLatest ← true.
       - Jika token = "NEWEST", set isLatest ← false.
  3. Mencoba membaca token berikutnya sebagai LIMIT (opsional):
       - Jika token kosong, LIMIT dianggap tidak ada (limit ← −1).
       - Jika token tidak kosong:
           * Mengonversi token menjadi bilangan bulat limit.
           * Jika limit ≤ 0:
               · Membersihkan sisa token.
               · Menuliskan pesan bahwa LIMIT harus bilangan bulat positif.
               · Prosedur berhenti.
           * Membaca token berikutnya:
               · Jika masih ada token lain, format perintah salah,
                 bersihkan sisa token dan prosedur berhenti.
  4. Jika belum ada post sama sekali pada sistem (POST_COUNT ≤ 0):
       - Mengambil username current user.
       - Menuliskan header feed untuk pengguna tersebut.
       - Menampilkan pesan bahwa feed kosong karena belum ada post.
       - Prosedur berhenti.
  5. Menyusun array boolean followed[0..USER_COUNT-1] yang menyatakan
     apakah current user mengikuti user ke-i:
       - Jika USER_COUNT ≤ 0 atau SOCIAL_GRAPH tidak valid untuk
         current user, prosedur membebaskan memori lokal dan menulis
         pesan kesalahan data sosial, kemudian berhenti.
       - Menghitung banyaknya akun yang diikuti (followedCount).
  6. Jika followedCount = 0:
       - Menuliskan header feed untuk current user.
       - Menampilkan pesan bahwa feed kosong karena current user
         belum mengikuti siapa pun.
       - Prosedur berhenti.
  7. Mengumpulkan seluruh post yang dibuat oleh akun yang diikuti:
       - Mengalokasikan array kandidat candidPosts dan candidAuthors
         berukuran POST_COUNT.
       - Menelusuri seluruh node pada list POSTS:
           * Jika elemen bertipe Post:
               · Mencari indeks penulis di USERS berdasarkan user_id.
               · Jika indeks valid dan pengguna tersebut di-follow,
                 menambahkan pointer ke post dan indeks penulis ke
                 array kandidat.
       - Jika alokasi memori kandidat gagal:
```

```
            * Menuliskan pesan kesalahan memori dan prosedur berhenti.
        - Jika tidak ditemukan post dari akun yang diikuti (candCount = 0):
            * Menuliskan header feed.
            * Menampilkan pesan bahwa belum ada postingan dari akun
              yang diikuti.
            * Membebaskan seluruh memori lokal dan prosedur berhenti.
  8. Membangun struktur heap H:
       - createHeap(H, candCount, isLatest)
         dengan isMax = true untuk LATEST dan false untuk NEWEST.
       - Untuk setiap indeks i pada kandidat:
            * Membuat elemen heap e dengan:
                · e.key     ← nilai waktu created_at post kandidat ke-i
                  yang dikonversi ke tipe long.
                · e.dataIdx ← i.
            * Menyisipkan e ke dalam heap dengan heapPush.
  9. Menentukan banyak post yang akan ditampilkan:
       - Jika limit > 0 dan limit < candCount, set maxToShow ← limit.
       - Jika tidak, set maxToShow ← candCount.
 10. Menuliskan header feed untuk current user dengan keterangan
     mode pengurutan (LATEST atau NEWEST).
 11. Selama printed < maxToShow dan heap tidak kosong:
       - Mengambil elemen prioritas tertinggi dengan heapPop.
       - Mengakses post kandidat dan author berdasarkan dataIdx.
       - Mengonversi post_id, title, created_at, dan username author
         menjadi string.
       - Menuliskan baris:
           "<nomor>. [post_id] title (waktu) - oleh username".
       - Menambah printed.
 12. Menuliskan footer feed yang berisi petunjuk perintah lanjutan.
 13. Membebaskan seluruh resource lokal:
       - deallocateHeap(H)
       - free(followed), free(candidPosts), free(candidAuthors).
}
```

Fitur Feed bertanggung jawab untuk menyajikan daftar post yang relevan bagi pengguna berdasarkan hubungan sosial yang sudah terbentuk. Secara struktural, fitur ini berdiri di atas beberapa modul lain, yaitu daftar global USERS dan POSTS, graf sosial SOCIAL_GRAPH, serta ADT Heap dan List Berkait. Relasi follow antar pengguna diwujudkan sebagai graf berarah sehingga Feed dapat dengan cepat menentukan akun mana saja yang harus menjadi sumber posting. Kumpulan post disimpan dalam list berkait heterogen yang menyimpan elemen Post, sehingga traversal dapat dilakukan tanpa batasan kapasitas awal. Heap digunakan sebagai mesin pengurut yang efisien, di mana setiap elemen menyimpan pasangan (key, dataIdx) dengan key berupa timestamp created_at dan dataIdx menunjuk ke indeks kandidat dalam array.

Prosedur commandShowFeed dirancang sebagai satu entry point yang mengatur seluruh alur logika dari pembacaan perintah hingga penampilan hasil. Bagian awal prosedur menangani seluruh validasi masukan, mulai dari autentikasi pengguna, pemeriksaan mode LATEST atau NEWEST, pengecekan format LIMIT, hingga deteksi argumen berlebih. Hal ini memastikan bahwa kesalahan sintaks dapat terdeteksi sedini mungkin sehingga tidak mengganggu logika inti. Setelah validasi, prosedur membentuk himpunan kandidat post dengan menelusuri struktur sosial dan daftar post yang ada. Pada tahap ini, hanya post yang berasal dari akun yang diikuti current

user yang akan dipertimbangkan, sehingga Feed benar-benar mencerminkan relasi sosial pengguna.

Penggunaan Heap memberikan keuntungan kompleksitas waktu yang baik saat menyortir post berdasarkan waktu pembuatan. Alih-alih mengurutkan secara penuh melalui algoritma sorting umum setiap kali perintah dijalankan, heap menyediakan operasi push dan pop dengan kompleksitas logaritmik. Mode LATEST menggunakan max-heap sehingga elemen dengan waktu terbesar (post terbaru) muncul terlebih dahulu, sedangkan mode NEWEST menggunakan min-heap sehingga post tertua tampil di awal. Di sisi antarmuka, commandShowFeed juga memberikan pesan khusus untuk berbagai keadaan seperti belum ada post sama sekali, pengguna belum mengikuti siapa pun, atau belum ada postingan dari akun yang diikuti. Dengan demikian, fitur Feed tidak hanya efisien secara komputasional tetapi juga informatif dan konsisten dari sudut pandang pengalaman pengguna.

4.11 Trending Analysis

src/utils/Trending/Trending.h

```
USE MesinKata, Heap, ListBerkait, GlobalData, Subgroddit, Helper, Kreativitas

type KeywordFreq : < keyword           : array[0..NMax-1] of char,
                     frequency         : integer,
                     hottestPostTitle  : array[0..NMax-1] of char >

procedure commandTrending()
{ Prosedur handler untuk command TRENDING yang menganalisis
  topik trending pada subgroddit tertentu dalam rentang waktu.

  I.S. CURRENT_USER terdefinisi, sistem dalam state siap menerima input
  F.S. Menampilkan hasil analisis trending berupa top 5 keyword dengan
       frekuensi tertinggi beserta post paling populer pada subgroddit
       yang diminta dalam periode waktu yang ditentukan. }

function normalizeKeyword(word : array of char, out : array of char) → boolean
{ Menormalisasi kata menjadi lowercase dan memvalidasi panjangnya.}

procedure extractKeywords(text : array of char,
                          input/output keywordArray : pointer to array of KeywordFreq,
                          input/output keywordCount : integer,
                          input/output capacity : integer,
                          postTitle : array of char)
{ Mengekstrak semua kata dari teks dan menghitung frekuensinya dengan
  filtering stopwords.

  I.S. text terdefinisi berisi teks yang akan dianalisis
       keywordArray adalah array dinamis berisi keyword yang sudah ada
       keywordCount adalah jumlah keyword saat ini
       capacity adalah kapasitas array
       postTitle adalah judul post untuk tracking hottest topic

  F.S. keywordArray bertambah dengan kata-kata baru yang valid dari text
       Frekuensi keyword yang sudah ada akan di-increment
       keywordCount updated sesuai jumlah keyword
       capacity diresize jika diperlukan }

procedure displayTrendingResults(subgrodditName : array of char,
```

```
                                timeValue : integer,
                                timeUnit : array of char,
                                topKeywords : array of KeywordFreq,
                                topCount : integer)
{ Menampilkan hasil analisis trending dalam format TUI.

  I.S. subgrodditName, timeValue, timeUnit terdefinisi
       topKeywords berisi array keyword yang sudah diurutkan
       topCount adalah jumlah keyword yang akan ditampilkan

  F.S. Tampilan hasil analisis trending tercetak ke layar dengan format:
       - Header dengan nama subgroddit dan periode waktu
       - Top N keywords dengan frekuensi kemunculan
       - Hottest topic (post dengan keyword terbanyak)
       - Styling menggunakan border, color, dan section dividers }
```

Fitur Trending Analysis menggunakan kombinasi dari beberapa struktur data untuk menyelesaikan persoalan analisis word dan ranking. Array dinamis, kami gunakan untuk menyimpan kata kunci besert metadanya (frekuensi dan judul post terpopuler), array ini dapat membesar secara dinamis sesuai jumlah kata unik yang ditemukan. Heap (priority queue), digunakan untuk sorting dan mendapatkan top-N keywords dengan efisien. Array statis (untuk stopword list), berisi 112 kata umum dalam bahasa Inggris dan bahasa Indonesia yang tidak relevan (e.g. “the”, “is”, “dan”, “atau”, “untuk”) yang kemudian digunakan untuk filtering noises dalam ekstraksi keyword.

Fitur ini menyelesaikan persoalan content discovery dan trend analysis dalam groddit. Jadi pengguna dapat mengetahui topik apa yang sedang ramai dibahas di suatu subgroddit dalam periode waktu tertentu. Pemilihan fungsi dan prosedur dirancang untuk efisiensi dan modularitas, contohnya seperti normalizeKeyword() memastikan konsistensi data, extractKeywords() melakukan text mining dengan stopword filtering untuk menghilangkan noises, dan displayTrendingResult() memisahkan logika algoritma. Penggunaan Heap untuk ranking di sini juga jadi memberikan kompleksitas O(n log k) di mana k adalah jumlah top keywords yang diinginkan, jauh lebih efisien dibanding full sorting O(n log n). Struktur KeywordFreq yang menyimpan metadata lengkap (frekuensi dan hottest topic) bisa memberikan analisis yang lebih tanpa query ulang ke database post.

4.12 Advanced Search

src/utils/AdvancedSearch/AdvancedSearch.h

```
USE Trie, GlobalData, Helper, Kreativitas, ListBerkait

procedure commandSearchUser()
{ I.S. Pita kata berada setelah token SEARCH_USER
  F.S. Jika format salah (prefix kosong atau argumen berlebih),
       menampilkan pesan error dengan panduan penggunaan.
       Jika valid:
       - Membangun Trie dari seluruh username di USERS (lowercase).
       - Mencari semua user dengan prefix yang cocok.
       - Menampilkan daftar username hasil pencarian.
       - Membebaskan memori Trie dan array hasil. }
```

```
procedure commandSearchPost()
{ I.S. Pita kata berada setelah token SEARCH_POST
  F.S. Jika format salah, menampilkan pesan error.
       Jika valid:
       - Membangun Trie dari judul seluruh Post (lowercase).
       - Mencari semua post dengan prefix judul yang cocok.
       - Menampilkan daftar post dalam format:
         "<no>. [<nama_subgroddit>] <judul_post>"
       - Membebaskan memori Trie dan array hasil. }


procedure commandSearchSubgroddit()
{ I.S. Pita kata berada setelah token SEARCH_SUBGRODDIT
  F.S. Jika format salah, menampilkan pesan error.
       Jika valid:
       - Membangun Trie dari nama seluruh SubGroddit (lowercase).
       - Mencari semua subgroddit dengan prefix nama yang cocok.
       - Menampilkan daftar nama subgroddit hasil pencarian.
       - Membebaskan memori Trie dan array hasil. }
```

Fitur Advanced Search menyediakan pencarian berbasis prefix untuk tiga entitas: User, Post, dan SubGroddit. Setiap command handler membangun Trie secara dinamis dari data global, melakukan pencarian O(k) di mana k adalah panjang prefix, lalu menampilkan hasil. Pencarian bersifat case-insensitive dengan mengonversi semua key dan prefix ke lowercase. Pemilihan Trie terpisah untuk setiap entitas memungkinkan pencarian yang terfokus dan efisien. Memory management dilakukan dengan cermat melalui pemanggilan freeTrie() dan free() untuk mencegah memory leak.

4.13 Following Recommendation

src/utils/FollRec/FollRec.h

```
procedure commandFriendRecommendation()
{ Menjalankan fitur rekomendasi teman berbasis graf sosial.

  I.S.  Sistem dapat dalam keadaan sudah login atau belum,
        pita kata berada setelah token FRIEND_RECOMMENDATION.
  F.S.  Jika seluruh prasyarat terpenuhi, sistem menampilkan
        maksimal 10 rekomendasi akun yang mungkin ingin di-follow
        oleh pengguna saat ini, diurutkan menurun berdasarkan
        banyaknya mutual connections. Dalam kasus tertentu, sistem
        menampilkan pesan kesalahan atau pesan bahwa tidak ada
        rekomendasi yang dapat ditampilkan.

  Langkah umum:
  1. Jika pengguna belum login:
       - Membersihkan sisa token pada pita input.
       - Menuliskan pesan bahwa pengguna harus login terlebih dahulu.
       - Prosedur berhenti.
  2. Membaca token berikutnya:
       - Jika panjang token tidak nol, berarti ada argumen tambahan.
         Format perintah salah sehingga:
           * Semua token sisa dibersihkan.
```

```
          * Dituliskan pesan bahwa format FRIEND_RECOMMENDATION salah.
          * Prosedur berhenti.
      - Jika tidak ada token lain, perintah dianggap valid.
3. Mengambil n ← USER_COUNT dan src ← CURRENT_USER_INDEX.
      - Jika n ≤ 1, atau src di luar rentang 0..n-1, atau src bukan
        vertex valid pada SOCIAL_GRAPH:
          * Mengambil username current user.
          * Menuliskan pesan bahwa tidak ada rekomendasi teman
            yang dapat ditampilkan.
          * Prosedur berhenti.
4. Mengalokasikan beberapa array berukuran n:
      - depth[0..n-1]   : integer
      - visited[0..n-1] : boolean
      - followed[0..n-1]: boolean
      - mutual[0..n-1]  : integer
      - queue[0..n-1]   : integer
    Jika salah satu alokasi gagal:
      - Menuliskan pesan kesalahan memori.
      - Membebaskan seluruh alokasi yang sudah berhasil.
      - Prosedur berhenti.
5. Inisialisasi:
      - Untuk setiap i di 0..n-1:
          * depth[i]   ← -1
          * visited[i] ← false
          * mutual[i]  ← 0
      - Memanggil prosedur pembantu buildFollowedArray untuk
        mengisi followed[i] = true jika current user mengikuti
        pengguna ke-i, false jika tidak. Jika SOCIAL_GRAPH tidak valid
        untuk src, prosedur pembantu hanya mengisi false untuk semua.
6. Melakukan penelusuran BFS pada SOCIAL_GRAPH mulai dari src:
      - Inisialisasi head ← 0, tail ← 0.
      - queue[tail] ← src; tail ← tail + 1.
      - visited[src] ← true, depth[src] ← 0.
      - Selama head < tail:
          * u ← queue[head]; head ← head + 1.
          * Jika depth[u] ≥ MAX_REC_DEPTH (konstanta 3),
            lanjutkan ke iterasi berikutnya.
          * Jika u bukan vertex valid, lanjutkan.
          * Menelusuri adjacency list SOCIAL_GRAPH.adj[u]:
              · Untuk setiap edge (u → v):
                  - Jika 0 ≤ v < n dan visited[v] = false:
                      visited[v] ← true
                      depth[v] ← depth[u] + 1
```

Fitur Following Recommendation memanfaatkan struktur graf sosial dan heap untuk memberikan saran akun yang relevan bagi pengguna. Graf sosial disimpan dalam SOCIAL_GRAPH sebagai graf berarah dengan adjacency list, di mana setiap vertex merepresentasikan satu pengguna dan edge (u → v) menyatakan bahwa u mengikuti v. Di atas struktur ini, fitur rekomendasi berupaya menjawab persoalan: “siapa saja pengguna yang mungkin menarik untuk di-follow, berdasarkan jaringan pertemanan tidak langsung dan jumlah koneksi bersama (mutual connections)”. Untuk itu, prosedur commandFriendRecommendation membangun beberapa array bantu seperti depth, visited, followed, mutual, dan queue untuk mengeksekusi algoritma BFS dan perhitungan mutual secara efisien.

Prosedur dimulai dengan validasi autentikasi dan format perintah sehingga kesalahan seperti argumen berlebih dapat segera ditangani. Selanjutnya, algoritma melakukan BFS dari current user hingga kedalaman maksimum tertentu (MAX_REC_DEPTH = 3) untuk menemukan pengguna yang berada pada jarak dua atau tiga langkah dalam graf sosial. Hanya pengguna dengan kedalaman tersebut, belum di-follow oleh current user, dan memiliki setidaknya satu mutual connection yang dianggap sebagai kandidat. Untuk setiap kandidat v, mutual connections dihitung sebagai irisan antara himpunan akun yang diikuti current user dan akun yang diikuti oleh v. Nilai mutual ini lalu digunakan sebagai key dalam sebuah max-heap sehingga rekomendasi dapat diurutkan menurun berdasarkan kedekatan jaringan sosial.

Penggunaan heap sebagai struktur prioritas memungkinkan pemilihan hingga sepuluh kandidat terbaik tanpa perlu mengurutkan seluruh pengguna secara global. Hal ini mengurangi biaya komputasi ketika jumlah pengguna besar, karena hanya kandidat dengan mutual connections positif yang masuk ke heap. Di sisi lain, pembatasan kedalaman BFS menjaga agar rekomendasi tetap relevan dan tidak terlalu jauh dari lingkaran sosial pengguna. Prosedur juga menangani secara eksplisit kasus tepi seperti tidak adanya kandidat atau kegagalan alokasi memori, dengan memberikan pesan yang jelas kepada pengguna. Dengan demikian, fitur Following Recommendation memadukan konsep graf, BFS terbatas kedalaman, dan priority queue untuk menyajikan rekomendasi teman yang informatif sekaligus efisien.

4.14 Content Moderation

src/utils/ContentModeration/ContentModeration.h

```
USE GlobalData, Helper, User, Profil, Voting

type BlackList : < words: array[0..MAX_BLACKLIST] of string, count: integer >

Blacklist GLOBAL_BLACKLIST

function content_moderation_init(pathConf : string) → boolean
{
I.S. pathConf menunjuk ke file konfigurasi yang dapat diakses atau tidak ada.
F.S. Jika file konfigurasi valid dan kata terlarang berhasil dimuat:
• GLOBAL_BLACKLIST terisi dengan kata-kata terlarang.
• Fungsi mengembalikan true.
Jika terjadi kesalahan pembacaan atau parsing:
• GLOBAL_BLACKLIST dikembalikan ke kondisi default (count = 0).
• Fungsi mengembalikan false.
}

function LoadBlacklistJSON(filepath : string) → integer
{
I.S. filepath adalah path ke file JSON yang ingin dibaca.
F.S. Jika pembacaan dan parsing berhasil:
• GLOBAL_BLACKLIST.words diisi dengan kata terlarang (semua disimpan
sesuai batas MAX_BLACKLIST dan MAX_WORD_LEN).
• GLOBAL_BLACKLIST.count di-set sesuai jumlah kata yang dimuat.
• Fungsi mengembalikan jumlah kata yang dimuat (>= 0).
Jika terjadi kesalahan (file tidak ada, format JSON invalid, overflow):
• GLOBAL_BLACKLIST tidak berubah atau di-reset ke default.
• Fungsi mengembalikan -1 sebagai tanda kegagalan.
}
```

```
function CheckBlacklistedContent(text : string,
out foundWords : array of string,
out foundCount : integer) → integer
{
I.S. GLOBAL_BLACKLIST terdefinisi (mungkin kosong). text berisi string yang
akan diperiksa; foundWords disediakan oleh pemanggil dengan kapasitas.
F.S. Jika teks mengandung satu atau lebih kata yang ada di GLOBAL_BLACKLIST:
• foundWords diisi dengan daftar kata terlarang yang muncul (tanpa duplikat,
hingga kapasitas MAX_FOUND).
• foundCount berisi jumlah kata yang ditemukan.
• Fungsi mengembalikan 1 (menandakan ada kata terlarang).
Jika tidak ditemukan kata terlarang:
• foundCount = 0.
• Fungsi mengembalikan 0.
Jika terjadi kesalahan input (mis. null pointer) atau kapasitas terlampaui:
• foundCount = 0 atau sesuai kondisi parsial.
• Fungsi mengembalikan -1 (menandakan error).
}

procedure ToLowercaseInplace(input/output s : string)
{
I.S. s adalah string yang valid dan dapat dimodifikasi in-place.
F.S. Semua karakter alfabet pada s diubah menjadi huruf kecil.
Digunakan untuk normalisasi sebelum pencocokan kata terlarang.
}
```

Fitur inisialisasi pada modul Content Moderation menggunakan sebuah struktur data Blacklist, yang berisi *array of string* untuk menyimpan kata-kata terlarang serta sebuah *counter* jumlah kata yang dimuat. Representasi berupa *array of string* dipilih karena ukuran daftar kata terlarang bersifat terbatas dan statis, sehingga tidak diperlukan struktur data dinamis yang lebih kompleks. Dengan bentuk seperti ini, setiap kata dapat diakses melalui indeks secara langsung, membuat proses pencocokan kata lebih sederhana dan efisien selama pemeriksaan konten.

Persoalan utama yang ingin diselesaikan oleh fitur ini adalah proses pemuatan daftar kata terlarang dari file konfigurasi eksternal dan menyiapkan data tersebut dalam bentuk internal yang mudah digunakan untuk validasi konten. Inisialisasi diperlukan agar seluruh modul yang membutuhkan pengecekan konten sensitif dapat bekerja berdasarkan satu sumber kebenaran yang konsisten. Tanpa tahap inisialisasi yang benar, sistem tidak dapat melakukan deteksi kata terlarang, dan fitur moderasi konten akan gagal berfungsi.

Pemilihan fungsi dan prosedur pada modul ini mengikuti alur kebutuhan tersebut. Fungsi content_moderation_init bertanggung jawab memulai keseluruhan mekanisme moderasi dengan memanggil proses pemuatan blacklist dan memastikan struktur global berada dalam kondisi valid. Fungsi LoadBlacklistJSON menangani pembacaan file JSON dan mengisi GLOBAL_BLACKLIST dengan kata-kata yang ditemukan dalam file, sehingga modul lain tidak perlu mengetahui detail *parsing*. Fungsi CheckBlacklistedContent menjalankan logika utama moderasi dengan memeriksa apakah suatu teks mengandung kata terlarang yang telah dimuat sebelumnya. Prosedur ToLowercaseInplace digunakan sebagai tahap normalisasi agar proses pencocokan kata tidak dipengaruhi perbedaan kapitalisasi. Dengan kombinasi fungsi tersebut, fitur inisialisasi pada modul Content Moderation memastikan bahwa seluruh sistem moderasi memiliki data blacklist yang lengkap dan siap digunakan secara deterministik.

4.15 Data Security

src/utils/Security/Security.h

```
USE GlobalData, Helper, MesinKata

procedure security_init(input config_path : string) → boolean
{
I.S. config_path menunjuk ke lokasi berkas konfigurasi keamanan.
F.S. Jika file konfigurasi ditemukan dan berhasil dibaca, status fitur keamanan
(password hashing, file encryption, dan seed) dimuat ke variabel global.
Mengembalikan true jika berhasil, false jika gagal.
}

function password_hashing_enabled() → boolean
{
I.S. Status password hashing tersimpan dalam variabel global.
F.S. Mengembalikan true jika password hashing aktif, false jika tidak aktif.
}

function file_encryption_enabled() → boolean
{
I.S. Status enkripsi file tersimpan dalam variabel global.
F.S. Mengembalikan true jika enkripsi file aktif, false jika tidak aktif.
}

procedure enable_password_hashing() → boolean
{
I.S. Password hashing mungkin aktif atau belum aktif.
F.S. Jika sebelumnya nonaktif, password hashing diaktifkan dan seluruh password
plaintext di memori dimigrasikan menggunakan hash_password.
Mengembalikan true.
}

procedure set_file_encryption(input on : boolean) → boolean
{
I.S. Parameter on valid.
F.S. Status enkripsi file diperbarui sesuai nilai on.
Mengembalikan true.
}

procedure security_set_seed(input seed : integer)
{
I.S. seed adalah bilangan 32-bit valid.
F.S. Nilai seed global diperbarui untuk digunakan dalam proses keystream.
}

function security_get_seed() → integer
{
I.S. Nilai seed global terdefinisi.
F.S. Mengembalikan nilai seed yang sedang digunakan.
}

function hash_password(input pw : string) → integer
{
I.S. pw adalah password plaintext.
F.S. Mengembalikan hash 32-bit hasil algoritma FNV-1a setelah:
1. Inisialisasi offset basis.
2. Melakukan XOR dengan setiap byte pw.
```

```
3. Melakukan perkalian dengan FNV prime.
}

procedure reset_keystream()
{
I.S. Seed global terdefinisi.
F.S. State internal keystream di-reset ke nilai seed.
}

procedure crypt_buffer(input/output buf : array of byte, input n : integer)
{
I.S. buf berisi plaintext atau ciphertext, keystream telah di-reset.
F.S. Setiap byte buf[i] diubah dengan XOR terhadap keystream yang dihasilkan LCG:
• menghasilkan ciphertext jika input plaintext
• menghasilkan plaintext jika input ciphertext
}

function read_encrypted_file(input path : string,
output out_len : integer,
output was_encrypted : boolean)
→ array of byte
{
I.S. path menunjuk ke file yang ingin dibaca.
F.S. Jika file memiliki header magic "GREN" dan versi valid:
• Payload dibaca dan didekripsi menggunakan crypt_buffer.
• was_encrypted = true.
• out_len berisi panjang plaintext.
• Mengembalikan buffer plaintext.

 Jika tidak memiliki header:
 • File dianggap tidak terenkripsi.
 • was_encrypted = false.
 • out_len berisi ukuran file.
 • Mengembalikan isi file apa adanya.
}

procedure write_encrypted_file(input path : string,
input buf : array of byte,
input len : integer)
→ boolean
{
I.S. buf berisi data yang ingin disimpan.
F.S. Jika enkripsi file aktif:
• Tulis header: magic "GREN" dan nomor versi.
• Enkripsi buf menggunakan crypt_buffer.
• Simpan ciphertext ke file.

 Jika enkripsi nonaktif:
 • buf ditulis langsung tanpa modifikasi.

 Mengembalikan true jika penulisan berhasil, false jika gagal membuka file.
}

procedure save_security_conf(input path : string) → boolean
{
I.S. path menunjuk ke lokasi penyimpanan security.conf.
F.S. File konfigurasi disimpan berisi status password hashing,
status file encryption, dan nilai seed.
```

```
Mengembalikan true jika file berhasil ditulis, false jika gagal.
}

procedure handleSecurityCommand()
{
I.S. MesinKata siap membaca input pengguna.
F.S. Perintah terkait keamanan diproses sesuai input:
• Mengaktifkan password hashing.
• Mengatur status enkripsi file.
• Mengubah nilai seed.
Output pesan status sesuai operasi.
}

procedure migrate_passwords_in_memory()
{
I.S. Struktur data user berisi password plaintext dan fitur hashing baru saja diaktifkan.
F.S. Semua password plaintext diubah menjadi hash menggunakan hash_password.
Password plaintext tidak lagi tersimpan di memori.
}
```

Fitur inisialisasi pada modul Security menggunakan beberapa variabel global sederhana untuk menyimpan status keamanan aplikasi, yaitu status *password hashing*, status *file encryption*, serta *seed* untuk pembangkit *keystream*. Struktur data ini dipilih karena konfigurasi keamanan bersifat *global* dan harus dapat diakses oleh seluruh modul lain tanpa perlu melakukan passing parameter yang berulang. Selain itu, modul ini juga menggunakan array karakter global_security_conf_path sebagai penampung lokasi berkas konfigurasi, serta variabel *seed* dan *state internal keystream* yang diperlukan untuk membangun proses enkripsi XOR berbasis LCG. Pendekatan berbasis variabel global ini sesuai dengan sifat konfigurasi program yang hanya perlu dibaca sekali saat inisialisasi, lalu dipertahankan selama aplikasi berjalan.

Persoalan utama yang ingin diselesaikan oleh fitur ini adalah bagaimana memuat konfigurasi keamanan dari file eksternal, menyiapkan nilai awal untuk fitur *password hashing* dan *file encryption*, serta memastikan bahwa status keamanan konsisten sebelum modul lain bekerja. Inisialisasi diperlukan agar aplikasi mengetahui apakah password harus disimpan dalam bentuk hash, apakah file *save* harus dienkripsi atau tidak, dan *seed* apa yang digunakan untuk menghasilkan *keystream* enkripsi. Jika konfigurasi tidak dimuat dengan benar, modul *hashing*, migrasi *password*, dan enkripsi file tidak dapat berfungsi secara deterministik.

Pemilihan fungsi dan prosedur pada modul ini berhubungan langsung dengan kebutuhan tersebut. Fungsi security_init menangani proses awal, yaitu membaca security.conf dan mengisi seluruh variabel global yang diperlukan. Fungsi password_hashing_enabled dan file_encryption_enabled disediakan agar modul lain dapat memeriksa status keamanan tanpa harus mengetahui detail implementasi. Prosedur enable_password_hashing digunakan untuk mengaktifkan *hashing* dan melakukan migrasi seluruh password *plaintext* yang sudah terlanjur dimuat ke memori. Prosedur set_file_encryption mengubah status enkripsi file sesuai konfigurasi. Fungsi security_set_seed dan security_get_seed mengelola *seed* yang menjadi dasar *keystream*. Fungsi hash_password menyediakan algoritma *hashing* FNV-1a yang digunakan ketika *hashing* password diaktifkan, sedangkan reset_keystream dan crypt_buffer menyediakan mekanisme enkripsi dan dekripsi file. Terakhir, prosedur save_security_conf menyimpan ulang

konfigurasi ke file, dan handleSecurityCommand memungkinkan pengguna mengatur parameter keamanan melalui CLI. Kombinasi fungsi dan prosedur ini memastikan modul keamanan dapat diinisialisasi dengan benar, memperbarui konfigurasi ketika diperlukan, dan menyediakan antarmuka yang konsisten bagi seluruh sistem.

4.16 Kreativitas

src/utils/Kreativitas/Kreativitas.h

```
{ Color Constants (ANSI Escape Codes):
  RESET, BOLD, DIM
  BLACK, RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE
  BOLD_RED, BOLD_GREEN, BOLD_YELLOW, BOLD_BLUE, BOLD_MAGENTA,
  BOLD_CYAN, BOLD_WHITE
  BG_BLACK, BG_RED, BG_GREEN, BG_YELLOW, BG_BLUE, BG_MAGENTA,
  BG_CYAN, BG_WHITE }

{ Box Drawing Characters (Unicode):
  BOX_TL (┌), BOX_TR (┐), BOX_BL (└), BOX_BR (┘)
  BOX_H (─), BOX_V (│), BOX_VR (├), BOX_VL (┤)
  BOX_HU (┴), BOX_HD (┬), BOX_CROSS (┼)
  DBOX_TL (╔), DBOX_TR (╗), DBOX_BL (╚), DBOX_BR (╝)
  DBOX_H (═), DBOX_V (║), DBOX_VR (╠), DBOX_VL (╣) }

procedure clearScreen()
{ I.S. Terminal dalam keadaan sembarang
  F.S. Layar terminal dibersihkan (clear screen).

  Implementasi:
    - Windows: system("cls")
    - Unix/Linux/macOS: system("clear") }

procedure printHorizontalLine(input width : integer,
                              input left : string,
                              input middle : string,
                              input right : string)
{ I.S. Parameter terdefinisi untuk membuat garis horizontal
  F.S. Garis horizontal dicetak dengan karakter border yang ditentukan.

  Format:
    <left><middle x (width-2)><right>

  Contoh:
    printHorizontalLine(40, "┌", "─", "┐")
    Output: ┌──────────────────────────────────────┐ }

procedure printDoubleLine(input width : integer)
{ I.S. width adalah lebar garis yang diinginkan
  F.S. Garis double-line horizontal tercetak menggunakan karakter ╔╗.

  Shortcut untuk printHorizontalLine dengan double-box characters }

procedure printSingleLine(input width : integer)
{ I.S. width adalah lebar garis yang diinginkan
  F.S. Garis single-line horizontal tercetak menggunakan karakter ┌┐.

  Shortcut untuk printHorizontalLine dengan single-box characters }
```

```
procedure printBoxedText(input text : string, input width : integer)
{ I.S. text dan width terdefinisi
  F.S. Text dicetak dalam box dengan padding center-aligned.

  Format:
    │<padding><text><padding>│

  Contoh:
    printBoxedText("WELCOME", 20)
    Output: │      WELCOME      │ }

procedure printSectionHeader(input emoji : string, input title : string)
{ I.S. emoji dan title terdefinisi (emoji dapat berupa string kosong)
  F.S. Section header dicetak dengan format colored dan underline.

  Format:
    <emoji> <title> (cyan bold)
    ──────────────────── (dim)

  Contoh:
    printSectionHeader("", "ACCOUNT INFORMATION")
    Output:
      ACCOUNT INFORMATION (cyan bold)
      ──────────────────────── (dim) }

procedure printSectionDivider()
{ I.S. Sistem siap mencetak divider
  F.S. Divider horizontal tercetak untuk memisahkan sections.

  Format:
    ├──────────────────────────────────────────────┤ (60 chars width, dim) }

procedure printBreadcrumb(input path : string)
{ I.S. path adalah string path navigasi
  F.S. Breadcrumb navigation tercetak dengan warna bold blue.

  Format:
    Main Menu > Category > Command (bold blue)

  Contoh:
    printBreadcrumb("Home > Profile > View User") }

procedure printStatusBar(input username : string,
                         input karma : integer,
                         input location : string)
{ I.S. Parameter status bar terdefinisi
  F.S. Status bar tercetak dengan border box menampilkan username,
       karma, dan location. Jika username kosong, tampilkan "Guest Mode".

  Format dengan user:
    ┌──────────────────────────────────────────┐
    │ kevingansss | Karma: 150 | Main Menu     │
    └──────────────────────────────────────────┘

  Format guest:
    ┌──────────────────────────────────────────┐
    │ Guest Mode | Main Menu                   │
```

```
    └──────────────────────────────────────────────────┘ }

procedure printSuccess(input message : string)
{ I.S. message terdefinisi
  F.S. Success message tercetak dengan prefix [SUCCESS] berwarna green.

  Format:
    [SUCCESS] <message> }

procedure printError(input message : string)
{ I.S. message terdefinisi
  F.S. Error message tercetak dengan prefix [ERROR] berwarna red.

  Format:
    [ERROR] <message> }

procedure printWarning(input message : string)
{ I.S. message terdefinisi
  F.S. Warning message tercetak dengan prefix [WARNING] berwarna yellow.

  Format:
    [WARNING] <message> }

procedure printInfo(input message : string)
{ I.S. message terdefinisi
  F.S. Info message tercetak dengan prefix [INFO] berwarna cyan.

  Format:
    [INFO] <message> }

procedure showProgress(input message : string, input percent : integer)
{ I.S. message dan percent (0-100) terdefinisi
  F.S. Progress bar tercetak dengan fill indicator dan percentage.

  Format:
    <message> [████████████░░░░░░░░░░░░░░░░░░░░] 60%

  Karakter:
    - Filled: ▉ (solid block)
    - Empty: ▒ (light shade)

  Bar width: 40 characters
  Update: carriage return (\r) untuk animasi in-place }

procedure spinnerAnimation(input message : string, input steps : integer)
{ I.S. message dan steps terdefinisi
  F.S. Spinner animation tercetak dengan rotating character selama
       steps iterasi (setiap step = 100ms), diakhiri dengan [DONE].

  Spinner characters (Unicode Braille):
    ⠋ ⠙ ⠹ ⠸ ⠼ ⠴ ⠦ ⠧ ⠇ ⠏ (10 frames)

  Format:
    ⠋ <message>  (rotating)
    [DONE] <message> (final)

  Timing: usleep(100000) = 100ms per frame }
```

```
procedure loadingBar(input length : integer, input duration : integer)
{ I.S. length adalah panjang bar, duration adalah delay per step (microseconds)
  F.S. Loading bar tercetak dengan animasi filling dari 0% sampai 100%.

  Format:
    [████░░░░░░░░░░░░░░░░░░░░] 40%
    [████████████████████████] 100% (green)

  Update: carriage return untuk smooth animation }

procedure loadingBarSmooth(input length : integer,
                           input delayMicroseconds : integer)
{ I.S. length dan delay terdefinisi
  F.S. Smooth loading bar dengan 4 phases per block tercetak.

  Phases (Unicode shading):
    ░ (light) → ▒ (medium) → ▓ (dark) → █ (solid)

  Total frames: length × 4
  Memberikan efek animasi lebih smooth }

procedure printMenu()
{ I.S. Sistem siap menampilkan help menu
  F.S. Complete command center menu tercetak dengan kategorisasi commands.

  Struktur Menu:
    - Header: "GRODDIT COMMAND CENTER"
    - 10 Categories:
      1. AUTHENTICATION
      2. NAVIGATION & READING
      3. CONTENT MANAGEMENT
      4. VOTING SYSTEM
      5. SOCIAL FEATURES
      6. DISCOVERY & SEARCH
      7. DATA MANAGEMENT
      8. SECURITY & PRIVACY
      9. SYSTEM
      10. DEBUG (dimmed)
    - Footer: Usage tip

  Format per command:
    <command name> (25 chars) │ <description>

  Active commands: BOLD_WHITE
  Inactive/debug commands: DIM }

procedure printCommandHelp(input command : string,
                           input description : string,
                           input isActive : integer)
{ I.S. Parameter command help terdefinisi
  F.S. Single command entry tercetak dengan format aligned.

  Format:
    <command (25 chars)> │ <description>

  Color:
    - isActive = 1: BOLD_WHITE (active command)
    - isActive = 0: DIM (inactive/debug command) }
```

```
procedure printBanner()
{ I.S. Sistem siap menampilkan banner
  F.S. ASCII art banner "GRODDIT" tercetak (22 lines).

  Banner menggunakan Unicode block characters dan gradasi:
    ▀ ▒ ▓ ░ untuk membuat logo 3D effect }

procedure printInputPrompt(input context : string)
{ I.S. context adalah string konteks input
  F.S. Styled input prompt tercetak dengan box drawing.

  Format:
    ┌─ <context>
    │
    └─> ▀▒▓░ <cursor>

  Visual effect: animated cursor dengan gradient blocks }
```

Penjelasan mengenai sketsa struktur data, persoalan apa yang diselesaikan, dan alasan pemilihan fungsi/prosedur yang ada pada fitur Inisialisasi.

5 Algoritma-Algoritma Menarik

5.1 Algoritma Hashing FNV-1a 32-bit

```
uint32_t hash_password(const char *pw)
{
    uint32_t hash = 2166136261u;
    for (size_t i = 0; pw[i] != '\0'; i++)
    {
        hash ^= (uint8_t)pw[i];
        hash *= 16777619u;
    }
    return hash;
}
```

Algoritma ini digunakan pada *Data-Security* bagian *Password Hashing*. Algoritma FNV (Fowler–Noll–Vo) ini bekerja dengan cara:

1. Memulai hash dengan sebuah offset basis (nilai awal konstan).
2. Untuk setiap byte karakter pada input:
    a. Hash di-XOR dengan byte tersebut.
    b. Hasilnya dikalikan dengan sebuah FNV prime.
3. Proses ini diulang hingga seluruh karakter diproses.
4. Nilai akhir hash dipakai sebagai hasil.

Hash yang dihasilkan merupakan bilangan 32-bit yang sensitif terhadap perubahan kecil pada input.

Algoritma FNV-1a menarik karena sangat sederhana, tetapi tetap menghasilkan nilai hash yang berbeda-beda dengan baik untuk berbagai input. Meskipun hanya memakai operasi XOR dan perkalian, algoritma ini sangat cepat dan mudah diimplementasikan. Karena kombinasi antara kesederhanaan dan kecepatannya, FNV-1a banyak dipakai untuk kebutuhan hashing dasar di berbagai program.

5.2 LCG + XOR

```
void reset_keystream(void)
{
    g_keystream_state = g_seed;
}
static uint8_t next_keystream_byte(void)
{
    // LCG params (glibc-like)
    g_keystream_state = (uint32_t)(g_keystream_state * 1664525u + 1013904223u);
    return (uint8_t)(g_keystream_state >> 24);
}

void crypt_buffer(uint8_t *buf, size_t n)
{
    reset_keystream();
    for (size_t i = 0; i < n; i++)
    {
        buf[i] ^= next_keystream_byte();
    }
}
```

Algoritma ini digunakan pada *Data-Security* bagian *File Encryption* dan *File Decryption*. Algoritma LCG (*Linear Congruential Generator*) + XOR ini bekerja dengan cara:

1. Inisialisasi seed (nilai awal) yang disimpan di `g_seed` (dapat diubah sesuai input pengguna).
2. Saat mulai enkripsi/dekripsi, state keystream di-set ke seed: `state = g_seed`.
3. Untuk menghasilkan setiap byte keystream:
    a. Perbarui state dengan LCG: `state = state * 1664525 + 1013904223`.
    b. Ambil satu byte keystream dari state: `keystream_byte = state >> 24`.
4. Untuk setiap byte data (plaintext saat enkripsi / ciphertext saat dekripsi):
    a. Ambil `keystream_byte` sesuai langkah (3).
    b. XOR-kan byte data dengan `keystream_byte`: `out[i] = in[i] ^ keystream_byte`.
5. Untuk penulisan file terenkripsi program menulis header terlebih dahulu: 4 byte magic `"GREN"` diikuti 1 byte version. Setelah itu ditulis ciphertext hasil XOR.
6. Untuk membaca file: cek header `"GREN"`+version; jika ada, ulangi langkah 2–4 untuk menghasilkan keystream yang sama dan XOR ciphertext → menghasilkan plaintext.

enkripsi dan dekripsi identik karena operasi XOR bersifat involutif (`P^K^K=P`). Seed `g_seed` menentukan keystream; tanpa seed yang sama, deskripsi gagal.

Algoritma LCG + XOR ini menarik karena mampu menghasilkan proses enkripsi yang sangat ringan dan mudah dipahami. Dengan hanya memanfaatkan generator angka semu (LCG)

dan operasi XOR, algoritma ini dapat membuat data berubah menjadi bentuk yang tidak terbaca tanpa menambah kompleksitas besar pada program. Selain itu, algoritma ini cepat, deterministik, dan mudah diimplementasikan, sehingga cocok untuk kebutuhan enkripsi sederhana.

6 Data Test

6.1 Config-1

config-1

Data test config-1 merupakan kumpulan berkas konfigurasi yang digunakan untuk menguji proses inisialisasi serta seluruh fitur utama dan bonus pada aplikasi Credit. Seluruh berkas CSV merepresentasikan state awal sistem secara lengkap, mencakup pengguna, struktur forum, konten, interaksi sosial, aktivitas voting, serta seluruh bonus. Berikut rincian fungsi masing-masing berkas terhadap pengujian.

1. **user.csv : Konfigurasi Dasar Pengguna Sistem**
   Memuat 7 entri pengguna dengan variasi:
   - tingkat karma berbeda-beda,
   - waktu pembuatan akun berjenjang,
   - kombinasi username unik.

   Fungsi dalam pengujian:
   - Memverifikasi proses load pengguna, autentikasi, dan penyusunan profil.
   - Menguji modul Social (FOLLOW/FOLLOWERS) dan keterhubungannya dengan indeks pengguna.
   - Menjadi referensi utama untuk relasi dan pemetaan ID pada post, komentar, dan voting.

2. **subgroddit.csv : Struktur Subgroddit**
   Berisi 4 subgroddit dengan tingkat aktivitas yang tidak merata. Fungsi dalam pengujian:
   - Menguji penampilan daftar subgroddit.
   - Menguji fitur navigasi dan tampilan post per subgroddit.
   - Menguji kasus tepi subgroddit tanpa post.

3. **post.csv : Konten Utama Platform**
   Memuat 10 post dengan distribusi:
   - mayoritas pada satu subgroddit (menguji filtering),
   - variasi atribut waktu (mendukung pengujian Feed),
   - nilai upvote/downvote yang bervariasi.

   Fungsi dalam pengujian:
   - Memastikan post dapat diload sebagai node List Berkait.
   - Menguji tampilan post, pemilihan post ID, dan fitur voting.
   - Menjadi dataset utama untuk Feed (LATEST/NEWEST), Trending Analysis, dan analisis rekomendasi berbasis mutual.

4. **comment.csv : Struktur Komentar Bertingkat**
   Memuat komentar dengan relasi *parent–child*. Fungsi dalam pengujian:
   - Menguji inisialisasi tree komentar.
   - Menguji mekanisme penampilan thread dan cascading delete.
   - Menguji vote pada komentar serta integritas referensi parent_comment_id.

5. **social.csv : Relasi Follow di lingkungan Social**
   Memuat 4 edge *follow* antar user. Fungsi dalam pengujian:
   - Menguji pembentukan graf SOCIAL_GRAPH dengan adjacency list.
   - Menguji fitur FOLLOW, UNFOLLOW, FOLLOWING, dan FOLLOWERS.
   - Menjadi basis untuk fitur Following Recommendation (kedalaman BFS dan mutual connections).

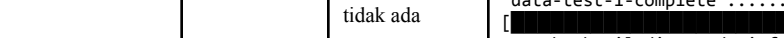
6. **voting.csv : Aktivitas Voting**
   Memuat 20 entri voting pada post dan komentar. Fungsi dalam pengujian:
   - Menguji konsistensi update upvotes/downvotes saat proses load.
   - Menjadi data dasar untuk:
     - perhitungan karma pengguna,
     - sorting HOT pada subgroddit, dan
     - analisis trending.
7. **security.conf: konfigurasi Data-Security**
   Memuat 3 baris konfigurasi
   - PASSWORD_HASHING=<value> dengan value bisa 1 (aktif) atau 0 (non-aktif)
   - FILE_ENCRYPTION=<value> dengan value bisa 1 (aktif) atau 0 (non-aktif)
   - ENC_SEED=<value> dengan value adalah angka positif berapapun
8. **blacklisted_words.json: konfigurasi Content-Moderation**
   Memuat List Kata yang Tidak Diperbolehkan Pada POST dan COMMENT dengan format json:
   - Contoh isi:

```
{
  "blacklisted_words": [
    "bodoh",
    "sampah",
    "jelek",
    "kasar"
  ]
}
```

7 Test Script

Tabel 7.1 Tabel Pengujian Fitur Wajib 1: Inisialisasi

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar

1	Menguji apakah load dari file konfigurasi berhasil	1. Masukkan nama folder	config-1;	Terdapat peringatan bahwa nama folder yang dimasukkan tidak ada	`Selamat datang di Groddit: CREDIT!`
					`Masukkan folder konfigurasi untuk dimuat:`
					`data-test-1;`
					
					`Memuat data dari folder 'data-test-1-complete'.....`
					`[`  `]`
					`Data berhasil dimuat dari folder 'data-test-1'!`

2	Menguji apakah load dari file konfigurasi	1. Masukkan nama folder yang tidak valid	data-test-999;	Terdapat peringatan bahwa nama folder yang	`Selamat datang di Groddit: CREDIT!`
					`Masukkan folder konfigurasi untuk dimuat:`
					`data-test-999;`
					
					`[Error] Folder 'data-test-999' tidak ditemukan.`
					`Silakan masukkan folder yang tepat.`

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	gagal ditangani			dimasukkan tidak ada	

Tabel 7.2 Tabel Pengujian Fitur Wajib 2: Perintah

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Selalu mengakhiri masukan perintah dengan ;	1. Masukkan nama folder	config-1;	Perintah tidak dieksekusi hingga muncul karakter MARK, yaitu ‘;’	Selamat datang di Groddit: CREDIT! Masukkan folder konfigurasi untuk dimuat: **data-test-1** **; blankspace** Memuat data dari folder 'data-test-1-complete'..... [▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓] Data berhasil dimuat dari folder 'data-test-1'!
2	Melakukan validasi perintah	1. Masukkan nama folder 2. Masukkan perintah yang tidak valid	config-1;	Terdapat peringatan bahwa masukan perintah tidak valid	▓▓▓▓▓ PROFILE bob 12345678; Format perintah PROFILE salah. Gunakan 'PROFILE <username>;' tanpa argumen lain.

Tabel 7.3 Tabel Pengujian Fitur Wajib 3: Pengguna

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	REGISTER dengan input password sesuai ketentuan (8-20 char)	1. Masukkan nama folder yang valid 2. Masukkan perintah REGISTRASI; 3. Masukkan username bebas 4. Masukkan password 8-20 char	config-1;	REGISTER berhasil	REGISTRATION SUCCESSFUL ────────────────────────────────────── Account ID: USER008 Username: 00 Created: 2025-12-06 16:54:43 Initial Karma: 0 └────────────────────────────────────── ──────────────────────┘
2.	REGISTER dengan input password tidak sesuai ketentuan	1. Masukkan nama folder yang valid 2. Masukkan perintah REGISTRASI; 3. Masukkan username bebas 4. Masukkan password <8 atau >20 char	config-1;	REGISTER gagal karena password tidak sesuai ketentuan	[ERROR] Invalid password length Password must be 8-20 characters. You entered 1 characters.
3.	LOGIN dengan username dan password valid yang ada di database	1. Masukkan nama folder yang valid 2. Masukkan perintah LOGIN; 3. Masukkan username valid 4. Masukkan password valid	config-1;	LOGIN berhasil	LOGIN SUCCESSFUL ────────────────────────────────────── Welcome back, kebinganteng! Account ID: USER001 Karma: 48 Member since: 2025-11-01 21:10:00 └────────────────────────────────────── ──────────────────────┘
4.	LOGIN dengan username yang tidak valid (tidak ada di database)	1. Masukkan nama folder yang valid 2. Masukkan perintah LOGIN; 3. Masukkan username tidak valid	config-1;	LOGIN gagal karena username tidak ada di database	[ERROR] Account not found Username crazypeople67 does not exist. Please register first. Use REGISTER; to create a new account.
5.	LOGIN dengan username	1. Masukkan nama folder yang valid	config-1;	LOGIN gagal karena password salah	[ERROR] Authentication failed Incorrect password. Please try again.

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	yang valid dan password tidak valid (tidak ada di database)	2. Masukkan perintah LOGIN; 3. Masukkan username valid 4. Masukkan password yang salah			
6.	LOGOUT setelah login	1. Masukkan nama folder yang valid 2. LOGIN dengan valid seperti pengujian nomor 3 3. Masukkan perintah LOGOUT;	config-1;	LOGOUT berhasil	SIGNING OUT Currently logged in as: kebinganteng [DONE] Saving session [DONE] Signing out LOGOUT SUCCESSFUL You have been signed out successfully. Session ended for: kebinganteng
7.	LOGOUT tanpa login	1. Masukkan nama folder yang valid 2. Masukkan perintah LOGOUT;	config-1;	LOGOUT gagal karna belum login	[ERROR] Not logged in You must be logged in to logout. Use LOGIN; to sign in to your account.

Tabel 7.4 Tabel Pengujian Fitur Wajib 4: Profil

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	Menguji validitas login (kasus belum login)	1. Masukkan nama folder config 2. Jalankan perintah PROFIL	config-1;	PROFIL gagal karena belum login	COMMAND INPUT > PROFILE kebinganteng; [ERROR] Not logged in You must be logged in to view profiles. Use LOGIN; to sign in to your account.
2.	Menguji validitas login (kasus sudah login)	1. Masukkan nama folder config 2. Login 3. Jalankan perintah PROFIL	config-1;	PROFIL berhasil	USER PROFILE : kebinganteng ACCOUNT INFORMATION User ID: USER001 Username: kebinganteng Karma: 48 Joined: 2025-11-01 21:10:00 ACTIVITY STATISTICS Posts: 1 Comments: 0 Followers: 1 Following: 1 CONNECTION STATUS

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					○ You are not following this user RECENT POSTS 1. [r/programming] Perbedaan BFS dan DFS (42↑) Posted: 2025-11-02 21:10:10 [INFO] Profile displayed successfully
3	Menguji profil untuk pengguna yang tidak ditemukan	1. Masukkan nama folder config 2. Login 3. Jalankan perintah PROFIL <user> dengan <user> tidak ada di data	config-1;	PROFIL gagal karena tidak ada <user> di dalam data	COMMAND INPUT > [redacted] PROFILE me; [ERROR] User not found No user with username me exists. Please check your input and try again.
4	Menguji profil untuk pengguna yang ada di data	1. Masukkan nama folder config 2. Login 3. Jalankan perintah PROFIL <user> dengan <user> ada di data	config-1;	PROFIL berhasil	USER PROFILE : kebinganteng ACCOUNT INFORMATION User ID: USER001 Username: kebinganteng Karma: 48 Joined: 2025-11-01 21:10:00 ACTIVITY STATISTICS Posts: 1 Comments: 0 Followers: 1 Following: 1 CONNECTION STATUS ○ You are not following this user RECENT POSTS 1. [r/programming] Perbedaan BFS dan DFS (42↑) Posted: 2025-11-02 21:10:10 [INFO] Profile displayed successfully

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					── ── ─

Tabel 7.5 Tabel Pengujian Fitur Wajib 5: Post

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Menguji command POST pada subgroddit yang belum ada	1. Masukkan nama folder config 2. Login 3. Jalankan perintah POST dan masukkan subgroddit yang belum ada	config-1	Ada pesan untuk harus membuat subgroddit terlebih dahulu	SUBGRODDIT SELECTION ── │ Enter subgroddit name (must start with r/): │ └─► r/gaadanih; [ERROR] Subgroddit not found Subgroddit r/gaadanih doesn't exist. Tip: Use CREATE_SUBGRODDIT; to create it first.
2	Menguji command POST pada subgroddit yang sudah ada	1. Masukkan nama folder config 2. Login 3. Jalankan perintah POST dan masukkan subgroddit yang sudah ada	config-1	Post berhasil	SUBGRODDIT SELECTION ── │ Enter subgroddit name (must start with r/): │ └─► r/algorithms; ├── ──────────────┤ POST CONTENT ── │ Enter post title: │ └─► Tarjan SCC is effective for codeforces round 1069!; │ Enter post content: │ └─► I managed to solve this very hard problem! ORZ; [DONE] Checking content moderation [SUCCESS] Content approved [DONE] Creating post [SUCCESS] Post created successfully ┌── ──────────────┐ │ Post ID : P011 │ Subgroddit : r/algorithms └── ──────────────┘
3	Menguji command POST untuk user yang belum login	1. Masukkan nama folder config 2. Jalankan perintah POST	config-1	Tidak bisa melakukan posting	┌─ COMMAND INPUT │ └─> ▒▒▒ POST; [ERROR] Not logged in You must be logged in to create posts. Use LOGIN; to sign in to your account.
4	Menguji command VIEW_POST untuk post yang belum ada	1. Masukkan nama folder config 2. Jalankan perintah VIEW_POST <id> untuk id yang tidak ada di data	config-1	Tidak bisa view post	┌─ COMMAND INPUT │ └─> ▒▒▒ VIEW_POST P123; [ERROR] Post not found Post with ID P123 doesn't exist.

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
5	Menguji command VIEW_POST untuk post yang sudah ada	1. Masukkan nama folder config 2. Jalankan perintah VIEW_POST <id> untuk id yang sudah ada di data	config-1	Post ditampilkan ke layar	POST DETAILS Subgroddit: r/algorithms Title: test Author: @kebinganteng Posted: 2025-12-06 18:26:38 CONTENT test VOTING ↑ Upvotes: 1 ↓ Downvotes: 0 COMMENTS [INFO] Post loaded successfully
6	Menguji command DELETE_POST untuk post yang bukan milik pengguna	1. Masukkan nama folder config 2. Login 3. Jalankan perintah DELETE_POST <id> untuk id yang bukan merupakan post milik user sekarang	config-1	Tidak bisa menghapus post	[DONE] Locating post [ERROR] Authorization failed Only the post author can delete this post. Post ID: P007
7	Menguji command DELETE_POST untuk post yang merupakan milik pengguna	1. Masukkan nama folder config 2. Login 3. Jalankan perintah DELETE_POST <id> untuk id yang merupakan post milik user sekarang	config-1	Post berhasil dihapus	COMMAND INPUT > DELETE_POST P001; Home > Delete Post DELETE POST [DONE] Locating post CONFIRMATION REQUIRED Post ID: P001 Title: Perbedaan BFS dan DFS This will permanently delete the post and ALL its comments! > Are you sure? (Y/N): y;

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					[DONE] Deleting post and comments [SUCCESS] Post deleted successfully ┌──┐ │ Deleted Post: P001 │ Comments removed: All associated comments │ └──┘
8	Menguji command DELETE_POST untuk post yang tidak ada	1. Masukkan nama folder config 2. Login 3. Jalankan perintah DELETE_POST <id> untuk id yang tidak ada di data	config-1	Post tidak bisa dihapus karena tidak ada	[DONE] Locating post [ERROR] Post not found No post exists with ID: P999 Tip: Use SHOW_FEED; to view available posts.

Tabel 7.6 Tabel Pengujian Fitur Wajib 6: Subgroddit

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Membuat suatu subgroddit baru	1. Masukkan nama folder config 2. Membuat subgroddit	config-1;	Subgroddit berhasil dibuat	**CREATE_SUBGRODDIT r/hello;** Subgroddit r/hello berhasil dibuat!
2	Mencoba membuat subgroddit yang sudah ada sebelumnya	1. Masukkan nama folder config 2. Jalankan perintah CREATE_SUBGRODDIT r/hello;	config-1;	Terdapat peringatan bahwa nama subgroddit tidak valid	**CREATE_SUBGRODDIT r/hello;** [Error] Nama Subgroddit HARUS UNIK dan diawali dengan "r/".
3	Melakukan validasi format subgroddit yang harus diawali r/	1. Masukkan nama folder config 2. Jalankan perintah CREATE_SUBGRODDIT hello;	config-1;	Terdapat peringatan bahwa nama subgroddit tidak valid	**CREATE_SUBGRODDIT hello;** [Error] Nama Subgroddit HARUS UNIK dan diawali dengan "r/".
4	Mengecek mode hot terurut dari tinggi ke rendah berdasarkan banyak upvote	1. Masukkan nama folder config 2. Jalankan perintah VIEW_SUBGRODDIT r/algorithms HOT DECR;	config-1;	Daftar posts pada subgroddit yang dilihat sesuai tujuan testing	**VIEW_SUBGRODDIT r/algorithms HOT DECR;** 🔥 Subgroddit: r/algorithms (sorted by HOT) ========== 1. [P003] Optimasi Dijkstra vs A* (87 / 2) - oleh drnefario 2. [P001] Perbedaan BFS dan DFS (45 / 3) - oleh kebinganteng 3. [P002] Cara kerja Topological Sort (28 / 1) - oleh alice 4. [P007] Kenapa Merge Sort stabil? (21 / 0) - oleh bob 5. [P009] Apa itu memoization? (17 / 2) - oleh alice 6. [P005] Implementasi Union-Find (12 / 0) - oleh minion_dave 7. [P006] Perbedaan Heaps vs Priority Queue (9 / 1) - oleh alice 8. [P010] Struktur data buat frequency counting? (6 / 0) - oleh alice 9. [P008] Cara optimize recursion di C (4 / 0) - oleh minion_veda ========== Gunakan VIEW_POST <ID> untuk melihat detail postingan.
5	Mengecek mode hot terurut dari rendah ke tinggi	1. Masukkan nama folder config 2. Jalankan perintah VIEW_SUBGRODDIT r/algorithms HOT INCR;	config-1;	Daftar posts pada subgroddit yang dilihat	**VIEW_SUBGRODDIT r/algorithms HOT INCR;** 🔥 Subgroddit: r/algorithms (sorted by HOT) ========== 1. [P008] Cara optimize recursion di C (4 / 0) - oleh minion_veda

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	berdasarkan banyak upvote			sesuai tujuan testing	2. [P010] Struktur data buat frequency counting? (6 / 0) - oleh alice 3. [P006] Perbedaan Heaps vs Priority Queue (9 / 1) - oleh alice 4. [P005] Implementasi Union-Find (12 / 0) - oleh minion_dave 5. [P009] Apa itu memoization? (17 / 2) - oleh alice 6. [P007] Kenapa Merge Sort stabil? (21 / 0) - oleh bob 7. [P002] Cara kerja Topological Sort (28 / 1) - oleh alice 8. [P001] Perbedaan BFS dan DFS (45 / 3) - oleh kebinganteng 9. [P003] Optimasi Dijkstra vs A* (87 / 2) - oleh drnefario ========== Gunakan VIEW_POST <ID> untuk melihat detail postingan.
6	Mengecek mode new terurut dari tinggi ke rendah berdasarkan tanggal pembuatan	1. Masukkan nama folder config 2. Jalankan perintah `VIEW_SUBGRODDIT r/algorithms NEW DECR;`	config-1;	Daftar posts pada subgroddit yang dilihat sesuai tujuan testing	**VIEW_SUBGRODDIT r/algorithms NEW DECR;** Subgroddit: r/algorithms (sorted by NEW) ========== 1. [P010] Struktur data buat frequency counting? (6 / 0) - oleh alice 2. [P008] Cara optimize recursion di C (4 / 0) - oleh minion_veda 3. [P005] Implementasi Union-Find (12 / 0) - oleh minion_dave 4. [P006] Perbedaan Heaps vs Priority Queue (9 / 1) - oleh alice 5. [P007] Kenapa Merge Sort stabil? (21 / 0) - oleh bob 6. [P009] Apa itu memoization? (17 / 2) - oleh alice 7. [P003] Optimasi Dijkstra vs A* (87 / 2) - oleh drnefario 8. [P001] Perbedaan BFS dan DFS (45 / 3) - oleh kebinganteng 9. [P002] Cara kerja Topological Sort (28 / 1) - oleh alice ========== Gunakan VIEW_POST <ID> untuk melihat detail postingan.
7	Mengecek mode new terurut dari rendah ke ringgi berdasarkan tanggal pembuatan	1. Masukkan nama folder config 2. Jalankan perintah `VIEW_SUBGRODDIT r/algorithms NEW INCR;`	config-1;	Daftar posts pada subgroddit yang dilihat sesuai tujuan testing	**VIEW_SUBGRODDIT r/algorithms NEW INCR;** Subgroddit: r/algorithms (sorted by NEW) ========== 1. [P001] Perbedaan BFS dan DFS (45 / 3) - oleh kebinganteng 2. [P002] Cara kerja Topological Sort (28 / 1) - oleh alice 3. [P003] Optimasi Dijkstra vs A* (87 / 2) - oleh drnefario 4. [P009] Apa itu memoization? (17 / 2) - oleh alice 5. [P007] Kenapa Merge Sort stabil? (21 / 0) - oleh bob 6. [P006] Perbedaan Heaps vs Priority Queue (9 / 1) - oleh alice 7. [P005] Implementasi Union-Find (12 / 0) - oleh minion_dave 8. [P008] Cara optimize recursion di C (4 / 0) - oleh minion_veda 9. [P010] Struktur data buat frequency counting? (6 / 0) - oleh alice ========== Gunakan VIEW_POST <ID> untuk melihat detail postingan.
8	Menguji apakah saat subgroddit	1. Masukkan nama folder config 2. Jalankan perintah `VIEW_SUBGRODDIT`	config-1;	Terdapat peringatan bahwa subgroddit	**VIEW_SUBGRODDIT r/not-a-subgroddit NEW INCR;** Subgroddit r/not-a-subgroddit belum ditemukan! Gunakan perintah CREATE_SUBGRODDIT untuk membuatnya terlebih dahulu.

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	tidak ada ditangani	`r/not-a-subgroddit NEW INCR;`		belum ditemukan	

9	Menguji apakah saat mode selain HOT atau NEW ditangani	1. Masukkan nama folder config
		2. Jalankan perintah `VIEW_SUBGRODDIT r/algorithms WARM INCR;`
			config-1;	Terdapar peringatan bahwa mode tidak valid	`VIEW_SUBGRODDIT r/algorithms WARM INCR;`
					
					`Mode WARM tidak dikenali!`
					`Gunakan HOT atau NEW sebagai mode yang valid.`

Tabel 7.7 Tabel Pengujian Fitur Wajib 7: Comment and Reply

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	COMMENT namun belum login	1. Masukkan nama folder yang valid 2. Masukkan perintah COMMENT <POST_ID> <COMMENT_ID>;	config-1;	DELETE_COMMENT gagal karena belum login	`[ERROR] Authentication required` `You must be logged in to delete comments.`

2.	COMMENT pada postingan yang valid dan COMMENT_ID yang valid	1. Masukkan nama folder yang valid
		2. Lakukan login yang valid
		3. Masukkan perintah COMMENT <POST_ID> <COMMENT_ID>; dengan post id dan comment id valid
			config-1;	Lanjut meminta isi content	`[DONE] Validating post`
					
					`COMMENT CONTENT`
					`─────────────────────────────────────────`
					
					`│ Commenting on post: P001`
					`├─────────────────────────────────────────`
					`└──────────────────────┤`
					
					`└→ Enter your comment:`

3.	COMMENT pada postingan yang tidak valid	1. Masukkan nama folder yang valid
		2. Lakukan login yang valid
		3. Masukkan perintah COMMENT <POST_ID> <COMMENT_ID>; dengan post id tidak valid karena tidak ada postingan tersebut di database
			config-1;	Tidak lanjut meminta isi content karena post tidak ada	`[ERROR] Post not found`
					`No post exists with ID: P676`
					
					`Tip: Use SHOW_FEED; to view available posts.`

4.	COMMENT pada postingan yang valid namun comment id yang tidak valid	1. Masukkan nama folder yang valid
		2. Lakukan login yang valid
		3. Masukkan perintah COMMENT <POST_ID> <COMMENT_ID>; dengan post id valid namun comment id tidak valid karena tidak ada komentar tersebut di database
			config-1;	Tidak lanjut meminta isi content karena comment tidak ada	`[ERROR] Parent comment not found`
					`No comment #999 exists on post P001`
					
					`Tip: Use VIEW_POST P001; to see existing comments.`

5.	COMMENT pada postingan valid dan comment valid dan isi content	1. Masukkan nama folder yang valid
		2. Lakukan login yang valid
		3. Masukkan perintah COMMENT
			config-1;	COMMENT berhasil ditambahkan	`[DONE] Validating post`
					
					`COMMENT CONTENT`
					`─────────────────────────────────────────`
					
					`│ Commenting on post: P001`

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	valid (ada isinya)	<POST_ID> <COMMENT_ID>; dengan post id dan comment id valid 3. Isi content bebas tapi jangan kosong			
6.	COMMENT pada postingan valid dan comment id valid namun content kosong	1. Masukkan nama folder yang valid 2. Lakukan login yang valid 3. Masukkan perintah COMMENT <POST_ID> <COMMENT_ID>; dengan post id dan comment id valid 3. Isi content dengan kosong yaitu ;	config-1;	COMMENT gagal ditambahkan karena content tidak boleh kosong	└→ Enter your comment: ; [ERROR] Invalid input Comment content cannot be empty.
7.	DELETE_COMMENT pada komentar yang valid (ada di database) dan komentar adalah milik user	1. Masukkan nama folder yang valid 2. Lakukan login yang valid 3. Masukkan perintah DELETE_COMMENT <POST_ID> <COMMENT_ID>; dengan post id dan comment id valid	config-1;	DELETE_COMMENT berhasil	CONFIRMATION REQUIRED Post ID: P001 Comment ID: #5 Content: APA This will permanently delete the comment and ALL its replies! └→ Are you sure? (Y/N): Y; [DONE] Deleting comment and replies [SUCCESS] Comment deleted successfully Deleted Comment: #5 From Post: P001 Replies removed: All nested replies
8.	DELETE_COMMENT pada komentar yang tidak valid (tidak ada di database)	1. Masukkan nama folder yang valid 2. Lakukan login yang valid 3. Masukkan perintah DELETE_COMMENT <POST_ID> <COMMENT_ID>; dengan post id tidak valid dan comment id tidak valid	config-1;	DELETE_COMMENT gagal karena postingan atau komentar tidak ada	***JIKA POSTINGAN TIDAK ADA*** [ERROR] Post not found No post exists with ID: P999 Tip: Use SHOW_FEED; to view available posts. ***JIKA POSTINGAN ADA TAPI KOMENTAR TIDAK ADA*** [ERROR] Comment not found No comment #999 exists on post P001 Tip: Use VIEW_POST P001; to see existing comments.
9	DELETE_COMMENT pada komentar yang bukan miliknya	1. Masukkan nama folder yang valid 2. Lakukan login yang valid 3. Masukkan perintah DELETE_COMMENT <POST_ID> <COMMENT_ID>; dengan post id valid	config-1;	DELETE_COMMENT gagal karena komentar yang ingin dihapus bukan miliknya	[ERROR] Authorization failed Only the comment author can delete this comment. Comment ID: #3 on post P001

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		tapi comment id bukan miliknya			

Tabel 7.8 Tabel Pengujian Fitur Wajib 8: Voting

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	Menguji UPVOTE_POST pada postingan orang lain	1. Login 2. Jalankan perintah: UPVOTE_POST <post_id orang lain>;	config-1;	Sistem memberikan upvote	**[DONE] Validating post** **[DONE] Processing vote** **[SUCCESS]** Upvote recorded Post **P003** received your upvote
2.	Menguji UPVOTE_POST pada postingan yang sudah diberikan upvote	1. Login 2. Jalankan perintah UPVOTE_POST <post_id orang lain>;	config-1;	Sistem menolak upvote	**[DONE] Validating post** **[DONE] Processing vote** **[WARNING]** Already upvoted You have already upvoted post **P002**
3.	Menguji UPVOTE_POST pada postingan yang sudah diberikan downvote	1. Login 2. Jalankan perintah UPVOTE_POST <post_id orang lain>;	config-1;	Sistem mengubah vote dari downvote menjadi upvote	**[DONE] Validating post** **[DONE] Processing vote** **[SUCCESS] Vote changed** **Your vote on post P002 changed to upvote**
4.	Menguji UPVOTE_POST pada postingan sendiri	1. Login 2. Jalankan perintah UPVOTE_POST <post_id sendiri>;	config-1;	Sistem menolak vote	**[DONE] Validating post** **[ERROR]** Self-voting not allowed You cannot vote on your own post. **Post ID: P001**
5.	Menguji UPVOTE_POST pada id post invalid	1. Login 2. Jalan perintah UPVOTE_POST <invalid post_id>;	config-1	Sistem tidak menemukan post id, sistem gagal memberikan upvote	**[DONE] Validating post** **[ERROR] Post not found** **No post exists with ID: P099** **Tip: Use SHOW_FEED; to view available posts.**
6.	Menguji DOWNVOTE_POST pada postingan orang lain	1. Login 2. Jalankan perintah: DOWNVOTE_POST <post_id orang lain>;	config-1;	Sistem memberikan downvote	**[DONE] Validating post** **[DONE] Processing vote** **[SUCCESS]** Downvote recorded Post **P003** received your downvote
7.	Menguji DOWNVOTE_POST pada postingan yang sudah diberikan downvote	1. Login 2. Jalankan perintah DOWNVOTE_POST <post_id orang lain>;	config-1;	Sistem menolak downvote	**[DONE] Validating post** **[DONE] Processing vote** **[WARNING]** Already downvote You have already downvote post **P002**
8.	Menguji DOWNVOTE_POST pada postingan yang sudah diberikan upvote	1. Login 2. Jalankan perintah DOWNVOTE_POST <post_id orang lain>;	config-1;	Sistem mengubah vote dari upvote menjadi downvote	**[DONE] Validating post** **[DONE] Processing vote** **[SUCCESS] Vote changed** **Your vote on post P002 changed to downvote**
9.	Menguji DOWNVOTE_POST pada	1. Login 2. Jalankan perintah DOWNVOTE_POST <post_id sendiri>;	config-1;	Sistem menolak vote	**[DONE] Validating post** **[ERROR]** Self-voting not allowed You cannot vote on your own post. **Post ID: P001**

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	postingan sendiri				
10.	Menguji DOWNVOTE_POST pada id post invalid	1. Login 2. Jalan perintah DOWNVOTE_POST <invalid post_id>;	config-1	Sistem tidak menemukan post id, sistem gagal memberikan downvote	**[DONE] Validating post** **[ERROR] Post not found** **No post exists with ID: P099** **Tip: Use SHOW_FEED; to view available posts.**
11.	Menguji UNDO_VOTE_POST pada post yang sudah diberikan vote	1. Login 2. Jalankan perintah UNDO_VOTE_POST <post_id>;	config-1	Sistem menghapus vote pada post terkait	**[DONE] Checking vote status** **[DONE] Removing vote** **[SUCCESS] Vote removed** **Your UPVOTE vote on post P002 has been cancelled.**
12.	Menguji UNDO_VOTE_POST pada post yang belum diberikan vote	1. Login 2. Jalankan perintah UNDO_VOTE_POST <post_id>;	config-1;	Sistem gagal menghapus vote pada post terkait	**[DONE] Checking vote status** **[WARNING] No vote found** **You haven't voted on post P002 yet.**
13.	Menguji UNDO_VOTE_POST pada invalid post id	1. Login 2. Jalankan perintah UNDO_VOTE_POST <post_id>;	config-1;	Sistem tidak menemukan post, gagal menghapus vote	**[DONE] Checking vote status** **[WARNING] No vote found** **You haven't voted on post P999 yet.**
14.	Menguji UPVOTE_COMMENT pada komentar user lain	1. Login 2. Jalankan perintah UPVOTE_COMMENT <post_id> <comment_id user lain>;	config-1;	Sistem memberikan upvote pada komentar terkait	**[DONE] Validating post and comment** **[DONE] Processing vote** **[SUCCESS] Upvote recorded** **Comment #2 on post P001 received your upvote**
15.	Menguji UPVOTE_COMMENT pada komentar sendiri	1. Login 2. Jalankan perintah UPVOTE_COMMENT <post_id> <comment_id sendiri>	config-1;	Sistem gagal memberikan upvote pada komentar sendiri	**[ERROR] Self-voting not allowed** **You cannot vote on your own comment.** **Comment: #4 on post P001**
16.	Menguji UPVOTE_COMMENT pada komentar yang sudah diberikan upvote	1. Login 2. Jalankan perintah UPVOTE_COMMENT <post_id> <comment_id>;	config-1;	Sistem gagal memberikan upvote pada komentar yang sudah diberikan upvote	**[DONE] Validating post and comment** **[DONE] Processing vote** **[WARNING] Already upvoted** **You have already upvoted comment #3**
17.	Menguji UPVOTE_COMMENT pada komentar yang sudah diberikan downvote	1. Login 2. Jalankan perintah UPVOTE_COMMENT <post_id> <comment_id>;	config-1;	Sistem mengubah vote pada komentar dari downvote ke upvote	**[DONE] Validating post and comment** **[DONE] Processing vote** **[SUCCESS] Vote changed** **Your vote on comment #3 changed to upvote**
18.	Menguji UPVOTE_COMMENT pada komentar invalid/tidak ada	1. Login 2. Jalankan perintah UPVOTE_COMMENT <post_id> <comment_id>;	config-1;	Sistem tidak menemukan komentar, upvote komentar gagal	**[ERROR] Comment not found** **No comment #6 exists on post P001** **Tip: Use VIEW_POST P001; to see comments.**
19.	Menguji UPVOTE_	1. Login	config-1;	Sistem tidak menemukan	**[DONE] Validating post and comment** **[ERROR] Post not found**

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	COMMENT pada post invalid/tidak ada	2. Jalankan perintah UPVOTE_COMMENT <post_id> <comment_id>;		post id, upvote komentar gagal	`No post exists with ID: P999` `Tip: Use SHOW_FEED; to view available posts.`

20.	Menguji UNDO_VOTE_COMMENT pada komentar yang sudah diberikan vote	1. Login
		2. Jalankan perintah UNDO_VOTE_COMMENT <post_id> <comment_id>;
			config-1;	Sistem menghapus vote pada komentar di post terkait	`[DONE] Validating post and comment`
					`[DONE] Checking vote status`
					`[DONE] Removing vote`
					`[SUCCESS] Vote removed`
					`Your UPVOTE vote on comment #1 has been cancelled.`

21.	Menguji UNDO_VOTE_COMMNT pada komentar yang belum diberikan vote	1. Login
		2. Jalankan perintah UNDO_VOTE_COMMENT <post_id> <comment_id>;
			config-1;	Sistem tidak menemukan vote terkait, gagal menghapus vote	`[DONE] Validating post and comment`
					`[DONE] Checking vote status`
					`[WARNING] No vote found`
					`You haven't voted on comment #2 yet.`

22.	Menguji UNDO_VOTE_POST pada post id tidak ada /invalid	1. Login
		2. Jalankan perintah UNDO_VOTE_COMMENT <post_id> <comment_id>;
			config-1;	Sistem tidak menemukan post terkait, gagal menghapus vote	`[DONE] Validating post and comment`
					`[ERROR] Post not found`
					`No post exists with ID: P999`
					`Tip: Use SHOW_FEED; to view available posts.`

Tabel 7.9 Tabel Pengujian Fitur Wajib 9: Social

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Menguji apakah FOLLOW berhasil untuk username valid	1. Login 2. Jalankan perintah: FOLLOW bob;	config-1;	Sistem menambah edge dan pesan sukses	`FOLLOW bob;` `Berhasil mengikuti bob.`
2	Menguji FOLLOW ke diri sendiri	1. Login sebagai kebinganteng 2. FOLLOW kebinganteng;	config-1;	Sistem menolak follow diri sendiri	`FOLLOW kebinganteng;` `Anda tidak dapat mengikuti diri sendiri.`
3	Menguji FOLLOW user tidak ditemukan	1. Login 2. FOLLOW not_exist;	config-1;	Sistem menolak dan memberi pesan user tidak ditemukan	`FOLLOW not_exist;` `User dengan username "not_exist" tidak ditemukan.`
4	FOLLOW yang sudah pernah dilakukan	1. Login 2. FOLLOW bob; 3. FOLLOW bob;	config-1;	Sistem menolak karena sudah di-follow	`FOLLOW bob;` `Anda sudah mengikuti bob.`
5	Menguji UNFOLLOW normal	1. Login 2. FOLLOW bob; 3. UNFOLLOW bob;	config-1;	Edge dihapus dan pesan sukses	`UNFOLLOW bob;` `Berhasil berhenti mengikuti bob.`
6	UNFOLLOW user	1. Login 2. UNFOLLOW bob;	config-1;	Sistem menolak	`UNFOLLOW bob;` `Anda belum mengikuti bob.`

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	belum di-follow				

7	UNFOLLOW user tidak ditemukan	1. Login
		2. UNFOLLOW not_exist;
			config-1;	Pesan user tidak ditemukan	`UNFOLLOW not_exist;`
					
					`User dengan username "not_exist" tidak ditemukan.`

8	UNFOLLOW saat belum login	1. Tidak login
		2. UNFOLLOW bob;
			config-1;	Sistem menolak	`UNFOLLOW bob;`
					
					`Anda belum login. Silakan login terlebih dahulu.`

9	Menampilkan daftar user yang di-follow	1. Login
		2. FOLLOW bob;
		3. FOLLOW alice;
		4. FOLLOWING;
			config-1;	Daftar bob dan alice	`FOLLOWING;`
					
					`Daftar akun yang diikuti oleh kebinganteng:`
					`- alice`
					`- bob`

10	FOLLOWING user tidak follow siapapun	1. Login user baru
		2. FOLLOWING
			config-1;	Tidak mengikuti siapapun	`FOLLOWING;`
					
					`Daftar akun yang diikuti oleh new_user:`
					`(Tidak mengikuti siapa pun)`

11	FOLLOWING tanpa login	1. Tidak login
		2. FOLLOWING
			config-1;	Sistem menolak	`FOLLOWING;`
					
					`Anda belum login. Silakan login terlebih dahulu.`

12	Menampilkan daftar followers user	1. Login bob
		2. FOLLOWERS;
			config-1;	Menampilkan followers	`FOLLOWERS;`
					
					`Daftar akun yang mengikuti bob:`
					`- kebinganteng`

13	Followers user tanpa followers	1. Login user baru
		2. FOLLOWERS
			config-1;	Belum ada followers	`FOLLOWERS;`
					
					`Daftar akun yang mengikuti new_user:`
					`(Belum ada yang mengikuti akun ini)`

14	Followers user lain	1. LOGIN new_user
		2. FOLLOW bob
		3. LOGOUT
		4. LOGIN bob
		5. FOLLOWERS;
			config-1;	new_user muncul sebagai follower bob	`FOLLOWERS;`
					
					`Daftar akun yang mengikuti bob:`
					`- new_user`

15	Followers user tidak valid	1. FOLLOWERS not_exist	config-1;	User tidak ditemukan	`FOLLOWERS not_exist;`
					
					`User dengan username "not_exist" tidak ditemukan.`

Tabel 7.10 Tabel Pengujian Fitur Wajib 10: Save & Load

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar

1	Mengecek apakah fitur Save berjalan	1. Masukkan folder config awal
		2. Masukkan perintah `SAVE;`
		3. Masukkan nama folder `config-2;`
			config-1;	Terbentuk folder config-2 pada folder parent config	`COMMAND INPUT`
					`> SAVE;`
					
					`FOLDER SELECTION`
					`Enter destination folder name:`
					
					`FOLDER NAME`
					`> config-2;`

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					[WARNING] Creating new folder Folder "config-2" does not exist. Creating it now... [DONE] Creating folder structure [SUCCESS] Folder created! Directory config-2 is ready for saving. SAVING PROCESS Destination folder: config-2 [DONE] Preparing to save data Saving users... [OK] 7 users Saving comments... [OK] 3 comments Saving posts... [OK] 10 posts Saving subgroddits... [OK] 4 subgroddits Saving social connections... [OK] 4 connections Saving votes... [OK] 20 votes Saving security config... [DONE] [SUCCESS] Data saved successfully! All data has been saved to config-2
2	Mengecek apakah Save pada folder yang sudah dibuat dapat dilakukan	1. Masukkan folder config awal 2. Masukkan perintah `SAVE;` 3. Masukkan nama folder `config-2;`	config-1;	Isi folder config-2 berubah menjadi yang terbaru pada folder parent config	COMMAND INPUT > SAVE; FOLDER SELECTION Enter destination folder name: FOLDER NAME > config-2; SAVING PROCESS Destination folder: config-2 [DONE] Preparing to save data Saving users... [OK] 7 users Saving comments... [OK] 3 comments Saving posts... [OK] 10 posts Saving subgroddits... [OK] 4 subgroddits Saving social connections... [OK] 4 connections Saving votes... [OK] 20 votes Saving security config... [DONE]

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					[SUCCESS] Data saved successfully! All data has been saved to config-2
4	Mengecek apakah load data berhasil atau tidak	1. Masukkan folder config awal 2. Masukkan perintah `LOAD;` 3. Masukkan nama folder `config-1;`	config-1;	config-1 berhasil dimuat ke dalam program	COMMAND INPUT > LOAD; FOLDER SELECTION Enter configuration folder name: FOLDER NAME > config-1; LOADING PROCESS Source folder: config-1 [DONE] Preparing to load data Loading comments... [OK] 3 comments Loading posts... [OK] 10 posts Loading users... [OK] 0 users Loading subgroddits... [OK] 4 subgroddits Loading social connections... [OK] 4 connections Loading votes... [OK] 20 votes [SUCCESS] Data loaded successfully! All data has been loaded from config-1
5	Menangani saat file config tidak tersedia	1. Masukkan folder config awal 2. Masukkan perintah `LOAD;` 3. Masukkan nama folder `config-999;`	config-999;	Pesan peringatan bahwa folder config yang ingin dimuat tidak ditemukan	COMMAND INPUT > LOAD; FOLDER SELECTION Enter configuration folder name: FOLDER NAME > config-999; [ERROR] Folder not found Configuration folder "config-999" does not exist! Tip: Check config/ directory for available folders.
6	Menangani kondisi saat pengguna sudah login	1. Masukkan folder config awal 2. Masukkan perintah `LOGIN;` 3. Masukkan perintah `LOAD;` 4. Masukkan nama folder `config-999;`	config-1;	Pesan peringatan bahwa pengguna masih login	COMMAND INPUT > LOAD; [ERROR] Cannot load while logged in You must logout first to perform this operation. Tip: Use LOGOUT; to logout from current session.

Tabel 7.11 Tabel Pengujian Fitur Bonus 1: Feed

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	SHOW_FEED normal tanpa LIMIT	1. Login ke user yang sudah memiliki following 2. SHOW_FEED LATEST;	config-1;	Feed terbaru dulu	SHOW_FEED LATEST; ═══ Feed: bob (sorted by LATEST - newest first) ═══ 1. [P010] Struktur data buat frequency counting? (2025-12-03 16:02:11) - oleh alice 2. [P008] Cara optimize recursion di C (2025-12-03 14:29:55) - oleh minion_veda 3. [P006] Perbedaan Heaps vs Priority Queue (2025-12-03 07:55:12) - oleh alice 4. [P009] Apa itu memoization? (2025-12-02 20:18:44) - oleh alice 5. [P002] Cara kerja Topological Sort (2025-11-02 21:10:10) - oleh alice 6. [P004] Best C textbooks? (2025-10-31 21:10:10) - oleh alice Gunakan VIEW_POST <ID> untuk melihat detail postingan. ═══
2	SHOW_FEED NEWEST	1. Login ke user yang sudah memiliki following 2. SHOW_FEED NEWEST	config-1;	Post dari yang paling lama	SHOW_FEED NEWEST; ═══ Feed: bob (sorted by NEWEST - oldest first) ═══ 1. [P004] Best C textbooks? (2025-10-31 21:10:10) - oleh alice 2. [P002] Cara kerja Topological Sort (2025-11-02 21:10:10) - oleh alice 3. [P009] Apa itu memoization? (2025-12-02 20:18:44) - oleh alice 4. [P006] Perbedaan Heaps vs Priority Queue (2025-12-03 07:55:12) - oleh alice 5. [P008] Cara optimize recursion di C (2025-12-03 14:29:55) - oleh minion_veda 6. [P010] Struktur data buat frequency counting? (2025-12-03 16:02:11) - oleh alice Gunakan VIEW_POST <ID> untuk melihat detail postingan. ═══
3	SHOW_FEED LIMIT valid	1. Login 2. FOLLOW user dengan >=3 post 3. SHOW_FEED LATEST 2;	config-1;	Hanya 2 post	SHOW_FEED LATEST 2; ═══ Feed: bob (sorted by LATEST - newest first) ═══ 1. [P010] Struktur data buat frequency counting? (2025-12-03 16:02:11) - oleh alice 2. [P008] Cara optimize recursion di C (2025-12-03 14:29:55) - oleh minion_veda Gunakan VIEW_POST <ID> untuk melihat detail postingan. ═══
4	LIMIT tidak valid	SHOW_FEED LATEST 0;	config-1;	Error LIMIT	SHOW_FEED LATEST 0; LIMIT harus berupa bilangan bulat positif.
5	Mode salah	SHOW_FEED WRONGMODE;	config-1;	Error mode	SHOW_FEED WRONGMODE; Mode feed tidak dikenal. Gunakan 'LATEST' atau 'NEWEST'.

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
6	Tidak ada yang di-follow	Login user baru SHOW_FEED	config-1;	Feed kosong	SHOW_FEED LATEST; Feed: new_user (sorted by LATEST - newest first) Feed kosong. Kamu belum mengikuti siapa pun.
7	Di-follow tapi belum ada postingan	1. Login 2. FOLLOW minion_aved 3. minion_aved tidak punya post 4. SHOW_FEED;	config-1;	Feed kosong	Feed: new_user (sorted by LATEST - newest first) Feed kosong. Belum ada postingan dari akun yang kamu ikuti.
8	Belum login	SHOW_FEED LATEST tanpa login	config-1;	Sistem menolak	SHOW_FEED LATEST; Anda belum login! Masuk terlebih dahulu untuk dapat mengakses Groddit.

Tabel 7.12 Tabel Pengujian Fitur Bonus 2: Trending Analysis

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	Menguji Trending Analysis dalam periode 24 jam terakhir	1. Login 2. Jalankan perintah TRENDING <subgroddit: e.g. r/algorithms> <int waktu> <hour/minute/second>	config-1;	Tidak ada data 24 jam terakhir sehingga sistem tidak bisa memberikan trending analysis dengan periode tersebut	**[DONE] Analyzing trending topics** **TRENDING TOPICS** **Subgroddit: r/algorithms** **Period: Last 24 hour** (No trending data for this period)
2.	Menguji Trending Analysis dalam periode 1440 hour (2 bulan) terakhir	1. Login 2. Jalankan perintah TRENDING <subgroddit: e.g. r/algorithms> <int waktu> <hour/minute/second>	config-1;	Sistem memberikan trending analysis dengan periode waktu tersebut	**[DONE] Analyzing trending topics** **TRENDING TOPICS** **Subgroddit: r/algorithms** **Period: Last 1440 hour** **1.** "**perbedaan**" - 3 mentions **2.** "**sort**" - 3 mentions **3.** "**bisa**" - 2 mentions **4.** "**contoh**" - 2 mentions **5.** "**efisien**" - 2 mentions **[INFO]** Hottest topic: "**perbedaan**" in "**Perbedaan BFS dan DFS**"
3.	Menguji Trending Analysis pada subgroddit yang tidak ada/invalid	1. Login 2. Jalankan perintah TRENDING <subgroddit: e.g. r/algorithms> <int waktu> <hour/minute/second>	config-1;	Sistem tidak menemukan subgroddit terkait sehingga tidak bisa melakukan trending analysis	**[DONE] Analyzing trending topics** **[ERROR]** Subgroddit not found No subgroddit named: **r/AkuCintaALPRO** **Tip:** Use **SEARCH_SUBGRODDIT r/;** to see all subgroddits.

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
4.	Menguji inovasi dari kami, yaitu Trending Analysis dalam periode ALL	1. Login 2. Jalankan perintah TRENDING <subgroddit: e.g. r/algorithms> ALL;	config-1;	Sistem mmberikan trending analysis dengan periode waktu awal-present (ALL)	**[DONE] Analyzing trending topics** **TRENDING TOPICS** **Subgroddit: r/algorithms** **Period: All Time** **1. "perbedaan" - 3 mentions** **2. "sort" - 3 mentions** **3. "bisa" - 2 mentions** **4. "contoh" - 2 mentions** **5. "efisien" - 2 mentions** **[INFO] Hottest topic:** **"perbedaan" in "Perbedaan BFS dan DFS"**

Tabel 7.13 Tabel Pengujian Fitur Bonus 3: Advanced Search

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Menguji advanced search untuk username	1. Masukkan nama folder config 2. Jalankan perintah SEARCH_USER <prefix>	config-1;	Menunjukkan hasil username dengan prefix <prefix>	[DONE] Searching users SEARCH RESULTS Prefix: "m" 1. @minion_aved 2. @minion_dave 3. @minion_veda [INFO] Total results: 3 users
2	Menguji advanced search untuk judul post	1. Masukkan nama folder config 2. Jalankan perintah SEARCH_POST <prefix>	config-1;	Menunjukkan hasil judul post dengan prefix <prefix>	COMMAND INPUT > SEARCH_POST C; Menampilkan hasil pencarian post dengan prefix "C" 1. [r/algorithms] Cara kerja Topological Sort 2. [r/algorithms] Cara optimize recursion di C
3	Menguji advanced search untuk nama subgroddit	1. Masukkan nama folder config 2. Jalankan perintah SEARCH_SUBGRODDIT <prefix>	config-1;	Menunjukkan hasil nama subgroddit dengan prefix <prefix>	[DONE] Searching subgroddits SEARCH RESULTS Name prefix: "r/a" 1. r/algorithms 2. r/askreddit [INFO] Total results: 2 subgroddits

Tabel 7.14 Tabel Pengujian Fitur Bonus 4: Following Recommendation

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Rekomendasi normal	1. Login A 2. Struktur follow mutual 3. FRIEND_RECOMMENDATION	config-1;	Menampilkan rekomendasi	TOP RECOMMENDATIONS Based on mutual connections 1. @kebinganteng (1 mutual) [INFO] Recommendations based on BFS graph traversal
2	Tidak ada mutual	1. Login 2. FOLLOW user tanpa mutual 3. FRIEND_RECOMMENDATION	config-1;	Tidak ada rekomendasi	RECOMMENDATIONS User: @kebinganteng (No recommendations available)
3	User belum follow siapapun	Login user baru FRIEND_RECOMMENDATION	config-1;	Tidak ada rekomendasi	RECOMMENDATIONS User: @new_user (No recommendations available)
4	Belum login	FRIEND_RECOMMENDATION tanpa login	config-1;	Sistem menolak	[ERROR] Authentication required You must be logged in to view recommendations. Tip: Use LOGIN; to access your account.
5	Argumen tambahan	FRIEND_RECOMMENDATION bob;	config-1;	Format salah	[ERROR] Invalid command format This command takes no arguments. Usage: FRIEND RECOMMENDATION;

Tabel 7.15 Tabel Pengujian Fitur Bonus 5: Content Moderation

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	Mencegah COMMENT dengan isi konten yang mengandung blacklisted words	Prekondisi: input config valid dan mengandung file blacklisted_words.json, sudah login, comment id dan post id valid 1. Masukkan COMMENT <POST_ID> <COMMENT_ID> 2. Isi konten dengan kandungan kata di blacklisted_words.json	config-1;	COMMENT gagal karena mengandung blacklisted words	Commenting on post: P001 ↳ Enter your comment: SAMPAH JELEK HAMA; [DONE] Checking content moderation [ERROR] Content moderation success Comment contains blacklisted word: "sampah"
2.	Mencegah POST dengan isi konten yang mengandun	Prekondisi: input config valid dan mengandung file blacklisted_words.json, sudah login,	config-1;	POST gagal karena mengandung blacklisted words	Enter post content: ↳ SAMPAH; [DONE] Checking content moderation

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	g blacklisted words	comment id dan post id valid 1. Masukkan POST; 2. Masukkan nama SubGroddit yang valid 3. Masukkan judul yang valid 4. Masukkan konten dengan kandungan kata yang ada di blacklisted_words.json			[ERROR] Content moderation success Post contains blacklisted words: sampah

Tabel 7.16 Tabel Pengujian Fitur Bonus 6: Data Security

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	Password Hashing	Prekondisi: input config valid 1. Masukkan SET_SECURITY PASSWORD;	config-1;	Seluruh password berhasil di hash dan tidak bisa kembali lagi	PASSWORD HASHING [DONE] Enabling password hashing [SUCCESS] Password hashing enabled Migrating existing passwords... Done [SUCCESS] Security enhanced! All passwords are now hashed using FNV-1a 32-bit algorithm. Note: Remember to SAVE; to persist this change. ***CONTOH HASIL PASSWORD HASH*** user_id,username,password,karma,created_at USER001,kebinganteng,3975399963,48,2025-11-01 21:10:00 USER002,bob,356875903,7,2025-11-02 21:10:00 USER003,drnefario,4009441878,99,2025-11-02 21:10:00 USER004,minion_dave,462869840,0,2025-11-03 21:10:00 USER005,minion_aved,1419334880,0,2025-11-03 21:10:00 USER006,minion_veda,2546057820,0,2025-11-03 21:10:00 USER007,alice,821948264,1250,2025-10-31 21:10:00
2.	File Encryption	Prekondisi: input config valid, konfigurasi file encryption off 1. Masukkan SET_SECURITY FILE ON;	config-1;	Seluruh file menjadi hex unreadable dari folder config	FILE ENCRYPTION [DONE] Enabling file encryption [SUCCESS] File encryption enabled Using LCG keystream cipher Current seed: 123456789 [SUCCESS] File encryption activated! All future SAVE operations will encrypt CSV files. ***CONTOH HASIL ISI FILE ENCRYPTION*** GRENC��%�6-�O$;=�+��璦 �"��TV�eUn�FO��Q�Y��/j���gP�p#�Œ=�w9c}h �F/&w�GB����g*W�L��

No.	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
3.	File Decryption	Prekondisi: input config valid, konfigurasi file encryption on 1. Masukkan SET_SECURITY FILE OFF;	config-1;	Seluruh file yang awalnya hex unreadable menjadi plaintext kembali	FILE ENCRYPTION [DONE] Disabling file encryption [SUCCESS] File encryption disabled [WARNING] Security reduced! Future SAVE operations will store files in plain text. Note: Remember to SAVE; to persist this change.
4.	Mengubah seed melalui input pengguna file encryption	Prekondisi: input config valid 1. Masukkan SET_SECURITY ENC_SEED 6767;	config-1;	Encryption seed berubah sesuai input	ENCRYPTION SEED [DONE] Updating encryption seed Previous seed: 123456789 New seed: 6767 [SUCCESS] Encryption seed updated! LCG keystream will use the new seed: 6767 Warning: Files encrypted with different seeds cannot be decrypted! Note: Remember to SAVE; to persist this change.
5.	Mengubah seed dengan invalid input	Prekondisi: input config valid 1. Masukkan SET_SECURITY ENC_SEED a;	config-1;	Encryption seed tidak berubah karena input invalid	[ERROR] Invalid seed value Seed must be a positive number. Example: SET_SECURITY ENC_SEED 987654321;

8 Pembagian Kerja dalam Kelompok

Tabel Pembagian Kerja ADT

No	Nama ADT	Penanggungjawab	NIM
1	ADT Sederhana	Kevin	13524019
2	List Berkait	Jordhy	13524026
3	Mesin Karakter atau Mesin Kata	Kevin	13524019
4	Tree	Jonathan	13524023
5	Graph	Samuelson	13524001
6	Trie	Jonathan	13524023
7	Heap	Samuelson	13524001

Tabel Pembagian Kerja Fitur Wajib

No	Nama Fitur	Penanggungjawab	NIM
1	Inisialisasi	Kevin	13524019
2	Perintah	Kevin	13524019
3	Pengguna	Haris	13524053
4	Profil	Jonathan	13524023
5	Post	Jonathan	13524023
6	Subgroddit	Kevin	13524019
7	Comment and Reply	Haris	13524053
8	Voting	Jordhy	13524026
9	Social	Samuelson	13524001
10	Save & Load	Kevin	13524019

Tabel Pembagian Kerja Fitur Bonus

No	Nama Fitur	Penanggungjawab	NIM
11	Feed	Samuelson	13524001
12	Trending Analysis	Jordhy	13524026
13	Advanced Search	Jonathan	13524023
14	Following Recommendation	Samuelson	13524001
15	Content Moderation	Haris	13524053
16	Data Security	Haris	13524053
17	Kreativitas	Jordhy	13524026

9 Lampiran

9.1 *Deskripsi Tugas Besar* 1

Spesifikasi Tugas Besar IF2110 Algoritma dan Pemrograman 2 2025/2026

9.2 *Notulen Rapat*

IF2110_FormAsistensiTB_1_K01-I.pdf

9.3 *Log Activity Anggota Kelompok*

Tanggal	Aktivitas	Penanggungjawab
14/11/2025	Inisialisasi dan implementasi ADT sederhana, mesinkarakter, mesinkata, dan utils	13524019 Kevin Wirya …
15/11/2025	Implementasi ADT ListBerkait	13524026 Made Branen…
15/11/2025	Implementasi Fitur Pengguna	13524053 Muhammad …
15/11/2025	Implementasi ADT Graph dan Progress feat:Social	13524001 Samuelson D.…
16/11/2025	Implementasi ADT Tree	13524023 Jonathan Kris …
19/11/2025	Implementasi fitur Profil	13524023 Jonathan Kris …
20/11/2025	Implementasi fitur Post	13524023 Jonathan Kris …
20/11/2025	Implementasi fitur Subgroddit	13524019 Kevin Wirya …
22/11/2025	Finalisasi implementasi fitur Social	13524001 Samuelson D.…
22/11/2025	Implementasi fitur Voting	13524026 Made Branen…
23/11/2025	Implementasi fitur Voting	13524026 Made Branen…
24/11/2025	Implementasi fitur Comment and Reply	13524053 Muhammad …
29/11/2025	Implementasi fitur bonus Feed	13524001 Samuelson D.…
30/11/2025	Implementasi fitur bonus Following Recommendations	13524001 Samuelson D.…
30/11/2025	Perbaikan pada fitur Subgroddit	13524019 Kevin Wirya …
03/12/2025	Implementasi ADT Trie	13524023 Jonathan Kris …
03/12/2025	Implementasi fitur bonus Trending Analysis	13524026 Made Branen…
03/12/2025	Implementasi fitur bonus Data-Security	13524053 Muhammad …

Tanggal	Aktivitas	Penanggungjawab
04/12/2025	Implementasi fitur bonus Content-Moderation	13524053 Muhammad …
04/12/2025	Implementasi bonus Kreativitas	13524026 Made Branen…
04/12/2025	Implementasi fitur bonus Advanced Search	13524023 Jonathan Kris …