Aplikasi CLI Growspace: Manajemen Workshop dengan Queue, Hasing, dan Fitur Undo

*Growspace CLI Application: Workshop Management with Queue, Hashing, and Undo Feature*

Mutia Saniya Rahma(G6401231002)[[1]](#footnote-0), Hanifah Syahidah (G6401231067)[[2]](#footnote-1), Muhammad Allif Qalbiy(G6401231084)[[3]](#footnote-2), Kasyifa Naila Alisha(G6401231139)[[4]](#footnote-3)

**Abstrak**

Pengelolaan pendaftaran workshop dan pelatihan secara manual seringkali tidak efisien dan rentan terhadap inkonsistensi data. Aplikasi *Growspace* dikembangkan sebagai solusi manajemen workshop berbasis CLI untuk mengatasi permasalahan tersebut melalui fitur yang terstruktur. Proyek ini mengimplementasikan struktur data inti: Queue kustom berbasis linked list digunakan untuk antrian verifikasi pendaftaran. Stack kustom berbasis std::vector mendukung fitur undo aksi admin, Hashing melalui std::unordered\_map mempercepat pencarian data event dan registrasi, dan std::sort dengan custom comparator digunakan untuk pengurutan daftar event. Hasilnya adalah aplikasi CLI fungsional yang memungkinkan peserta melihat dan mendaftar event, sementara admin dapat mengelola event, memverifikasi pendaftaran, dan membatalkan aksi terakhir secara interaktif. Implementasi struktur data yang tepat terbukti meningkatkan efisiensi, responsivitas, dan integrasi data dalam pengelolaan workshop.

Kata Kunci: Aplikasi CLI, Hashing, Manajemen Workshop, Queue, Sorting, Stack

*Manual management of workshop registrations and execution is often inefficient and prone to data inconsistencies. The Growspace application was developed as a CLI-based workshop management solution to address these issues through structured features. This project implements core data structures: a custom linked-list-based Queue is used for the registration verification queue, a custom std::vector-based Stack supports the admin's undo action feature, Hashing via std::unordered\_map accelerates event and registration data lookup, and std::sort with a custom comparator is used for sorting event lists. The result is a functional CLI application allowing participants to view and register for events, while administrators can manage events, verify registrations, and interactively undo their last actions. The proper implementation of these data structures demonstrably improves efficiency, responsiveness, and data integrity in workshop management.*

*Keywords: CLI Application, Hashing, Queue, Sorting, Stack, Workshop Management*

# BAB 1

# PENDAHULUAN

**1.1 Latar Belakang**

Workshop dan pelatihan merupakan sarana penting dalam meningkatkan kompetensi baik di lingkungan akademik maupun profesional. Kegiatan ini memberi kesempatan untuk pembelajaran yang praktis, terfokus, dan interaktif.

Namun, dalam implementasinya, proses administrasi dan manajemen workshop yang dilakukan secara manual seringkali menjadi sumber berbagai kendala. Permasalahan umum yang muncul meliputi kesulitan dalam pelacakan data pendaftar beserta statusnya (misalnya pembayaran), potensi antrian pendaftaran yang tidak terorganisir yang dapat berujung pada kelebihan kuota, serta tantangan dalam penyebaran informasi detail acara secara merata dan konsisten. Lebih lanjut, input data manual sangat rentan terhadap kesalahan manusia (*human error*) yang dapat menyebabkan inkonsistensi data.

Kompleksitas permasalahan tersebut menggarisbawahi urgensi akan sebuah solusi sistem manajemen terkomputerisasi yang mana tidak hanya dapat meningkatkan efisiensi dan akurasi dalam pengelolaan workshop secara drastis, tetapi juga memberikan kemudahan dan kenyamanan lebih bagi penyelenggara dalam mengelola event dan bagi peserta dalam melakukan pendaftaran serta mendapatkan informasi.

**1.2 Solusi yang Ditawarkan**

Menjawab kebutuhan tersebut, dikembangkanlah "Growspace", sebuah aplikasi manajemen workshop dan pelatihan yang dioperasikan melalui Antarmuka Baris Perintah (CLI). Pemilihan platform CLI dilandasi alasan efisiensi, kecepatan akses tanpa antarmuka grafis yang berat, serta sebagai tantangan akademis untuk membangun sistem dari nol.

Growspace dirancang untuk menyediakan serangkaian fitur yang mendukung keseluruhan siklus manajemen workshop. Untuk peserta, aplikasi ini menyediakan fasilitas untuk melihat daftar event, melakukan pendaftaran, dan memeriksa status pendaftaran. Bagi administrator, Growspace menawarkan kemampuan untuk mengelola data event (membuat dan menghapus), melakukan verifikasi pendaftaran yang masuk dalam antrean, serta fitur penting berupa pembatalan aksi terakhir (undo) untuk meningkatkan fleksibilitas dan mengurangi risiko kesalahan.

* Aplikasi Sejenis

Beberapa platform populer seperti *Eventbrite* dan *Ticketmaster* sudah menyediakan layanan manajemen event berbasis web dengan fitur sangat lengkap, termasuk sistem pembayaran dan pemasaran. Selain itu, beberapa solusi open-source seperti *Attendize* atau *Open Conference Systems* (OCS) juga menawarkan fungsionalitas serupa dengan kompleksitas yang bervariasi. Berbeda dengan mereka, *Growspace* berfokus pada implementasi CLI yang ringan dan ditulis menggunakan C++ murni. Aplikasi ini tidak ditujukan untuk bersaing dalam jumlah fitur, melainkan sebagai studi akademis untuk memahami dan menerapkan konsep manajemen event dengan struktur data dasar secara efisien.

* Referensi

Pengembangan *Growspace* mengacu pada referensi klasik di bidang algoritma dan struktur data, seperti Introduction to Algorithms (Cormen et al., 2009) dan Data Structures and Algorithm Analysis in C++ (Weiss, 2013). Struktur data yang digunakan meliputi:

1. Queue Kustom: Diimplementasikan menggunakan struktur linked list untuk mengelola antrean pendaftar berdasarkan prinsip FIFO (First-In, First-Out). Operasi enqueue, dequeue, dan enqueueFront (untuk kebutuhan undo) dirancang untuk memiliki kompleksitas waktu O(1).
2. Stack Kustom: Diimplementasikan menggunakan std::vector sebagai kontainer dasar untuk mendukung fitur "Undo" bagi administrator. Operasi push dan pop yang merupakan inti dari Stack juga memiliki kompleksitas waktu O(1).
3. Hashing: Memanfaatkan std::unordered\_map dari C++ Standard Library. Ini memungkinkan pencarian dan pengaksesan data event maupun data registrasi berdasarkan ID unik dengan kompleksitas waktu rata-rata O(1). Fungsi hash juga diterapkan untuk penyimpanan password pengguna secara lebih aman.
4. Sorting: Menggunakan algoritma std::sort dari C++ Standard Library, yang dikombinasikan dengan custom comparator (fungsi lambda), untuk menyajikan daftar event kepada pengguna dalam urutan yang logis dan mudah dicerna (misalnya, berdasarkan tanggal atau judul) dengan kompleksitas waktu O(NlogN), yang merupakan standar efisien untuk algoritma pembandingan.

**1.3 Tujuan**

Berdasarkan latar belakang masalah yang telah diuraikan dan landasan teori yang digunakan, tujuan dari proyek tugas akhir ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan aplikasi manajemen workshop dan pelatihan berbasis CLI bernama "Growspace" menggunakan bahasa pemrograman C++.
2. Menerapkan struktur data Queue kustom yang efisien untuk pengelolaan antrean pendaftaran dan proses verifikasi peserta.
3. Mengimplementasikan struktur data Stack kustom untuk menyediakan fungsionalitas undo yang interaktif dan robust bagi tindakan administrator.
4. Memanfaatkan teknik Hashing melalui std::unordered\_map untuk optimasi pencarian, sekaligus menerapkan hashing pada password untuk keamanan dasar.
5. Mengaplikasikan algoritma Sorting standar (std::sort) untuk penyajian daftar event yang terstruktur dan informatif bagi pengguna.
6. Menghasilkan sebuah aplikasi CLI yang fungsional, efisien dalam penggunaan sumber daya, dan mampu mempermudah proses manajemen workshop secara keseluruhan.

# BAB 2

# **METODE**

Pada bab ini, akan diuraikan secara komprehensif metodologi yang diterapkan dalam perancangan dan pengembangan aplikasi Growspace. Setiap tahapan metode dideskripsikan dengan tujuan untuk memberikan pemahaman yang jelas mengenai proses rekayasa perangkat lunak yang dilakukan, sehingga memungkinkan pembaca untuk memahami dasar pengambilan keputusan desain dan, secara teoritis, mereplikasi pendekatan yang digunakan.

**2.1. Ruang Lingkup Proyek**

Aplikasi Growspace merupakan sistem manajemen workshop dan pelatihan berbasis Command Line Interface (CLI) yang dikembangkan untuk memenuhi kebutuhan pengelolaan kegiatan secara terstruktur bernama “Growspace”. Dibangun menggunakan bahasa pemrograman C++ dengan standar C++17, sistem ini mengandalkan penyimpanan data persisten berbentuk file teks berformat Comma-Separated Values (CSV) yang beroperasi secara mandiri tanpa memerlukan sistem database eksternal. Fokus utama pengembangan terletak pada tiga aspek krusial: perancangan, implementasi, dan demonstrasi penggunaan struktur data fundamental. Secara spesifik, sistem ini mengimplementasikan Queue, Stack, Hashing, juga algoritma Sorting. Dari sisi fitur, Growspace menyediakan dua level akses pengguna:

1. Fungsi peserta mencakup kemampuan melihat daftar event dengan fitur filter dan sorting, melakukan pendaftaran event, serta memeriksa status pendaftaran
2. Fungsi administrator meliputi sistem login terproteksi, manajemen event (meliputi pembuatan dan penghapusan event dengan logika pencegahan apabila sudah terdapat pendaftar), proses verifikasi pendaftaran peserta melalui sistem antrean, dan fitur "Undo" yang memungkinkan pembatalan aksi administratif terakhir

Sebagai bagian dari desain inti, aplikasi ini sengaja dikembangkan tanpa antarmuka pengguna grafis (GUI) untuk mempertahankan fokus pada optimasi struktur data dan efisiensi sistem secara keseluruhan.

**2.2 Metode Pengembangan Perangkat Lunak**

Pengembangan Growspace menggunakan pendekatan iteratif yang berfokus pada penyempurnaan bertahap melalui siklus-siklus berulang. Setiap siklus diawali dengan analisis kebutuhan fitur secara mendalam untuk menentukan spesifikasi teknis yang diperlukan. Kemudian dilanjutkan dengan perancangan model data yang mengoptimalkan struktur data fundamental dan logika layanan yang efisien. Tahap implementasi dilakukan dengan penulisan kode mengikuti standar C++17, diikuti pengujian unit dan integrasi menyeluruh untuk setiap modul utama guna memastikan fungsionalitas bekerja sesuai harapan. Pendekatan ini memberikan fleksibilitas tinggi dalam melakukan penyesuaian dan perbaikan selama proses pengembangan berlangsung. Setiap iterasi memungkinkan evaluasi menyeluruh terhadap fitur yang sedang dikembangkan, sekaligus mempertahankan stabilitas sistem secara keseluruhan. Mekanisme pengujian berkala dan proses iteratif yang ketat memastikan kualitas aplikasi tetap terjaga, sementara tetap memungkinkan adaptasi terhadap kebutuhan pengembangan yang mungkin berubah.

**2.2. Implementasi Aplikasi Growspace**

Pada bagian ini akan diuraikan langkah-langkah teknis utama dalam membangun aplikasi Growspace, meliputi implementasi struktur data kustom dan penerapan struktur data serta algoritma dari pustaka standar C++.

**2.2.1. Sub Metode 1: Implementasi Struktur Data Kustom**

1. Queue Kustom

Struktur data Queue kustom diimplementasikan untuk mengelola antrean pendaftaran yang menunggu verifikasi oleh administrator (\_verifQueue dalam AdminService). Implementasi ini menggunakan struktur data singly linked list untuk memastikan efisiensi operasi. Setiap elemen dalam antrean direpresentasikan oleh sebuah Node yang berisi data (objek Registration) dan pointer ke Node berikutnya. Pengelolaan antrean dilakukan melalui pointer head (menunjuk ke elemen terdepan) dan tail (menunjuk ke elemen terakhir), serta variabel current\_size untuk melacak jumlah elemen. Langkah-langkah implementasi dan operasi kunci meliputi:

1. Definisi Struktur Node: Pembuatan struct Node internal yang menyimpan data bertipe T dan pointer next ke node berikutnya.
2. Inisialisasi Queue: Constructor kelas Queue menginisialisasi pointer head dan tail menjadi nullptr, serta current\_size menjadi 0, menandakan antrean kosong.
3. Operasi enqueue(const T& item): Penambahan elemen baru dilakukan di akhir antrean (pada tail). Node baru dibuat dan pointer tail->next (jika antrean tidak kosong) atau head (jika antrean kosong) diarahkan ke node baru tersebut, kemudian tail diperbarui. Operasi ini memiliki kompleksitas waktu O(1).
4. Operasi dequeue(): Penghapusan elemen dilakukan dari depan antrean (dari head). Pointer head dipindahkan ke node berikutnya dan node head lama dihapus dari memori. Jika setelah dequeue antrean menjadi kosong, tail juga diatur menjadi nullptr. Operasi ini memiliki kompleksitas waktu O(1).
5. Operasi enqueueFront(const T& item): Operasi khusus ini ditambahkan untuk mendukung fitur undo pada proses verifikasi pendaftaran. Elemen baru ditambahkan di depan antrean dengan menjadikan node baru sebagai head baru dan mengarahkan pointer next-nya ke head lama. Operasi ini juga memiliki kompleksitas waktu O(1).
6. Operasi pendukung lainnya seperti front() (mengakses elemen terdepan), isEmpty() (memeriksa kekosongan antrean), dan size() (mengembalikan jumlah elemen) juga diimplementasikan dengan kompleksitas waktu O(1).
7. Stack Kustom

Struktur data Stack kustom digunakan untuk mendukung fungsionalitas "Undo Aksi Terakhir" bagi administrator. Stack ini, yang dinamakan \_actionStack dalam AdminService, menyimpan objek UndoRecord yang mencatat detail dari setiap aksi yang bisa dibatalkan. Implementasi Stack ini memanfaatkan std::vector dari C++ Standard Library sebagai kontainer penyimpan data dasarnya, dikarenakan efisiensi std::vector dalam operasi penambahan dan penghapusan elemen di akhir. Tahapan implementasi dan operasi utama adalah sebagai berikut:

1. Penggunaan std::vector sebagai Penyimpanan Internal: Kelas Stack memiliki sebuah private member std::vector<T> elements untuk menyimpan elemen-elemen tumpukan.
2. Operasi push(const T& item): Menambahkan elemen baru ke atas tumpukan diimplementasikan dengan memanggil metode push\_back() dari objek std::vector internal. Operasi ini memiliki kompleksitas waktu ammortized O(1).
3. Operasi pop(): Menghapus elemen teratas dari tumpukan diimplementasikan dengan memanggil metode pop\_back() dari objek std::vector internal. Operasi ini memiliki kompleksitas waktu O(1).
4. Operasi lainnya seperti top() (mengakses elemen teratas menggunakan std::vector::back()), isEmpty() (menggunakan std::vector::empty()), dan size() (menggunakan std::vector::size()) juga diimplementasikan dengan memanfaatkan fungsi anggota std::vector yang relevan, sehingga memiliki kompleksitas waktu O(1).

**2.2.2. Sub Metode 2: Penggunaan Struktur Data & Algoritma dari Library C++**

1. Hashing (std::unordered\_map dan std::hash)
2. Hashing Password: Sebuah fungsi utilitas hashPassword(const std::string& password) dikembangkan untuk mengubah password yang dimasukkan pengguna menjadi sebuah nilai hash. Implementasi ini menggunakan std::hash<std::string> dari C++ Standard Library untuk menghasilkan nilai hash numerik, yang kemudian dikonversi menjadi representasi string heksadesimal untuk konsistensi penyimpanan. Hash password ini disimpan dalam file CSV (untuk User dan Registration) dan digunakan saat proses verifikasi login, sehingga password asli pengguna tidak pernah disimpan secara langsung.
3. Optimasi Pencarian Data: Untuk mempercepat operasi pencarian data Event dan Registration berdasarkan ID uniknya, struktur data std::unordered\_map dari C++ Standard Library digunakan sebagai mekanisme indeks internal. Di dalam ParticipantService terdapat \_eventIndex dan \_registrationIndex, sementara di AdminService terdapat \_adminEventIndex dan \_adminRegistrationIndex. Map ini memetakan ID (bertipe int) ke pointer dari objek Event atau Registration yang tersimpan dalam std::vector utama. Penggunaan std::unordered\_map memungkinkan operasi pencarian (melalui metode .find()) dilakukan dengan kompleksitas waktu rata-rata O(1). Pembuatan dan pembaruan indeks ini dikelola oleh fungsi buildIndexes() (di ParticipantService) atau buildAdminIndexes() (di AdminService), yang dipanggil saat inisialisasi service dan setelah operasi yang memodifikasi jumlah total elemen data (seperti penambahan atau penghapusan event). (Disarankan untuk menyertakan contoh deklarasi salah satu std::unordered\_map di kelas service dan contoh singkat penggunaan metode .find() dalam fungsi findEventById()).d pengguna dan untuk optimasi pencarian data.
4. Sorting (std::sort)

Untuk menyajikan daftar event kepada pengguna secara terstruktur dan mudah dinavigasi, aplikasi Growspace memanfaatkan algoritma sorting. Implementasi sorting menggunakan fungsi std::sort dari C++ Standard Library, yang pada umumnya memiliki kompleksitas waktu rata-rata O(NlogN), di mana N adalah jumlah event yang diurutkan.

Fleksibilitas pengurutan berdasarkan berbagai kriteria seperti ID event, judul, tanggal mulai, atau sisa kuota dicapai dengan menyediakan lambda function sebagai argumen custom comparator pada pemanggilan fungsi std::sort. Fitur pengurutan event ini diimplementasikan dalam metode ParticipantService::listEvents().

**2.2.3 Sub Metode 3: Desain Fitur Utama**

Struktur data yang telah diimplementasikan berperan penting dalam alur kerja fitur-fitur inti aplikasi.

1. Alur Pendaftaran Peserta dan Verifikasi Admin

Proses pendaftaran peserta dimulai dengan pembuatan objek Registration yang disimpan dalam std::vector<Registration> (\_registrations). Jika event memerlukan verifikasi, salinan objek juga dimasukkan ke \_verifQueue, sebuah antrean berbasis linked list yang dikelola AdminService. Indeks pencarian diperbarui via buildIndexes().

Admin memproses pendaftar melalui front() dan memperbarui status registrasi sesuai hasil verifikasi, lalu mengeluarkannya dari antrean (dequeue). Jika terjadi kesalahan, fitur undo memungkinkan admin mengembalikan status registrasi ke kondisi sebelumnya. Bila status kembali PENDING, objek dimasukkan kembali ke depan antrean menggunakan enqueueFront(). Setiap aksi verifikasi dicatat dalam \_actionStack sebagai UndoRecord.

1. Alur Manajemen Event oleh Admin (Create, Delete)

Admin dapat membuat dan menghapus event. Saat event dibuat, objek Event ditambahkan ke \_events dan indeks diperbarui. Aksi ini dicatat dalam \_actionStack.

Sebelum event dihapus, sistem mengecek apakah ada peserta terdaftar. Jika ada, penghapusan dibatalkan. Jika tidak, objek event diserialisasi dan disimpan dalam UndoRecord, lalu dihapus dari \_events.

1. Fungsionalitas Undo Admin

Fitur undo berlaku untuk semua aksi modifikatif (verifikasi, create, delete). Setiap aksi menghasilkan UndoRecord yang berisi ActionType, target\_id, dan pre\_state. Record ini disimpan di \_actionStack. Saat undo dilakukan, peekLastActionDescription() digunakan untuk menampilkan deskripsi aksi, dan bila dikonfirmasi, data dikembalikan ke kondisi awal. Untuk VERIFY\_REGISTRATION, status lama dipulihkan dari string gabungan. Untuk DELETE\_EVENT, objek event dikembalikan melalui proses deserialisasi.

# BAB 3

# **HASIL DAN PEMBAHASAN**

**3.1 Tingkat Kerumitan Masalah dan Struktur Data yang Diimplementasikan**

Pada aplikasi Growspace, kami menghadirkan solusi manajemen kegiatan yang menghubungkan antara pengguna dan penyelenggara event. Aplikasi ini dibuat untuk mempermudah kebutuhan pengguna dalam menemukan dan mengikuti berbagai kegiatan, juga membantu pihak penyelenggara event dalam meraih kesuksesan acaranya, baik dari segi jumlah peserta maupun keuntungan yang diperoleh. Growspace mengelola dua peran utama, yaitu admin dan peserta. Admin memiliki akses untuk menambahkan event ke dalam sistem sebagai data dinamis, yang nantinya akan muncul pada sisi peserta sebagai informasi kegiatan yang bisa diikuti. Selain itu, admin juga memiliki fitur untuk memverifikasi data peserta maupun event yang masuk, sehingga pengelolaan acara menjadi lebih tertata dan profesional.

Pada pengembangan aplikasi CLI Growspace, digunakan berbagai struktur data untuk memastikan pengolahan data berjalan secara efisien. Salah satunya adalah struktur data antrian (queue kustom) yang digunakan untuk mengatur urutan pendaftaran workshop atau peserta berdasarkan prinsip *First In First Out*. Berarti, peserta yang mendaftar lebih awal akan diproses terlebih dahulu, dan proses ini berjalan dalam waktu konstan secara rata-rata O(1). Selanjutnya struktur data hash table yang dimanfaatkan untuk menyimpan data workshop dengan pendekatan key-value, sehingga pencarian, penambahan, maupun penghapusan data dapat dilakukan secara cepat dan efisien. Hash table memungkinkan sistem dengan cepat menemukan detail workshop berdasarkan kunci (misalnya ID atau nama workshop) tanpa perlu memeriksa seluruh daftar. Growspace juga menyediakan fitur *undo* yang memungkinkan pengguna membatalkan setiap riwayat perintah terakhir dan kembali ke kondisi sebelumnya jika diperlukan. Fitur ini dibuat dengan menggunakan struktur data stack yang mengikuti prinsip *Last In First Ou*t, di mana data terakhir yang masuk akan menjadi yang pertama diproses kembali untuk di-*undo*. Stack untuk fitur undo memastikan bahwa setiap perintah berurutan dapat dibatalkan dengan simpel, sehingga kesalahan pengguna dapat dikoreksi. Selain itu, penyortiran data juga menjadi bagian penting dalam tampilan daftar workshop. Growspace menerapkan algoritma penyortiran seperti *Quick Sort* atau *Merge Sort* agar data dapat ditampilkan dalam urutan tertentu, baik berdasarkan abjad, waktu pelaksanaan, maupun kategori tertentu. Dengan kompleksitas yang efisien, penyortiran ini memastikan pengguna dapat dengan mudah menemukan workshop yang mereka butuhkan. Secara keseluruhan, pemilihan struktur data ini menyeimbangkan kerumitan masalah dan efisiensi operasional, menjaga agar aplikasi dapat menangani skala data yang meningkat dengan tingkat performa yang baik.

**3.1 Hasil Implementasi dan *Workflow* Sistem**

Berikut hasil implementasi antarmuka CLI Growspace juga *workflow system* berdasarkan peran pengguna, yaitu Peserta dan Admin. Setiap pengguna berinteraksi dengan sistem melalui perintah CLI, yang kemudian diproses untuk memanipulasi data internal. Data peserta dan kegiatan disimpan dalam struktur data terstruktur seperti array atau list, memungkinkan akses cepat berdasarkan indeks atau kunci. Sistem memberikan respon tertulis untuk setiap perintah yang dieksekusi, seperti konfirmasi operasi berhasil atau daftar informasi yang diminta.

**3.1.1 Implementasi dan *Workflow* Sistem bagi Peserta**

Peserta berinteraksi dengan sistem menggunakan perintah CLI. Peserta memulai sesi dengan memasukkan *input* 1. Lihat dan daftar event, untuk id peserta akan muncul ketika pengguna memulai pendaftarannya. Untuk step selanjutnya adalah peserta memiliki beberapa opsi pilihan yang akan saya cantumkan pada tabel perintah dan operasi sistem yang terjadi:

| Perintah Input | Operasi Sistem | Keterangan sistem |
| --- | --- | --- |
| List Event yang Tersedia | kata kunci pencarian event -> kata untuk pengurutan -> urutan asc/desc->filter&sort dari events.csv->tampilan hasil tabel | * Jika tidak ada event tampilan error * pengurutan Quick sort (efisiensi O(n log n)) * peserta melihat detail event dengan ID |
| Daftar ke Event | mengisikan event ID->Validasi : ketersediaan event & quota *check* ->konfirmasi(y/n)->simpan data ke regristation.csv (stat PENDING) | * jika quota habis/tidak ditemukan,error * Pendaftaran masuk ke antrean verifikasi (Queue) |
| Cek Status Pendaftaran | mengisikan : regristation ID, email, password->validasi kecocokan data di regristation.csv->tampilkan detail (status & pembayaran) | * Email/password salah : error * Status : PENDING/APPROVED/REJECTED * Pembayaran: VERIFIED/UNVERIVIED |
| Keluar dari Menu Peserta | Kembali ke menu utama |  |

**3.1.2 Implementasi dan *Workflow* Sistem bagi Admin**

Admin memiliki akses yang lebih luas dibandingkan dengan peserta, termasuk fitur verifikasi dan manajemen data. Setelah login pada tahap awal, admin selanjutnya dapat memilih beberapa opsi, berikut tercantum dalam tabel perintah dan operasi sistem yang terjadi pada admin:

| Perintah Input | Operasi Sistem | Keterangan sistem |
| --- | --- | --- |
| Verifikasi Pendaftaran | tampilan antrian pendaftaran (Queue) dari *registration.csv* (stat PENDING)->memilih : APPROVED/REJECT-> stat terupdate di *registration.csv->*Undo (jika diperlukan) | * menggunakan struktur data Queue (FIFO) * jika approved : quota event berkurang |
| Manajement Event | * Tambah event->data baru terinput ke *events.csv* * Hapus event berdasarkan event\_id : validasi tidak ada peserta terdaftar   Undo (jika diperlukan) | * Validasi input : format tanggal (YYYY-MM–DD) * Operasi CRUD dicatat di stack |
| Logout | keluar dari sesi admin | Kembali ke menu login |

**3.2 Analisis Kompleksitas dan Efisiensi**

Sebagai bagian dari pengembangan aplikasi CLI Growspace, analisis kompleksitas dan efisiensi sistem menunjukkan bahwa sebagian besar proses yang dijalankan mampu bekerja secara optimal. Setiap peserta yang mendaftar akan diproses menggunakan struktur data antrian (queue) dengan prinsip *First In First Out*, sehingga peserta yang mendaftar lebih awal akan diproses terlebih dahulu. Proses enqueue maupun dequeue ini berlangsung dalam waktu konstan rata-rata O(1), yang artinya tidak tergantung pada jumlah data yang sedang ada di dalam antrian. Fitur undo yang ditawarkan Growspace juga dirancang untuk efisiensi. Dengan menggunakan struktur data stack, setiap perintah yang masuk dapat disimpan dan kemudian dibatalkan dengan operasi push dan pop, yang juga berjalan dalam waktu rata-rata O(1). Hal ini memungkinkan pengguna membatalkan kesalahan input dengan cepat dan menjaga alur interaksi tetap lancar. Selanjutnya, hash table dimanfaatkan untuk menyimpan dan mengakses data workshop dengan cara yang cepat melalui pendekatan key-value. Pencarian, penyisipan, dan penghapusan data dalam struktur ini rata-rata berjalan dengan kompleksitas Θ(1). Meskipun pada kondisi tertentu seperti terjadinya banyak *clash* kunci performa bisa menurun menjadi O(n), namun hal ini sangat jarang terjadi jika penanganan *clash* dilakukan dengan baik. Penyortiran data juga menjadi bagian penting dalam tampilan daftar workshop. Growspace menerapkan algoritma seperti Quick Sort atau Merge Sort yang memiliki kompleksitas rata-rata O(n log n), tergantung pada jumlah data yang ingin diurutkan. Walaupun kompleksitasnya lebih tinggi dibanding struktur data lainnya, fungsi penyortiran hanya dijalankan saat pengguna mengakses daftar yang terurut, sehingga tidak mempengaruhi performa sistem secara keseluruhan. Dari sisi memori, penggunaan struktur queue, hash table, dan stack masing-masing menyimpan hingga n data, yang berarti kompleksitas ruangnya O(n).

Secara keseluruhan, kombinasi struktur data yang digunakan dalam Growspace telah menciptakan sistem yang efisien untuk penggunaan sehari-hari, dan memungkinkan aplikasi tetap berjalan lancar bahkan ketika menangani puluhan hingga ratusan workshop secara bersamaan. Berikut adalah ringkasan kompleksitas waktu rata-rata dari setiap struktur data yang digunakan dalam aplikasi:

| Struktur Data | Operasi Utama | Kompleksitas Waktu (rata-rata) |
| --- | --- | --- |
| Queque kustom | enqueue / dequeue | O(1) |
| Stack (*Undo)* | push / pop | O(1) |
| Hash table | insert/ search / delete | Θ(1) (Amortized) |
| Sort (*Quick/Merge*) | sort ( n element) | O(n log n) (rata-rata) |

|  |  |  | |  | |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**3.3. Kelebihan dan Kekurangan Produk**

**3.1. Kelebihan**

Growspace menggunakan struktur data klasik yang membuat operasi manajemen workshop berjalan dengan efisien. Queue menjaga proses berjalan adil dengan waktu konstan, hash table mempercepat pencarian data workshop, dan stack undo meningkatkan keandalan sistem dengan memungkinkan pembatalan kesalahan tanpa mengurangi kecepatan. Aplikasi CLI-nya ringan dan tidak membutuhkan antarmuka grafis, sehingga bisa dijalankan di berbagai lingkungan. Kemudian juga adanya fitur sorting yang diimplementasikan juga membuat tampilan data lebih informatif. Dengan pemilihan struktur data yang tepat, aplikasi ini tetap cepat dan responsif.

**3.2. Kekurangan**

Sebagai aplikasi CLI, antarmukanya kurang ramah untuk pengguna non-teknis dibandingkan GUI. Meskipun Growspace menggunakan file CSV untuk penyimpanan data, sistem ini belum memiliki mekanisme backup otomatis dan manajemen data yang lebih robust seperti sistem database profesional. Ketika data sangat besar, bisa terjadi *clash* di hash table yang memperlambat pencarian (kasus terburuk O(n)). Proses sorting juga bisa menjadi kendala jika daftar workshop tumbuh sangat besar, meski jarang terjadi di skala normal. Kapasitas stack undo yang terbatas pada memori juga bisa menjadi masalah untuk operasi undo yang sangat banyak. Namun, semua keterbatasan ini wajar untuk versi awal dan bisa diperbaiki di pengembangan selanjutnya, seperti dengan menambahkan penyimpanan persisten dan antarmuka yang lebih baik.

**BAB 4**

**KESIMPULAN**

//EMPTY

# Ucapan Terima Kasih

Puji syukur kehadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga laporan penelitian dengan judul "Manajemen Workshop / Pelatihan" ini dapat diselesaikan dengan baik.

Penyusunan laporan ini tidak lepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, dengan segala kerendahan hati, peneliti mengucapkan terima kasih yang tulus kepada:

1. **Dr.Eng. Annisa S.Kom., M.Kom.** selaku dosen koordinator mata kuliah struktur data yang telah meluangkan waktu, memberikan bimbingan, arahan, serta masukan yang sangat berharga sejak awal hingga akhir penyusunan laporan ini.
2. **Endang Purnama Giri S.Kom., M.Kom.** selaku dosen pengajar yang telah memberikan dukungan, saran, dan kritik konstruktif demi penyempurnaan laporan penelitian ini.
3. **Sri Arini Ismayasari dan Muhammad Zaky Ghoetti Anand** selaku Asisten Praktikum Mata kuliah struktur data yang telah memberikan fasilitas dan dukungan selama proses penelitian.
4. **Semua pihak** yang tidak dapat disebutkan satu per satu, atas bantuan dan dukungan yang telah diberikan baik secara langsung maupun tidak langsung.

Peneliti menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa mendatang. Semoga laporan penelitian ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan pihak-pihak yang membutuhkan.

[Bogor, 31 Mei 2025]

# Daftar Pustaka

*Format untuk berkala ilmiah:*

Nama penulis 1, 2, 3. Tahun terbit. Judul artikel. *Nama Berkala Ilmiah*. Vol (edisi): halaman.

*Contoh:*

Nefti S, Oussalah M, Kaymak U. 2008. A new fuzzy set merging technique using inclusion-based fuzzy clustering. *IEEE Trans Fuzzy Syst*. 16 (1): 145-161.

*Format untuk berkala ilmiah online:*

Nama penulis 1, 2, 3. Tahun terbit. Judul artikel. *Nama berkala ilmiah* [Internet]. Waktu pembaharuan (jika ada); waktu unduh [diunduh tahun bulan tanggal]; Volume (edisi): lokasi. Catatan (jika ada).

*Contoh:*

Chunyu H, Hwangnam K, Jennifer CH, Dennis C, Sai S N. 2010. A distributed approach of proportional bandwidth allocation for real-time services in ultra wide band (UWB) WPANs. *IEEE/ACM Trans. Parallel and Distributed System* [Internet]. [diunduh 2011 Des 29]; 21(11):1626-1643 <http://doi.ieeecomputersociety.org/10.1109/> TPDS.2010.32.

*Format untuk buku:*

Nama Penulis 1, 2. Tahun terbit. *Judul Buku.* Kota terbit (kode negara): Nama Penerbit.

*Contoh:*

Pressman RS. 2005. *Software Engineering: A Practitioner’s Approach Ed ke-6*. Boston (US): McGraw Hill.

*Format untuk skripsi, tesis, dan disertasi:*

Nama penulis. Tahun terbit. Judul [jenis publikasi]. Kota terbit (kode negara): Nama Perguruan Tinggi.

*Contoh:*

Giri EP. 2008. Model klasifikasi menggunakan *feature* semantik dan *association rules* untuk *biomedical* *image retrieval system* citra *pap smear* [tesis]. Depok (ID): Universitas Indonesia.

*Contoh penyusunan daftar pustaka:*

Azuma RT. 1997. A survey of augmented reality. *Presence-Teleop Virt* 6(4):355-385.

Brown RG. 2004. *Smoothing, Forecasting and Prediction of Discrete Time Series*. New York(US): Dover Publ.

Feiner S, Hollerer T, MacIntyre B, Webster A. 1997. A touring machine: prototyping 3D mobile augmented reality systems for exploring the urban environment. *Pers Tech* 1(4):208-217.

Gotow JB, Zienkiewicz K, White J, Schmidt DC. 2010. Addressing challenges with augmented reality applications on smartphones. Di dalam: Cai Y, Magedanz T, Li M, Xia Jinchun, Gianelli C, editor. *Mobile Wireless Middleware, Operating Systems, and Applications*. New York(US): Springer. hlm 129-143.

Huang Y, Mwemezi JJ. 2011. Optimal facility location on spherical surfaces: algorithm and application. *New York Sci J* 4(7):21-28.

Oh S, Shuster M. 1981. Three-axis attitude determination from vector observations. *J Guid Control* 4(1):70-77.

Paucher R, Turk M. 2010. Location-based augmented reality on mobile phones. Di dalam: *Proceedings of* *Computer Vision and Pattern Recognition Workshops*; San Francisco, 2010 Jun 13-18. Washington(US): IEEE Comp Soc. hlm 9-16.

1. Nama, alamat, dan nomor kontak organisasi penulis pertama; [↑](#footnote-ref-0)
2. Nama, alamat, dan nomor kontak organisasi penulis kedua;

   \*Penulis Korespondensi: Tel/Faks: 0000-00000000; Surel: penuliskorespondensi@ipb.ac.id [↑](#footnote-ref-1)
3. Nama, alamat, dan nomor kontak organisasi penulis pertama; [↑](#footnote-ref-2)
4. Nama, alamat, dan nomor kontak organisasi penulis pertama; [↑](#footnote-ref-3)