

**LAPORAN TUGAS BESAR**  
**IF2111 Algoritma dan Struktur Data STI**


**PURRMART**

Dipersiapkan oleh:

18223109	Fadhil Rifqi Rabbani Pane
18223110	Desati Dinda Saraswati
18223136	Geraldo Linggom Samuel T.
18223125	Matilda Angelina Sumaryo
18222133	Hanan Fitra Salam

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	<b>Sekolah Teknik Elektro dan Informatika ITB</b>	<b>Nomor Dokumen</b>		<b>Halaman</b>
		<i>IF2111-TB-03-03</i>		<i>34</i>
		<i>Revisi</i>	<i>&lt;no revisi&gt;</i>	<i>&lt;Tgl release&gt;</i>

# Daftar Isi

1 Ringkasan.....	4
2 Penjelasan Tambahan Spesifikasi Tugas.....	6
2.1 Perincian spesifikasi pada TEBAK ANGKA.....	6
3 Struktur Data (ADT).....	6
3.1 ADT Kustom.....	6
3.1.1 ADT User.....	6
3.1.2 ADT Barang.....	6
3.2 ADT Stack.....	7
3.3 ADT Set Map.....	7
3.4 ADT Linked List.....	8
4 Program Utama.....	9
5 Algoritma-Algoritma Menarik.....	11
5.1 Makefile.....	11
5.2 Handler.....	11
5.3 Animasi.....	11
6 Data Test.....	12
6.1 PROFILE.....	12
6.2 CART ADD <nama> <n>.....	12
6.3 CART REMOVE <nama> <n>.....	13
6.4 CART SHOW.....	15
6.5 CART PAY.....	16
6.6 HISTORY <n>.....	18
6.7 WISHLIST ADD.....	20
6.8 WISHLIST SWAP <i> <j>.....	20
6.9 WISHLIST REMOVE <i>.....	21
6.10 WISHLIST REMOVE.....	22
6.11 WISHLIST CLEAR.....	23
6.12 WISHLIST SHOW.....	23
6.13 Update STORE LIST.....	24
7 Test Script.....	26
8 Pembagian Kerja dalam Kelompok.....	29
9 Lampiran.....	31
9.1 Deskripsi Tugas Besar.....	31
9.2 Notulen Rapat.....	31
9.2.1 Rapat I.....	31
9.3 Log Activity Anggota Kelompok.....	31
9.4 Notulensi Asistensi.....	32

## Daftar Gambar

Gambar 6.1.1 Test case ketika profile berhasil dijalankan.....	12
Gambar 6.2.1 Test case jika berhasil menambahkan barang ke keranjang.....	13
Gambar 6.2.2 Test case jika tidak ada barang yang ingin ditambah ke keranjang.....	13
Gambar 6.3.1 Test case jika berhasil mengurangi barang dari keranjang.....	14
Gambar 6.3.2 Test case jika jumlah yang ingin dikurangi lebih dari barang yang ada di keranjang.....	14
Gambar 6.3.3 Test case jika tidak ada barang yang valid di keranjang.....	14
Gambar 6.4.1 Test case jika ada barang di dalam keranjang.....	15
Gambar 6.4.2 Test case jika keranjang kosong.....	15
Gambar 6.5.1 Test case jika pembayaran berhasil dilakukan.....	17
Gambar 6.5.2 Test case jika uang pengguna tidak mencukupi untuk melakukan pembayaran.....	17
Gambar 6.5.3 Test case jika pengguna membatalkan pembayaran.....	17
Gambar 6.5.4 Test case jika pengguna memasukkan masukan aneh.....	17
Gambar 6.5.5 Test case jika keranjang pengguna masih kosong.....	17
Gambar 6.6.1 Test case jika riwayat pembelian $N < \text{total riwayat}$ .....	18
Gambar 6.6.2 Test case jika riwayat pembelian $N \geq \text{total riwayat}$ .....	19
Gambar 6.6.3 Test case jika riwayat pembelian kosong.....	19
6.7.1 Test case ketika berhasil menambahkan barang ke wishlist.....	20
6.7.2 Test case ketika mencoba menambahkan barang yang sudah ada di wishlist.....	20
6.7.3 Test case ketika barang tidak ada di list barang.....	20
6.8.1 Test case ketika berhasil menukar posisi barang pada wishlist.....	21
6.8.2 Test case ketika gagal menukar posisi barang pada wishlist.....	21
6.9.1 Test case ketika berhasil menghapus barang dari wishlist menggunakan indeks.....	22
6.9.2 Test case ketika gagal menghapus barang dari wishlist karena memasukkan urutan barang yang tidak ada.....	22
6.9.3 Test case ketika gagal menghapus barang karena wishlist kosong.....	22
6.9.4 Test case ketika gagal menghapus barang karena input tidak valid.....	22
6.10.1 Test case ketika barang berhasil dihapus dari wishlist.....	23
6.10.2 Test case ketika barang gagal dihapus dari wishlist.....	23
6.11.1 Test case ketika wishlist berhasil dikosongkan.....	23
6.12.1 Test case ketika berhasil menampilkan seluruh isi wishlist.....	24
6.12.2 Test case ketika menampilkan isi wishlist kosong.....	24
6.13.1 Test case ketika berhasil menampilkan seluruh isi store list beserta harganya.....	24
6.13.2 Test case ketika menampilkan isi store list kosong.....	25

# 1 Ringkasan

Sebagai tim programmer yang dipercaya oleh OWCA untuk mengatasi permasalahan yang dialami OWCA yaitu tidak memiliki transportasi untuk pergi ke Bojongsoang menghampiri supermarket Borma karena Borma adalah komponen penting agar bisa membawa kemenangan untuk OWCA pada pertarungan sengit melawan Dr. Asep Spakbor. Untuk itu solusi yang paling tepat adalah membuat dan merancang PURRMART, sebuah aplikasi sistem jual beli untuk mempermudah Agen Purry menyelesaikan misi terbarunya menggunakan bahasa C. Pada milestone 2 ini kami tim programmer OWCA menambahkan beberapa fitur pada PURRMART untuk mempermudah dan menambahkan pengalaman User, fitur - fitur tersebut seperti CART dan WISHLIST.

Saat program dijalankan, pengguna akan disambut oleh menu utama yang mencakup berbagai perintah seperti **START**, **LOAD**, **HELP**, dan **QUIT** serta dapat memasukkan perintah lainnya untuk memulai atau melanjutkan aktivitas di PURRMART,.

Aplikasi PURRMART, memiliki beberapa fitur unggulan, seperti memilih pekerjaan untuk mendapatkan uang, menyelesaikan tantangan **Tebak Angka** dan **WORDL3**, selain itu pengguna juga dapat memilih pekerjaan dengan memanfaatkan ADT mesin kata, mesin karakter, serta list. Selain itu, pengguna dapat menambahkan barang ke dalam antrian menggunakan *Queue*, menghapus barang dari antrian, membuat permintaan barang, mengatur harga barang, hingga menyimpan progres aplikasi melalui perintah **SAVE** sebelum keluar dengan perintah **QUIT**. Sistem mendukung pengelolaan barang yang dimasukkan ke dalam keranjang, seperti menambah **CART ADD**, menghapus **CART REMOVE**, melihat isi keranjang **CART SHOW**, dan membayar belanjaan **CART PAY**. Selain itu, fitur wishlist memanfaatkan **ADT linked list**, memungkinkan pengguna untuk menambahkan barang favorit, menghapus barang, menukar posisi barang, atau mengosongkan daftar wishlist dengan mudah. Aplikasi ini juga menyediakan fitur untuk melihat profil pengguna **PROFILE**, menyimpan progres permainan **SAVE**, dan memuat data dari konfigurasi sebelumnya **LOAD**.

Alur aplikasi dimulai dengan membaca file konfigurasi yang menyimpan data toko, pengguna, dan status sebelumnya. Setelah permainan dimulai, pengguna dapat menjalankan berbagai perintah di menu utama, seperti melihat daftar barang di toko **STORE LIST**, mengelola keranjang belanja, atau mengatur wishlist. Pada wishlist, pengguna dapat menambah barang **WISHLIST ADD**, menghapus barang berdasarkan nama atau posisi **WISHLIST REMOVE**, menukar posisi barang **WISHLIST SWAP**, melihat isi wishlist **WISHLIST SHOW**, atau menghapus seluruh isi wishlist **WISHLIST CLEAR**. Fitur wishlist yang didukung oleh ADT linked list memastikan operasi seperti penambahan, penghapusan, dan pengelolaan posisi barang dapat dilakukan dengan efisien. Dengan berbagai fitur yang diimplementasikan menggunakan struktur data seperti list dinamis, queue, dan linked list.

Laporan Tugas Besar pada milestone 2 ini secara umum berisi tentang ringkasan tentang sistem jual beli yang dapat diakses oleh pengguna pada aplikasi PURRMART, fitur-fitur tambahan apa saja yang ditambahkan, penjelasan mengenai struktur data yang dipakai pada keseluruhan fungsi, penjelasan dan bukti *data test* dari setiap fitur, beserta dengan

lampiran-lampiran yang mendukung kebutuhan tugas ini, seperti : deskripsi tugas, notulen rapat, *log activity* anggota kelompok, dan notulen asistensi.

Tugas besar ini secara umum menghasilkan suatu aplikasi sederhana berbasis *Command Line Interface (CLI)*, Purrmart yang memuat beberapa fitur yang tidak sederhana. Aplikasi ini hanya berjalan di terminal sebagai output dengan *user*. Purrmart memuat fitur seperti *store* yang dapat melihat, menambah, menghapus, dan memverifikasi permintaan untuk menambah. Selain itu ada *work* yang dapat digunakan untuk menambah uang *user* dengan berbagai cara menarik. *User* juga mampu menyimpan dan melanjutkan permainan dengan kombinasi fitur *load*, *login*, dan *save*. Selain fitur-fitur yang disebutkan terdapat fitur penopang seperti *start*, *logout*, *exit*, *help*, register. Pada milestone 2 ini terdapat 2 fitur utama yaitu Wishlist dimana user bisa menambah, menghapus, menukar, melihat, dan menghapus keseluruhan isi Wishlistnya dan juga Cart, user dapat menambah, menambah, melihat serta membayar keranjang barangnya yang nantinya akan disimpan ke dalam history. Lebih rincinya akan dijelaskan pada [bagian 6](#).

## 2 Penjelasan Tambahan Spesifikasi Tugas

### 2.1 Perincian spesifikasi pada *TEBAK ANGKA*

Pada spesifikasi yang diberikan pada *TEBAK ANGKA*, hanya dijelaskan bahwa Jumlah kesempatan yang dipakai akan mempengaruhi uang yang didapatkan, menurut kami kalimat disini sedikit rancu, sehingga kami menentukan untuk memberikan hadiah sebesar 50\*sisa kesempatan yang user miliki, contohnya akan tercantum pada [data test \*TEBAK ANGKA\*](#).

## 3 Struktur Data (ADT)

Pada Bab ini akan dijelaskan mengenai jenis jenis ADT (*Abstract Data Type*) yang digunakan, alasan pemilihan ADT tersebut untuk mengatasi permasalahan pada tiap fungsi, berikut jenis ADT yang digunakan antara lain :

### 3.1 ADT Kustom

#### 3.1.1 ADT User

```
typedef struct {
    char name[MAX_LEN];
    char password[MAX_LEN];
    int money;
    Keranjang keranjang;
    History riwayat_pembelian;
    Wishlist wishlist;
} User;
```

ADT Kustom User ini berfungsi untuk mendefinisikan dan merepresentasikan data pengguna. ADT ini menyimpan nama *user* dalam *char name*, *password user* dalam *char password*, dan menyimpan uang *user* dalam *int money*, pada milestone 2 ini ditambahkan Keranjang dengan implementasi struktur data *SetMap*, History dengan struktur data *Stack*, dan Wishlist dengan *Linked List*, Struktur data User ini dibuat untuk mempermudah dan memenuhi spesifikasi yang ada nantinya saat user berbelanja ataupun menggunakan PURRMART.

#### 3.1.2 ADT Barang

```
typedef struct {
    char name[MAX_LEN];
    int price;
} Barang;
```

ADT Kustom ini memiliki fungsi yang serupa dengan ADT User. ADT ini berfungsi untuk mendefinisikan dan merepresentasikan data suatu barang. ADT ini menyimpan nama barang dalam *char name* dan harganya dalam *int price*.

### 3.2 ADT Stack

```
typedef struct {
    char item[MaxStack + 1][Max_Length];
    int price[MaxStack + 1];
    int quantity[MaxStack + 1];
    int cost[MaxStack + 1];
    int TOP;
    int total;
} History;
```

History yang menggunakan konsep ADT Stack ini digunakan untuk menyelesaikan persoalan mengenai riwayat pada fitur History ini akan menyimpan nama barang yang dimasukan ke riwayat pembelian adalah barang dengan total harga (harga barang \* kuantitas) terbesar. ADT Stack ini dipilih karena konsep FILO (First In Last Out) ini sangat sesuai dengan konsep riwayat pembelian yang akan disimpan setelah User membayar (Fitur Cart Payz0 dengan fitur history yang ada dalam aplikasi. ADT ini diimplementasikan sebagai ADT Stack dengan nama file header “stack.h”. ADT Stack digunakan pada fitur History dan Cart Pay untuk mengembalikan barang sesuai ketentuan diatas.

### 3.3 ADT Set Map

```
typedef struct
{
    Barang item[MaxBasket];
    int quantity[MaxBasket];
    int count;
} Keranjang;
```

ADT **Keranjang** di atas adalah struktur data yang digunakan untuk mengelola barang dalam fitur keranjang belanja (cart). Keranjang ini terdiri dari daftar barang (item) beserta jumlahnya (quantity) dengan kapasitas maksimum tertentu (MaxBasket), dan variabel count untuk menghitung jumlah total barang. Dalam implementasinya, ADT Keranjang menerapkan konsep **Set** untuk memastikan tidak ada barang yang duplikat di dalam keranjang, dan **Map** digunakan untuk memetakan setiap barang ke jumlahnya, sehingga memudahkan pencarian atau pembaruan data barang. ADT ini digunakan di seluruh fitur cart untuk menambahkan barang, memperbarui jumlah barang, atau menghapus barang dari keranjang secara efisien.

### 3.4 ADT Linked List

```
typedef Barang infoBarang;

typedef struct tElmtlist *address;

typedef struct tElmtlist {
    infoBarang info;
    address next;
} ElmtList;

typedef struct {
    address First;
} Wishlist;

typedef struct {
    char name[MAX_LEN];
    int price;
} Barang;
```

ADT Linked List ElmtList ini berisi info berupa infoBarang, infoBarang menyimpan nama barang dan harga untuk setiap barang dan next yang merupakan address ke elemen berikutnya, Linked List sangat baik digunakan karena pada PURRMART terdapat fitur WISHLIST yang setiap subfiturnya ada salah satunya WISHLIST SWAP yang sangat diuntungkan apabila menggunakan Linked List. Wishlist, ElmtList, dan infoBarang merupakan tiga struktur yang saling berkesinambungan. ListBerkait merupakan inisiasi awal agar dapat mengakses “node” pertama atau lagu pertama dari sebuah playlist. Pada struct ElmtList, terdapat infoBarang yang terdiri dari nama dan harga dari sebuah barang. ADT ini sangat perlu digunakan pada semua fitur wishlist. Dengan menggunakan konsep ini, wishlist akan dapat dimanipulasi dengan mudah. ADT List Berkait ini memiliki nama file header “linkedlist.h” dan digunakan pada semua fitur wishlist.



## 4 Program Utama

Program utama PURRMART akan menampilkan sebuah *welcome page* dan beberapa *command awal* yang dapat diakses *user* yaitu START, LOAD, dan QUIT. Command START adalah command yang bisa digunakan *user* pertama kali. Jika *user* pada *welcome page* memilih command START maka program akan membaca file default konfigurasi. Lalu, Command LOAD adalah command yang bisa digunakan *user* ketika pernah menyimpan sebuah progress aktivitas aplikasi PURRMART. Sedikit berbeda isi dari file default konfigurasi, tapi fungsi ini berguna untuk membaca file teks juga. Sedangkan command HELP digunakan oleh *user* untuk melihat apa saja fungsi yang tersedia pada aplikasi PURRMART. Command yang dapat dipanggil pada PURRMART setelah *user* memilih START atau LOAD sebagai berikut:

- a. LOGIN, Untuk masuk ke dalam akun dan memulai sesi.
- b. REGISTER, Untuk melakukan pendaftaran akun baru.
- c. QUIT, Untuk keluar dari program.
- d. MENU, Untuk kembali ke menu awal.

Kemudian setelah *user* login *user* bisa memilih beberapa Menu yang ada pada PURRMART, seperti :

- a. WORK, Untuk pekerjaan biasa.
- b. WORK CHALLENGE, Untuk pekerjaan spesial.
- c. STORE, Untuk berbelanja, dan mengatur barang yang mau dijual seperti menghapus, menambah, dan meminta penambahan barang.
- d. CART, Untuk melihat, menambahkan, mengeluarkan, dan membayar barang yang ada di shopping cart.
- e. WISHLIST, Untuk melihat wishlist, menambahkan barang ke wishlist, menukar posisi barang di wishlist, dan menghapus barang dari wishlist.
- f. PROFILE, Untuk melihat username, password, serta jumlah uang yang dimiliki.
- g. HISTORY, untuk menampilkan riwayat pembelian.
- h. SAVE, Untuk menyimpan state ke dalam file.
- i. LOGOUT, Untuk keluar dari sesi.

Berikut menu-menu yang bisa *user* akses pada STORE:

- a. STORE LIST, Untuk melihat barang-barang di toko.
- b. STORE REQUEST, Untuk meminta penambahan barang..
- c. STORE SUPPLY, Untuk menambahkan barang dari permintaan.
- d. STORE REMOVE, Untuk menghapus barang.
- e. MENU, Untuk kembali ke menu sebelumnya.

Berikut menu-menu yang bisa *user* akses pada WORK CHALLENGE:

- a. TEBAK ANGKA, Tebak kata dengan biaya bermain 200 dan kesempatan memenangkan hingga 500.
- b. WORDL3, Tebak kata dengan biaya bermain 500 dan kesempatan memenangkan 1500.
- c. MENU, Untuk kembali ke menu sebelumnya.

Berikut menu-menu yang bisa *user* akses pada CART:

- a. CART ADD, Untuk menambahkan barang ke shopping cart.
- b. CART REMOVE, Untuk mengeluarkan kuantitas barang dari shopping cart.
- c. CART SHOW, Untuk melihat barang-barang yang ada di shopping cart.
- d. CART PAY, Untuk membayar semua barang yang ada di shopping cart.
- e. MENU, Untuk kembali ke menu sebelumnya.

Berikut menu-menu yang bisa user akses pada WISHLIST:

- a. WISHLIST ADD, Untuk menambahkan barang ke wishlist.
- b. WISHLIST SWAP <i> <j>, Untuk menukar posisi barang.
- c. WISHLIST REMOVE <i>, Untuk menghapus barang ke-i.
- d. WISHLIST REMOVE, Untuk menghapus barang berdasarkan nama.
- e. WISHLIST CLEAR, Untuk menghapus seluruh barang.
- f. WISHLIST SHOW, Untuk menunjukkan barang di wishlist.
- g. MENU, Untuk kembali ke menu sebelumnya.

Beberapa command yang dapat dipanggil pada welcome page, command START dan LOAD, tidak bisa digunakan kembali jika sudah memanggil salah satu diantara kedua command tersebut. Selain dari kedua command sebelumnya dapat digunakan lebih dari satu kali. Setelah user menggunakan berbagai command yang tertera pada PURRMART, user dapat memilih untuk mengakhiri aktivitas pada aplikasi PURRMART. Command yang bisa dipilih untuk menutup program PURRMART adalah SAVE dan QUIT. Command SAVE digunakan untuk menyimpan perubahan yang sudah dilakukan pada aplikasi PURRMART. Command ini akan menyimpan state aplikasi terbaru ke dalam suatu file. Sedangkan command QUIT digunakan untuk keluar dari aplikasi PURRMART tanpa menyimpan perubahan yang telah dilakukan. Dalam menjalankan beberapa command diatas, dapat terjadi kasus dimana pengguna memasukan inputan yang salah, memilih command yang tidak tertera pada program, dll. Hal tersebut dapat membuat program mengeluarkan pesan berupa “Invalid command. Please try again.” yang mengartikan bahwa terdapat sesuatu ketidaktepatan input yang user lakukan.

## 5 Algoritma-Algoritma Menarik

### 5.1 Makefile

Menurut kami algoritma Makefile menarik karena memiliki sintaksis-sintaksis tersendiri. Dalam pengimplementasiannya juga melewati banyak kesulitan hingga menemukan solusi untuk masalah ini. Alasan kenapa menggunakan karena algoritma ini sangat mempermudah kita dalam meng-*compile* suatu file atau program kita, dari yang seharusnya panjang dengan gcc dan semua file yang diminta menjadi hanya 'make <function>'. Makefile ini sepertinya memiliki banyak cara yang berbeda untuk setiap *operating system* (OS), seperti pada windows atau linux itu berbeda dan memiliki sintaksnya sendiri-sendiri. Keberadaan make file ini juga dapat diterapkan pada linux dan windows sehingga mempermudah kami. Mempelajari cara bekerja Makefile ini sangat membantu dalam pembuatan program keseluruhan

### 5.2 Handler

Pada program kami, kami membuat file handler yang berisi fungsi-fungsi untuk mengoperasikan setiap menu. Kami membuat algoritma ini untuk mempermudah pemeliharaan program. Keberadaan program ini yang terpisah dari bagian *Main* membantu kami lebih mudah meneliti letak kesalahan atau keberadaan *bug*. Selain itu, keberadaan algoritma ini membantu dalam proses penyusunan. Mempelajari bagaimana algoritma ini bekerja dan beroperasi sangat membantu dalam pembuatan program keseluruhan. Meskipun pada prosesnya sempat membingungkan karena file ini tidak berada di folder yang sama dengan main, akhirnya kami bisa mengatasinya.

### 5.3 Animasi

Pada program aplikasi *purrmart* ini, kami menambahkan algoritma animasi. Algoritma ini berfungsi untuk mengoperasikan animasi-animasi dari *ascii* yang kita miliki. Kami membuat algoritma ini dengan tujuan mempermudah dan mempersingkat untuk pemanggilan fungsi di *handler* sehingga program akan lebih rapi. Fungsi-fungsi yang berada di algoritma *animasi* akan dipanggil ke *handler* dan beroperasi di *handler*. Namun, setiap *ascii* yang ada di kelola dengan baik di algoritma *animasi*. Keberadaan algoritma ini menjadikan program ini lebih baik dan mudah untuk dipelihara dan disusun. Mempelajari pembuatan algoritma ini membuat kami sebagai penyusun lebih paham tentang proses *ascii* dan lebih paham tentang bahasa C itu sendiri.

## 6 Data Test

Pada bagian Data test terdiri atas 2 bagian,yaitu hasil test untuk setiap fitur dan perubahan fitur yang ada pada milestone 1, masing-masing akan dilampirkan hasil yang diharapkan dan hasil yang diperoleh dan penjelasan apabila hasilnya tidak sesuai.

### 6.1 PROFILE

PROFILE adalah *command* yang digunakan untuk melihat data diri pengguna. PROFILE hanya dapat dipanggil saat status pengguna telah login

Fitur yang di tes: START

Hasil yang diharapkan:

```
>> PROFILE
Nama : Purry
Saldo: 2000

(Silahkan kreasikan atribut yang ditampilkan)
// Kembali ke menu utama
```

Hasil yang didapatkan:



Gambar 6.1.1 Test case ketika profile berhasil dijalankan

### 6.2 CART ADD <nama> <n>

CART ADD adalah *command* yang digunakan untuk menambahkan barang dengan kuantitas tertentu ke dalam keranjang belanja.

Fitur yang di tes: CART ADD

Hasil yang diharapkan:

Jika barang tidak ada di toko maka akan menampilkan bahwa barang tidak ada di toko

```
>> CART ADD AK47 20
Berhasil menambahkan 20 AK47 ke keranjang belanja!
// Kembali ke menu utama
```

```
>> CART ADD BebekKaliya 240
Barang tidak ada di toko!
// Perintah invalid; Kembali ke menu utama
```

Hasil yang didapatkan :

```
>> STORE LIST
+-----+
| No | Nama                      | Harga |
+-----+
| 1  | AK47                      | 10    |
| 2  | Lalabu                    | 20    |
| 3  | Ayam Goreng Crisbar      | 20    |
| 4  | Meong                     | 500   |
+-----+

Enter command: cart add Meong 1
Berhasil menambahkan 1 Meong ke keranjang belanja!
```

Gambar 6.2.1 Test case jika berhasil menambahkan barang ke keranjang

```
>> STORE LIST
+-----+
| No | Nama                      | Harga |
+-----+
| 1  | AK47                      | 10    |
| 2  | Lalabu                    | 20    |
| 3  | Ayam Goreng Crisbar      | 20    |
| 4  | Meong                     | 500   |
+-----+

Enter command: cart add ak48 1
Barang tidak ada di toko!
```

Gambar 6.2.2 Test case jika tidak ada barang yang ingin ditambah ke keranjang

### 6.3 CART REMOVE <nama> <n>

CART REMOVE adalah *command* yang digunakan untuk mengurangi barang sejumlah kuantitas tertentu dari keranjang belanja. Perlu dilakukan validasi terhadap kuantitas yang

diberikan, bila kuantitas pada keranjang belanja lebih sedikit dari N maka perintah akan gagal.

Fitur yang di tes: CART REMOVE

Hasil yang diharapkan:

>> <b>CART REMOVE AK47 10</b> Berhasil mengurangi 10 AK47 dari keranjang belanja! <i>// Kembali ke menu utama</i>
>> <b>CART REMOVE AK47 70</b> Tidak berhasil mengurangi, hanya terdapat 10 AK47 pada keranjang! <i>// Asumsi di keranjang belanja jumlah AK47 &lt;70 sehingga perintah invalid; Kembali ke menu utama</i>
>> <b>CART REMOVE BintangSkibidi 70</b> Barang tidak ada di keranjang belanja! <i>// Asumsi tidak ada Bintang Skibidi di keranjang belanja; Kembali ke menu utama</i>

Hasil yang didapatkan:

```
>> STORE LIST
+-----+
| No | Nama                | Harga |
+-----+
| 1  | AK47                | 10    |
| 2  | Lalabu              | 20    |
| 3  | Ayam Goreng Crisbar | 20    |
| 4  | Meong               | 500   |
+-----+

Enter command: cart remove Meong 1
Berhasil mengurangi 1 Meong dari keranjang belanja!
```

Gambar 6.3.1 Test case jika berhasil mengurangi barang dari keranjang

```
>> STORE LIST
+-----+
| No | Nama                | Harga |
+-----+
| 1  | AK47                | 10    |
| 2  | Lalabu              | 20    |
| 3  | Ayam Goreng Crisbar | 20    |
| 4  | Meong               | 500   |
+-----+

Enter command: cart remove Meong 20
Tidak berhasil mengurangi, hanya terdapat 10 Meong pada keranjang!
```

Gambar 6.3.2 Test case jika jumlah yang ingin dikurangi lebih dari barang yang ada di keranjang

```
>> STORE LIST
+-----+
| No | Nama                | Harga |
+-----+
| 1  | AK47                | 10    |
| 2  | Lalabu              | 20    |
| 3  | Ayam Goreng Crisbar | 20    |
| 4  | Meong               | 500   |
+-----+

Enter command: cart remove Meong 1
Barang tidak ada di keranjang belanja!
```

Gambar 6.3.3 Test case jika tidak ada barang yang valid di keranjang

## 6.4 CART SHOW

CART SHOW adalah *command* yang digunakan untuk menunjukkan barang-barang yang sudah dimasukkan ke dalam keranjang.

Fitur yang di tes: CART SHOW

Hasil yang diharapkan:

```
// Contoh dimana keranjang memiliki isi
>> CART SHOW
Berikut adalah isi keranjangmu.
Kuantitas Nama  Total
2      AK47  20
1      Lalabu 10
Total biaya yang harus dikeluarkan adalah 30.

// Command mati; Kembali ke menu utama

// Contoh yang kosong
>> CART SHOW
Keranjang kamu kosong!
```

Hasil yang didapatkan :

```
Enter command: cart show
>> CART
+-----+
| Kuantitas | Nama          | Total |
+-----+
| 1         | Meong         | 500   |
+-----+
Total biaya yang harus dikeluarkan adalah 500.

Ketik 'back' untuk keluar.
```

Gambar 6.4.1 Test case jika ada barang di dalam keranjang

```
Enter command: cart show
Keranjang kamu kosong!

Ketik 'back' untuk keluar.
```

Gambar 6.4.2 Test case jika keranjang kosong

## 6.5 CART PAY

CART PAY adalah *command* yang digunakan untuk membeli barang-barang yang sudah dimasukan ke dalam keranjang. Perlu dipastikan bahwa **pengguna memiliki uang yang cukup** untuk membeli seluruh barang keranjang. Pembelian akan mengurangi uang yang dimiliki pengguna dan menambahkan riwayat pembelian.

Nama barang yang dimasukan ke riwayat pembelian adalah barang dengan total harga (harga barang \* kuantitas) terbesar. Jika terdapat lebih dari 1 barang dengan total yang sama, maka yang disimpan adalah barang dengan urutan lexical yang lebih besar. Dimasukan juga total harga pada pembelian tersebut.

Fitur yang di tes: CART PAY

Hasil yang diharapkan:

<pre>// Contoh pembayaran yang berhasil (Pengguna memasukan Ya) &gt;&gt; <b>CART PAY</b> Kamu akan membeli barang-barang berikut. Kuantitas Nama Total 2 AK47 20 1 Lalabu 10 Total biaya yang harus dikeluarkan adalah 30, apakah jadi dibeli? (Ya/Tidak): <b>Ya</b>  Selamat kamu telah membeli barang-barang tersebut!  // Command mati; Kembali ke main menu</pre>
<pre>// Contoh pembayaran yang gagal (Pengguna tidak memiliki uang yang cukup) &gt;&gt; <b>CART PAY</b> Kamu akan membeli barang-barang berikut. Kuantitas Nama Total 2 AK47 20 1 Lalabu 20 Total biaya yang harus dikeluarkan adalah 40, apakah jadi dibeli? (Ya/Tidak): <b>Ya</b>  Uang kamu hanya 15, tidak cukup untuk membeli keranjang!  // Command mati; Kembali ke main menu</pre>
<pre>// Contoh pembayaran yang gagal (Pengguna memasukan Tidak) &gt;&gt; <b>CART PAY</b> Kamu akan membeli barang-barang berikut. Kuantitas Nama Total 2 AK47 20</pre>



<p>1      Lalabu   10</p> <p>Total biaya yang harus dikeluarkan adalah 30, apakah jadi dibeli? (Ya/Tidak): <b>Tidak</b></p> <p>// Command mati; Kembali ke main menu</p>
<p>// Contoh pembayaran yang gagal (Pengguna memasukan masukan aneh)</p> <p>&gt;&gt; <b>CART PAY</b></p> <p>Kamu akan membeli barang-barang berikut.</p> <p>Kuantitas   Nama   Total</p> <p>2      AK47   20</p> <p>1      Lalabu   10</p> <p>Total biaya yang harus dikeluarkan adalah 30, apakah jadi dibeli? (Ya/Tidak): <b>Purry</b></p> <p>// Command mati; Kembali ke main menu</p>
<p>// Contoh pembayaran yang gagal (Keranjang kosong)</p> <p>&gt;&gt; <b>CART PAY</b></p> <p>Keranjang kamu kosong!</p>

Hasil yang didapatkan :

```

Enter command: cart pay
>> CART
+-----+
| Kuantitas | Nama          | Total  |
+-----+
| 1         | AK47          | 10     |
+-----+
Total biaya yang harus dikeluarkan adalah 10.
Apakah jadi dibeli? (Ya/Tidak): ya
Selamat kamu telah membeli barang-barang tersebut!, uang kamu tersisa 90.

```

Gambar 6.5.1 Test case jika pembayaran berhasil dilakukan

```

Enter command: cart pay
>> CART
+-----+
| Kuantitas | Nama          | Total  |
+-----+
| 1         | Meong         | 500    |
+-----+
Total biaya yang harus dikeluarkan adalah 500.
Apakah jadi dibeli? (Ya/Tidak): ya
Uang kamu hanya 90, tidak cukup untuk membeli barang yang ada di keranjang!

```

Gambar 6.5.2 Test case jika uang pengguna tidak mencukupi untuk melakukan pembayaran

```
Enter command: cart pay
>> CART
+-----+
| Kuantitas | Nama          | Total  |
+-----+
| 10        | Meong         | 5000   |
+-----+
Total biaya yang harus dikeluarkan adalah 5000.
Apakah jadi dibeli? (Ya/Tidak): tidak
Pembayaran dibatalkan.
```

Gambar 6.5.3 Test case jika pengguna membatalkan pembayaran

```
Enter command: cart pay
>> CART
+-----+
| Kuantitas | Nama          | Total  |
+-----+
| 10        | Meong         | 5000   |
+-----+
Total biaya yang harus dikeluarkan adalah 5000.
Apakah jadi dibeli? (Ya/Tidak): purry
Input tidak valid. Pembayaran dibatalkan.
```

Gambar 6.5.4 Test case jika pengguna memasukkan masukan aneh

```
Enter command: cart pay
Keranjang kamu kosong!

Ketik 'back' untuk keluar.
```

Gambar 6.5.5 Test case jika keranjang pengguna masih kosong

## 6.6 HISTORY <n>

HISTORY adalah *command* yang digunakan untuk menunjukan riwayat pembelian seorang pengguna. N merupakan jumlah riwayat yang ditampilkan, contoh N=3 maka akan menampilkan 3 riwayat pembelian terbaru. Jika N melebihi jumlah riwayat pembelian yang ada, maka seluruh riwayat pembelian akan ditampilkan. Urutan penunjukan adalah dari yang paling baru ke paling tua.

Fitur yang di tes: HISTORY

Hasil yang diharapkan:

```
// Contoh menunjukan riwayat pembelian N < total riwayat
>> HISTORY 3
Riwayat pembelian barang:
1. AK47 40
2. AK47 100
3. Lalabu 35

// Command mati; Kembali ke main menu
```

// Contoh menunjukan riwayat pembelian N >= total riwayat

>> **HISTORY 10**

Riwayat pembelian barang:

1. AK47 40
2. AK47 100
3. Lalabu 35
4. AK47 10
5. Meong 500
6. Ayam Goreng Crisbar 20

// Command mati; Kembali ke main menu

// Contoh riwayat pembelian kosong

Kamu belum membeli barang apapun!

Hasil yang didapatkan :

Enter command: history 3

Riwayat pembelian barang:

+-----+		
No	Nama Barang	Total Harga
+-----+		
1	AK47	10
2	Ayam Goreng Crisbar	20
3	Meong	500
+-----+		

Gambar 6.6.1 Test case jika riwayat pembelian N < total riwayat

Enter command: history 10

Riwayat pembelian barang:

+-----+		
No	Nama Barang	Total Harga
+-----+		
1	AK47	10
2	Ayam Goreng Crisbar	20
3	Meong	500
4	AK47	10
5	Lalabu	35
6	AK47	100
7	AK47	40
+-----+		

Gambar 6.6.2 Test case jika riwayat pembelian N >= total riwayat

Kamu belum membeli barang apapun!

Gambar 6.6.3 Test case jika riwayat pembelian kosong

## 6.7 WISHLIST ADD

WISHLIST ADD merupakan *command* yang digunakan untuk menambahkan suatu barang ke *wishlist*.

Fitur yang di tes: WISHLIST ADD

Hasil yang diharapkan:

>> <b>WISHLIST ADD</b> Masukkan nama barang: <b>Ayam Geprek Bakar Crispy Besthal</b>  Berhasil menambahkan Ayam Geprek Bakar Crispy Besthal ke wishlist!
>> <b>WISHLIST ADD</b> Masukkan nama barang: <b>Ayam Geprek Bakar Crispy Besthal</b>  Ayam Geprek Bakar Crispy Besthal sudah ada di wishlist!
>> <b>WISHLIST ADD</b> Masukkan nama barang: <b>Ayam Geprek Sambalado Besthal</b>  Tidak ada barang dengan nama Ayam Geprek Sambalado Besthal!

Hasil yang didapatkan :

```
Enter command: wishlist add
Masukkan nama barang: Lalabu
Berhasil menambahkan Lalabu ke wishlist!
```

### 6.7.1 Test case ketika berhasil menambahkan barang ke wishlist

```
Enter command: wishlist add
Masukkan nama barang: Lalabu
Lalabu sudah ada di wishlist!
```

### 6.7.2 Test case ketika mencoba menambahkan barang yang sudah ada di wishlist

```
Enter command: wishlist add
Masukkan nama barang: ayam crispy bakar besthal
Tidak ada barang dengan nama ayam crispy bakar besthal!
```

### 6.7.3 Test case ketika barang tidak ada di list barang

## 6.8 WISHLIST SWAP <i><j>

WISHLIST SWAP merupakan *command* yang digunakan untuk menukar barang posisi ke-i dengan barang posisi ke-j pada *wishlist*. Posisi i dan j merupakan urutan barang pada *wishlist*, urutan dimulai dari 1.

Fitur yang di tes: WISHLIST SWAP

Hasil yang diharapkan:

```
>> WISHLIST SWAP 1 2
```

Berhasil menukar posisi Ayam Geprek Bakar Crispy Besthal dengan Ayam Mangut Besthal pada wishlist!

// Urutan Ayam Geprek Bakar Crispy Besthal berubah dari 1 menjadi 2. Sebaliknya, urutan Ayam Mangut Besthal berubah dari 2 menjadi 1

```
>> WISHLIST SWAP 1 2
```

Gagal menukar posisi Ayam Geprek Bakar Crispy Besthal!

// Hanya terdapat satu barang (Ayam Geprek Bakar Crispy Besthal) pada *wishlist* sehingga posisinya tidak dapat ditukar

Hasil yang didapatkan :

```
Enter command: wishlist swap 1 3
```

```
Berhasil menukar posisi Lalabu dengan Ayam Goreng Crisbar pada wishlist!
```

6.8.1 Test case ketika berhasil menukar posisi barang pada wishlist

```
Enter command: wishlist swap 1 5
```

```
Indeks tidak valid! Wishlist hanya memiliki 3 barang.
```

6.8.2 Test case ketika gagal menukar posisi barang pada wishlist

## 6.9 WISHLIST REMOVE <i>

WISHLIST REMOVE adalah *command* yang digunakan untuk menghapus barang dengan posisi ke-*i* dari *wishlist*.

Fitur yang di tes: WISHLIST REMOVE <*i*>

Hasil yang diharapkan:

```
// Contoh menghapus barang ke-i dari WISHLIST
```

```
>> WISHLIST REMOVE 2
```

Berhasil menghapus barang posisi ke-2 dari wishlist!

```
// Command mati; Kembali ke main menu
```

```
// Contoh penghapusan wishlist yang gagal (Pengguna memasukkan urutan barang yang tidak ada)
```

```
//Misalnya dalam kasus ini, hanya terdapat 5 barang di dalam wishlist
```

```
>> WISHLIST REMOVE 10
```

Penghapusan barang WISHLIST gagal dilakukan, Barang ke-10 tidak ada di WISHLIST!

```
// Command mati; Kembali ke main menu
```

```
// Contoh penghapusan wishlist yang gagal (Wishlist kosong)
```

```
//Misalnya dalam kasus ini, tidak ada barang di dalam wishlist
```

```
>> WISHLIST REMOVE 1
```

Penghapusan barang WISHLIST gagal dilakukan, WISHLIST kosong!

// Command mati; Kembali ke main menu

// Contoh penghapusan wishlist yang gagal (Pengguna memasukkan perintah yang tidak valid)

>> **WISHLIST REMOVE XY**

Penghapusan barang WISHLIST gagal dilakukan, command tidak valid!

// Command mati; Kembali ke main menu

Hasil yang didapatkan :

```
Enter command: wishlist remove 1
```

```
Berhasil menghapus barang posisi ke-1 dari wishlist!
```

6.9.1 Test case ketika berhasil menghapus barang dari wishlist menggunakan indeks

```
Enter command: wishlist remove 10
```

```
Penghapusan barang WISHLIST gagal dilakukan, Barang ke-10 tidak ada di WISHLIST!
```

6.9.2 Test case ketika gagal menghapus barang dari wishlist karena memasukkan urutan barang yang tidak ada

```
Enter command: wishlist remove 1
```

```
Penghapusan barang WISHLIST gagal dilakukan, WISHLIST kosong!
```

6.9.3 Test case ketika gagal menghapus barang karena wishlist kosong

```
Enter command: wishlist remove y
```

```
Invalid index provided.
```

6.9.4 Test case ketika gagal menghapus barang karena input tidak valid

## 6.10 **WISHLIST REMOVE**

WISHLIST REMOVE adalah *command* yang digunakan untuk menghapus barang dari wishlist berdasarkan nama barang yang dimasukkan pengguna.

Fitur yang di tes: WISHLIST REMOVE

Hasil yang diharapkan:

// Contoh menghapus barang “Lalabu” dari WISHLIST

>> **WISHLIST REMOVE**

Masukkan nama barang yang akan dihapus : **Lalabu**

Lalabu berhasil dihapus dari WISHLIST!

// Command mati; Kembali ke main menu

// Contoh penghapusan wishlist yang gagal (Barang tidak ada di WISHLIST)

>> **WISHLIST REMOVE**

Masukkan nama barang yang akan dihapus : **LoremIpsum**  
Penghapusan barang WISHLIST gagal dilakukan, LoremIpsum tidak ada di WISHLIST!

// Command mati; Kembali ke main menu

Hasil yang didapatkan :

```
Enter command: wishlist remove
Masukkan nama barang yang akan dihapus: AK47
Berhasil menghapus barang AK47 dari WISHLIST!
```

#### 6.10.1 Test case ketika barang berhasil dihapus dari wishlist

```
Enter command: wishlist remove
Masukkan nama barang yang akan dihapus: purry
Penghapusan barang WISHLIST gagal dilakukan, purry tidak ada di WISHLIST!
```

#### 6.10.2 Test case ketika barang gagal dihapus dari wishlist

### 6.11 WISHLIST CLEAR

WISHLIST CLEAR adalah *command* yang digunakan untuk menghapus semua barang yang terdapat di dalam WISHLIST.

Fitur yang di tes: WISHLIST CLEAR

Hasil yang diharapkan:

```
>> WISHLIST CLEAR
Wishlist user berhasil dihapus.
```

Hasil yang didapatkan :

```
Enter command: wishlist clear
Wishlist user1 berhasil dihapus
```

#### 6.11.1 Test case ketika wishlist berhasil dikosongkan

### 6.12 WISHLIST SHOW

WISHLIST SHOW adalah *command* yang digunakan untuk menunjukkan barang-barang yang sudah dimasukkan ke dalam wishlist.

Fitur yang di tes: WISHLIST SHOW

Hasil yang diharapkan:

```
>>> WISHLIST SHOW
Berikut adalah isi wishlist-mu:
1 Ayam Geprek Bakar Crispy Besthal
2 Ayam Mangut Besthal
3 Karaage Don
4 Torikatsu Don
```

```
>> WISHLIST SHOW
Wishlist kamu kosong!
```

Hasil yang didapatkan:

```
Enter command: wishlist show
>> WISHLIST
+-----+
| No | Nama                |
+-----+
| 1  | Ayam Goreng Crisbar |
| 2  | AK47                 |
+-----+

Ketik 'back' untuk keluar.
```

6.12.1 Test case ketika berhasil menampilkan seluruh isi wishlist

```
Enter command: wishlist show
Wishlist kamu kosong!

Ketik 'back' untuk keluar.
```

6.12.2 Test case ketika menampilkan isi wishlist kosong

### 6.13 Update *STORE LIST*

STORE LIST akan menampilkan nama barang yang dijual **beserta harganya**.

Hasil yang diharapkan :

```
>> STORE LIST
List barang yang ada di toko :
- Platypus Laser - Harga: 100
- Shrink Ray - Harga: 500
- Net Shooter - Harga: 250
- Camouflage Cloak - Harga: 150
- Sleep Dart Gun - Harga: 300
- Bubble Blaster - Harga: 200
```

```
>> STORE LIST
TOKO KOSONG
```

Hasil yang didapatkan :



```
Enter command: store list
```

```
>> STORE LIST
```

+-----+			
No	Nama	Harga	
+-----+			
1	AK47	10	
2	Lalabu	20	
3	Ayam Goreng Crisbar	20	
4	Meong	500	
+-----+			

```
Ketik 'back' untuk keluar.
```

6.13.1 Test case ketika berhasil menampilkan seluruh isi store list beserta harganya

```
Enter command: store list
```

```
>> STORE LIST
```

```
TOKO KOSONG
```

```
Ketik 'back' untuk keluar.
```

6.13.2 Test case ketika menampilkan isi store list kosong

## 7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Fitur Start	Memeriksa apakah fitur start dapat membaca file.txt	Mengetikkan command START pada terminal	<a href="#">START</a>	Fitur dapat membaca config.txt konfigurasi dan berhasil mengeluarkan output yang diharapkan	Sesuai yang diharapkan
2	Fitur Load	Memeriksa apakah fitur load dapat membaca file txt lain	Mengetikkan command LOAD dilanjut dengan nama file.txt pada terminal	<a href="#">LOAD</a>	Fitur dapat membaca file txt eksternal dan berhasil mengeluarkan output yang diharapkan	Sesuai yang diharapkan
3	Fitur Save	Memeriksa apakah fitur save dapat menyimpan state ke dalam suatu file	Mengetikkan command SAVE diikuti dengan nama file	<a href="#">SAVE</a>	Fitur dapat menyimpan state ke dalam suatu file tertentu dan mengeluarkan output yang diharapkan	Sesuai yang diharapkan
4	Fitur Profile	Memeriksa apakah fitur Profile dapat menampilkan data diri status pengguna setelah login	Mengetikkan command PROFILE	PROFILE	Fitur dapat menampilkan data dari pengguna seperti nama dan saldo yang dimiliki	Sesuai yang diharapkan
5	Fitur Cart Add	Memeriksa apakah fitur Cart Add dapat menambahkan barang dengan kuantitas tertentu ke dalam keranjang belanja	Mengetikkan command CART ADD diikuti dengan nama dan jumlah barang yang ingin ditambahkan ke keranjang	CART ADD <nama> <n>	Fitur dapat menambahkan barang sesuai dengan jumlah yang dimasukkan pengguna ke keranjang	Sesuai yang diharapkan
6	Fitur Cart Remove	Memeriksa apakah fitur Cart Remove dapat mengurangi barang dengan kuantitas tertentu dari keranjang belanja	Mengetikkan command CART REMOVE diikuti dengan jumlah barang yang ingin dikurangi dari keranjang	<a href="#">CART REMOVE</a> <nama> <n>	Fitur dapat mengurangi isi dari keranjang seorang pengguna	Sesuai yang diharapkan
7	Fitur Cart Show	Memeriksa apakah fitur Cart Show dapat menunjukkan barang yang sudah dimasukkan ke dalam keranjang	Mengetikkan command CART SHOW	<a href="#">CART SHOW</a>	Fitur dapat menampilkan isi seluruh dari keranjang	Sesuai yang diharapkan
8	Fitur Cart Pay	Memeriksa apakah fitur Cart Pay dapat membeli barang	Mengetikkan command CART PAY setelah itu	<a href="#">CART PAY</a>	Fitur dapat membeli barang sesuai di keranjang	Sesuai yang diharapkan

		yang sudah dimasukkan ke dalam keranjang	mengonfirmasi dengan mengetikkan Ya/Tidak sesuai dengan kondisi barang jadi ingin dibeli atau tidak		sesuai dengan jumlah dan harganya, nanti uang pengguna tersebut juga akan berkurang	
9	Fitur History	Memeriksa apakah fitur History dapat menunjukkan riwayat pembelian seorang pengguna	Mengetikkan command HISTORY lalu diikuti dengan jumlah history pembelian yang sudah pernah dilakukan	<a href="#">HISTORY</a> <a href="#">&lt;n&gt;</a>	Fitur dapat menampilkan riwayat pembelian barang sesuai urutannya dan sesuai dengan input jumlah dari pengguna	Sesuai yang diharapkan
10	Fitur Wishlist Add	Memeriksa apakah fitur Wishlist Add dapat digunakan untuk menambahkan suatu barang ke wishlist	Mengetikkan command WISHLIST ADD lalu diikuti dengan nama barang yang ingin dimasukkan ke dalam wishlist	<a href="#">WISHLIS</a> <a href="#">T ADD</a>	Fitur dapat mendaftarkan barang ke dalam wishlist	Sesuai yang diharapkan
11	Fitur Wishlist Swap	Memeriksa apakah fitur Wishlist Swap dapat menukar barang pada wishlist	Mengetikkan command WISHLIST SWAP lalu diikuti dengan index kedua barang dalam wishlist yang ingin ditukar posisinya	<a href="#">WISHLIS</a> <a href="#">T SWAP</a> <a href="#">&lt;i&gt; &lt;j&gt;</a>	Fitur dapat menukar posisi dari barang pada wishlist	Sesuai yang diharapkan
12	Fitur Wishlist Remove <i>	Memeriksa apakah fitur Wishlist Remove <i> dapat menghapus barang dengan posisi ke-i dari wishlist	Mengetikkan command WISHLIST REMOVE lalu diikuti dengan index barang yang mau dihapus	<a href="#">WISHLIS</a> <a href="#">T REMOVE</a> <a href="#">&lt;i&gt;</a>	Fitur dapat menghapus barang sesuai dengan index yang dimasukkan oleh pengguna	Sesuai yang diharapkan
13	Fitur Wishlist Remove	Memeriksa apakah fitur Wishlist Remove dapat menghapus barang dari wishlist berdasarkan nama barang yang di input pengguna	Mengetikkan command WISHLIST REMOVE lalu diikuti dengan nama barang yang ingin dihapus dari wishlist	<a href="#">WISHLIS</a> <a href="#">T REMOVE</a>	Fitur dapat menghapus nama barang yang diminta oleh pengguna	Sesuai yang diharapkan
14	Fitur Wishlist Clear	Memeriksa apakah fitur Wishlist Clear dapat menghapus semua barang di dalam wishlist	Mengetikkan command WISHLIST CLEAR	<a href="#">WISHLIS</a> <a href="#">T CLEAR</a>	Fitur dapat menghilangkan seluruh isi dari wishlist seorang pengguna	Sesuai yang diharapkan
15	Fitur Wishlist Show	Memeriksa apakah fitur Wishlist Show dapat menunjukkan	Mengetikkan command WISHLIST SHOW	<a href="#">WISHLIS</a> <a href="#">T SHOW</a>	Fitur dapat menunjukkan barang yang sudah	Sesuai yang diharapkan

		barang yang sudah dimasukkan ke dalam wishlist			pernah dimasukkan ke dalam wishlist	
16	Store List	Memeriksa apakah fitur store list dapat menampilkan daftar barang	Mengetikkan command STORE LIST pada terminal	<a href="#">Update STORE LIST</a>	Fitur dapat mengeluarkan menampilkan list barang dengan output yang diharapkan	Sesuai yang diharapkan

## 8 Pembagian Kerja dalam Kelompok

Fitur	NIM Coder	NIM Tester	Catatan
PROFILE	18223125	18223136 18222133	
CART ADD	18223110	18223136 18222133	
CART REMOVE	18223110	18223136 18222133	
CART SHOW	18223136	18223136 18222133	
CART PAY	18223136	18223136 18222133	
HISTORY	18223110	18223136 18222133	
WISHLIST ADD	18222133	18222133	
WISHLIST SHOW	18222133	18222133	
WISHLIST SWAP	18223109	18222133	
WISHLIST REMOVE	18223109	18222133	
WISHLIST REMOVE <i>	18223109	18222133	
WISHLIST CLEAR	18223125	18222133	
ADT CUSTOM	18223110	18223110	
ADT STACK	18223110	18223110	
ADT SETMAP	18223110	18223110	
ADT LINKED LIST	18223110	18223110	
LOAD	18223125	18223125	
START	18223125	18223125	

SAVE	18223125	18223125	
STORE LIST	18223110	18223110	

## 9 Lampiran

### 9.1 Deskripsi Tugas Besar

Buatlah sebuah aplikasi simulasi berbasis CLI (command-line interface). Sistem ini dibuat dalam bahasa C dengan menggunakan struktur data yang sudah kalian pelajari di mata kuliah ini. Kalian boleh menggunakan (atau memodifikasi) struktur data yang sudah kalian buat untuk praktikum pada tugas besar ini. Daftar ADT yang wajib digunakan dapat dilihat pada bagian Daftar ADT. Library yang boleh digunakan hanya `stdio.h`, `stdlib.h`, `time.h`, dan `math.h`.

Tentang sistem, PURRMART adalah sebuah aplikasi yang dapat mensimulasikan aktivitas beli barang pada e-commerce. PURRMART memiliki beberapa fitur utama, yaitu:

- Menampilkan barang toko
- Meminta dan menyuplai barang baru ke toko
- Menyimpan dan membeli barang dalam keranjang
- Menampilkan barang yang sudah dibeli
- Membuat dan menghapus wishlist
- Bekerja untuk menghasilkan uang

Ketika program pertama kali dijalankan, PURRMART akan memperlihatkan main menu yang berisi welcome menu dan beberapa command yaitu START, LOAD, dan juga HELP. Setelah itu, program akan memasuki login menu yang memiliki command LOGIN, REGISTER, dan juga HELP. Jika pengguna berhasil memasuki kredensial suatu akun, maka mereka akan masuk ke menu selanjutnya. Main menu menerima masukan berupa command seperti WORK, WORK CHALLENGE, STORE LIST, STORE REQUEST, STORE SUPPLY, STORE REMOVE, PROFILE, CART ADD, CART REMOVE, CART SHOW, CART PAY, HISTORY, WISHLIST ADD, WISHLIST SWAP, WISHLIST REMOVE, WISHLIST CLEAR, WISHLIST SHOW, LOGOUT, dan SAVE. Program akan terus menerima command sampai diberikan command QUIT yang berlaku pada seluruh menu.

### 9.2 Notulen Rapat

#### 9.2.1 Rapat I

Tanggal : 1 December 2024

Waktu : 15.00 - 16.00

Pembahasan : Pembagian tugas, penyesuaian persepsi terkait spesifikasi yang ada


### 9.3 Log Activity Anggota Kelompok

NIM	Nama	Keterangan
18223109	Fadhil Rifqi Rabbani Pane	16/12/2024 - 17/12/2024: Mengerjakan Wishlist Remove dan Wishlist Remove <i> 17/12/2024 - 19/12/2024: Mengerjakan Wishlist Swap <i> <j> 19/12/2024 : Mengerjakan laporan

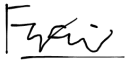




18223110	Desati Dinda Saraswati	09/12/2024 : Mengerjakan ADT Stack 10/12/2023 : Mengerjakan ADT SetMap 12/12/2024 : Mengerjakan ADT LinkedList dan Store List Update 15/12/2024 : Mengerjakan History 16/12/2024 : Mengerjakan Cart Add dan Cart Remove 18/12/2024 - 21/12/2024 : Mengerjakan Main 22/12/2024 : Membantu laporan
18223136	Geraldo Linggom Samuel T.	16/12/2024 - 17/11/2024 : Mengerjakan CART SHOW, CART PAY 18/12/2024 : Penambahan fitur pada CARTPAY yaitu mengembalikan barang untuk history 20/12/2024 : Testing sesuai pembagian fungsi 20/12/2024 - 21/12/2024 : Mengerjakan laporan
18223125	Matilda Angelina Sumaryo	15/12/2024 : Mulai melakukan revisi load dan start 16/12/2024 : Mengerjakan wishlist clear, mulai melakukan revisi save 19/12/2024 - 20/11/2024 : Mengerjakan profile, main, dan merge 20/12/2024 : Memperbaiki wishlist remove, wishlist remove <i>, dan wishlist swap <i> <j> 21/12/2024 : Membantu laporan
18222133	Hanan Fitra Salam	16/12/2024 : Mengerjakan Wishlist Add dan Wishlist show 20/12/2024 : Mengerjakan laporan

## 9.4 Notulensi Asistensi

### 9.4.1 Notulensi Asistensi II

<b>Tanggal : Rabu, 18 Desember 2024</b>	Catatan Asistensi: - Save: - Untuk test case seperti save yang mengharuskan input ada tambahan itu harus ikutin spesifikasi, ngga bisa didalemnya ditambahin input lain  - Laporan: - Karena laporan milestone 1 dan 2 terpisah, untuk ringkasan tugas besarnya gunain dari milestone 1 dan ditambahin sama fitur baru dari milestone 2 nya  - History: - Untuk history kan liat harga terbesar atau lexical, nanti status nya itu ditaro di cart pay, history itu buat nunjukin aja
<b>Tempat : Zoom Meeting</b>	
<b>Kehadiran Anggota Kelompok:</b> No NIM Tanda tangan  1 18222133  2 18223109	



 3 18223110  4 18223136  5 18223125 	<ul style="list-style-type: none"> <li>- Untuk stack nya, ngikutin urutan stack aja ya, yang paling atas itu berarti yang paling baru</li> <li>- Wishlist Add: <ul style="list-style-type: none"> <li>- Untuk wishlist add itu kalo ngga ada di list barang juga ngga bisa ditambahin, sesuai dengan test case 3 di spesifikasi ya</li> </ul> </li> <li>- ADT Stack: <ul style="list-style-type: none"> <li>- Untuk adt stack itu ngga perlu dinamis, kalo untuk adt stack statis juga ngga perlu banyak-banyak alokasinya</li> </ul> </li> </ul>
	<p><b>Tanda Tangan Asisten:</b></p>  18221067 <b>Fawwaz Abrial Saffa</b>