

BAB 3

METODE PENELITIAN

Penelitian ini adalah penelitian rancang-bangun. Penelitian berawal dari latar belakang permasalahan yang ada, memetakan proses-proses, mencari sumber permasalahan, dan akhirnya merancang dan mengembangkan suatu sistem yang dapat menampilkan data yang telah diambil melalui proses scraping kedalam bentuk grafik yang bermanfaat untuk analisis harga maupun monitoring harga rata-rata komponen komputer di beberapa Marketplace yaitu Tokopedia, Shopee dan Blibli, aplikasi yang dikembangkan akan berbasis web.

3.1 BAHAN DAN ALAT PENELITIAN

Bahan penelitian didapatkan dengan menggali informasi melalui metode scraping data pada ketiga Marketplace yaitu Tokopedia, Blibli dan Shopee. Selain dari Marketplace referensi penelitian didapat dari jurnal, dan skripsi yang telah dibuat oleh peneliti sebelumnya dengan tema yang mirip.

Alat yang digunakan dalam penelitian ini adalah komputer dengan spesifikasi cukup untuk menjalankan sistem operasi dan perangkat lunak pengembangan serta koneksitas Internet.

Sistem Operasi dan program-program aplikasi yang dipergunakan dalam pengembangan aplikasi ini adalah:

1. Sistem Operasi: Windows 10 atau versi terbaru dan MacOS.
2. Framework: ReactJS, Flask
3. Database engine: MongoDB 5.0.
4. Bahasa Pemrograman: Python 3.9+
5. Text Editor: Visual Studio Code.

3.2 JALAN PENELITIAN

Dalam pengembangan sistem platform analisis komponen harga dari Marketplace menggunakan metodologi prototype. metode prototype merupakan salah satu metodologi dari sebuah *Software Development Life Cycle* (SDLC) yang

menggunakan prototype untuk menggambarkan sistem yang akan dibuat kepada calon pengguna sehingga pengguna dapat mempunyai gambaran yang jelas terhadap sistem yang akan dibangun. Pada penelitian ini secara umum terbagi menjadi 6 tahap yaitu:

3.2.1 Analisis Kebutuhan

Pada tahap ini pengumpulan data dilakukan dengan cara melakukan web scraping pada Tokopedia, Shopee, dan Blibli demi memenuhi kebutuhan perangkat lunak dan pengguna. Pustaka Python yang diunakan untuk web scraping yaitu Selenium.

3.2.2 Desain Apilkasi

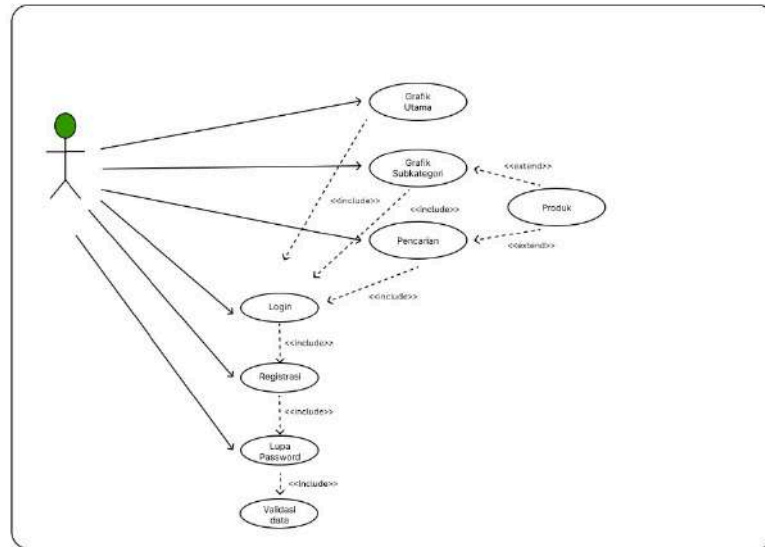
Sebelum menulis code dalam pembuatan apilkasi, desain dibuat dengan menggunakan apilkasi open-source yaitu Figma. Tahap ini berguna untuk memberikan gambaran meliputi:

- *Use Case dan Activity Diagram*
- *Software Architecture*
- *User Interface*
- *Database*

3.2.3 Use Case dan Activity Diagram.

Dalam pengembangan sistem analisis Cparts *Use Case Diagram* dibuat untuk menggambarkan interaksi antara pengguna dengan sistem Cparts. Sistem ini memiliki 6 Fitur utama yaitu *login, register, lupa kata sandi, menu grafik seluruh sub-kategori, menu grafik berdasarkan ub-kategori dan pencarian produk berdasarkan kata kunci dan sub-kategori*. Pengguna dapat mengakses platform ini dengan mendaftar terlebih dahulu pada sistem, lalu masuk ke menu utama dengan akun yang sudah dibuat.

Untuk gambar dan tabel penjelasan *Use Case Diargam* dapat dilihat pada Gambar 3.1 dan Tabel 3.1.



Gambar 3.1 Use Case Diagram

Tabel 3.1 Tabel Penjelasan Use Case Diagram

No	Nama Use Case	Penjelasan	Aktor yang terlibat (pengguna)
1.	Registrasi	Use Case ini menggambarkan aktivitas yang dilakukan oleh pengguna, dalam melakukan registrasi akun kedalam sistem agar pengguna dapat mengakses halaman utama dengan melakukan login.	<ul style="list-style-type: none"> • Penjual komputer • Penambang crypto • Perakit komputer • Konsumen
2.	Login	Use Case ini menggambarkan aktivitas yang dilakukan oleh pengguna, dalam melakukan login kedalam sistem Cparts.	<ul style="list-style-type: none"> • Penjual komputer • Penambang crypto • Perakit komputer • Konsumen
3.	Lupa Password	Use Case ini menggambarkan aktivitas yang dilakukan oleh pengguna apabila pengguna melupakan kata sandi pada akun yang dimiliki pengguna dengan cara memasukan email yang terdaftar kedalam sistem.	<ul style="list-style-type: none"> • Penjual komputer • Penambang crypto • Perakit komputer • Konsumen

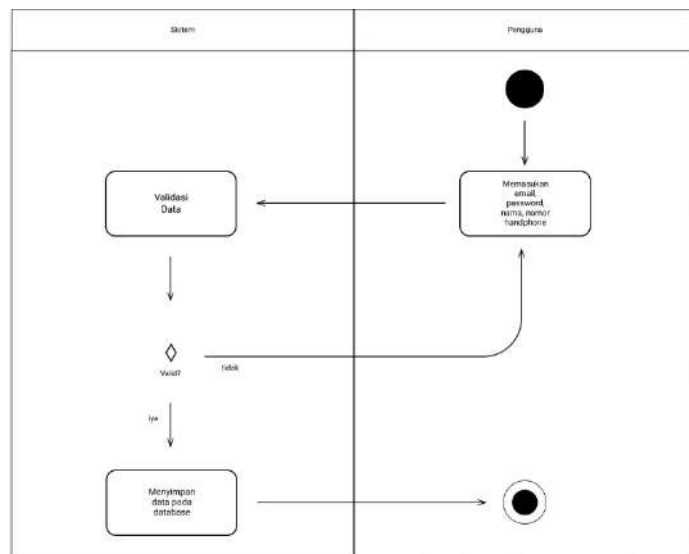
4.	Validasi Data	<i>Use Case</i> ini merupakan bagian dari aktivitas login, registrasi dan lupa password, sistem melakukan validasi data akun yang dimasukan oleh pengguna.	<ul style="list-style-type: none"> • Penjual komputer • Penambang crypto • Perakit komputer • Konsumen
5.	Grafik Utama	<i>Use Case</i> ini menggambarkan aktivitas yang dilakukan oleh pengguna dalam mengakses halaman utama, sistem menampilkan grafik berdasarkan seluruh subkategori yang tersedia.	<ul style="list-style-type: none"> • Penjual komputer • Penambang crypto • Perakit komputer • Konsumen
6.	Grafik Subkategori	<i>Use Case</i> ini menggambarkan aktivitas yang dilakukan oleh pengguna dalam mengakses halaman subkategori, yang menampilkan grafik rata-rata harga komponen komputer berdasarkan Marketplace, beserta produk yang tersedia sesuai dengan subkategori yang dipilih.	<ul style="list-style-type: none"> • Penjual komputer • Penambang crypto • Perakit komputer • Konsumen
7.	Pencarian	<i>Use Case</i> ini menggambarkan aktivitas yang dilakukan oleh pengguna dalam mengakses halaman pencarian dan mencari produk berdasarkan subkategori dan kata kunci yang dimasukan pada sistem, hasil dari pencarian adalah grafik utama dan grafik subkategori beserta produk yang sesuai dengan pencarian yang telah dilakukan pengguna.	<ul style="list-style-type: none"> • Penjual komputer • Penambang crypto • Perakit komputer • Konsumen

Pengguna memiliki alur kegiatan secara umum yang dapat digambarkan dengan *activity diagram*, pada sistem Cparts memiliki 6 *activity diagram* yaitu

halaman utama, halaman sub-kategori, halaman pencarian, login, registrasi, dan lupa password.

1. Registrasi

Sebelum pengguna dapat mengakses platform analisis Cparts, pengguna diharuskan untuk mendaftar terlebih dahulu, dengan memasukan nama, email, katasandi dan nomor ponsel pada halaman registrasi, setelah itu sistem akan melakukan pengecekan data pada database apabila akun email yang akan didaftarkan sudah terdaftar pada database maka pengguna diminta untuk memasukan kembali alamat email yang berbeda, jika sistem tidak menemukan email yang akan didaftarkan pada database maka sistem akan memasukan data pengguna kedalam database dan membuat akun Cparts. *Activity Diagram* registrasi dapat dilihat pada Gambar 3.2.



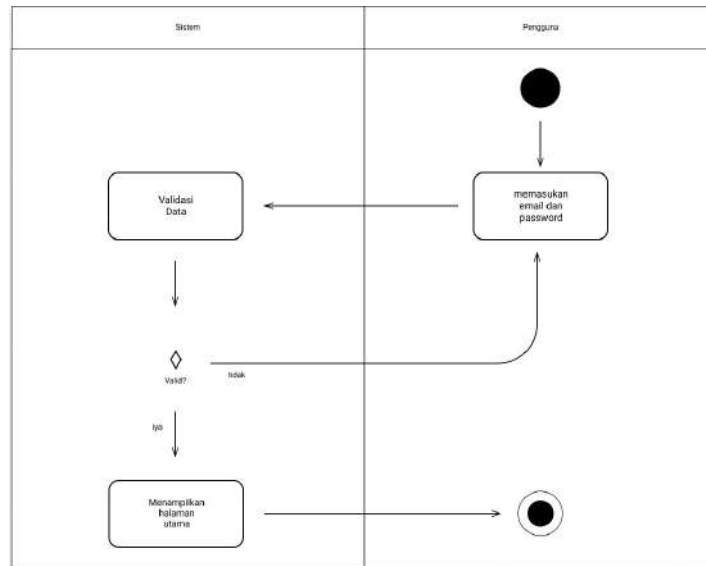
Gambar 3.2 *Activity Diagram* registrasi

2. Login

Setelah pengguna memiliki akun Cpart, pengguna dapat memasukan email dan password yang sudah terdaftar, kemudian sistem akan melakukan validasi data *login* apabila tidak data terverifikasi maka pengguna diminta untuk memasukan kembali email dan password, jika data login sudah terverifikasi maka sistem akan menampilkan halaman utama dan

memberikan akses kepada pengguna untuk menggunakan fitur pada platform Cparts.

Activity Diagram login dapat dilihat pada Gambar 3.3 di bawah ini

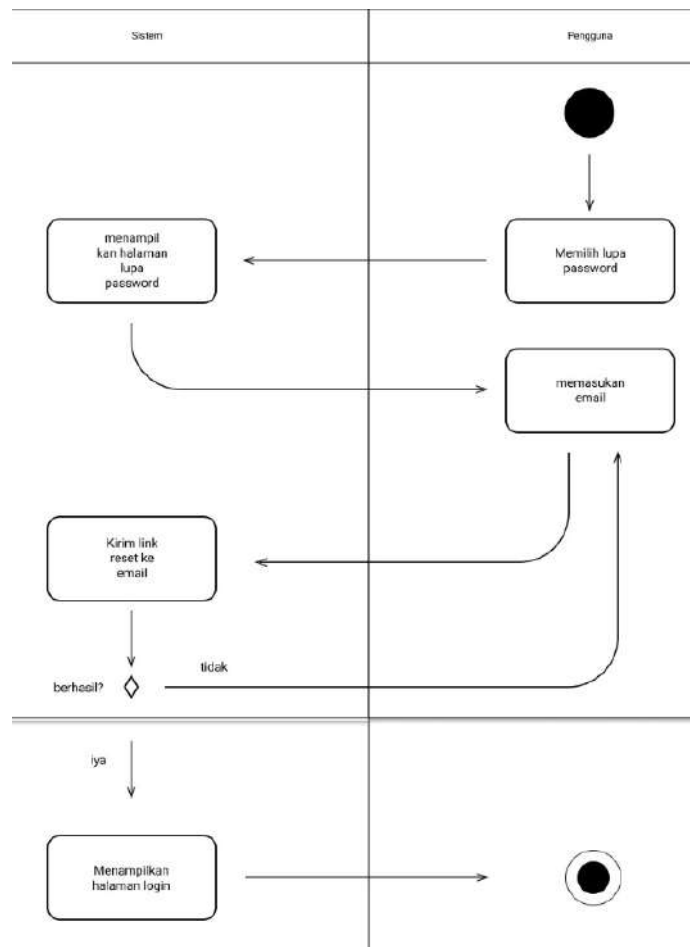


Gambar 3.3 *Activity Diagram Login*

3. Lupa Password

Pada kasus tertentu pengguna melupakan kata sandi dari akun yang sudah dibuat, sistem menyediakan fitur untuk mereset kata sandi pengguna dengan mengirimkan email kepada pengguna. Untuk melakukannya pengguna memilih lupa password pada halaman *login*, kemudian memasukan email pada kolom masukan yang disediakan, setelah itu sistem akan mengirimkan link untuk mengubah kata sandi ke email pengguna apabila tidak berhasil pengguna diminta untuk memasukan kembali alamat email, jika berhasil maka sistem akan menampilkan halaman login.

Activity Diagram lupa password dapat dilihat pada Gambar 3.4 di bawah ini

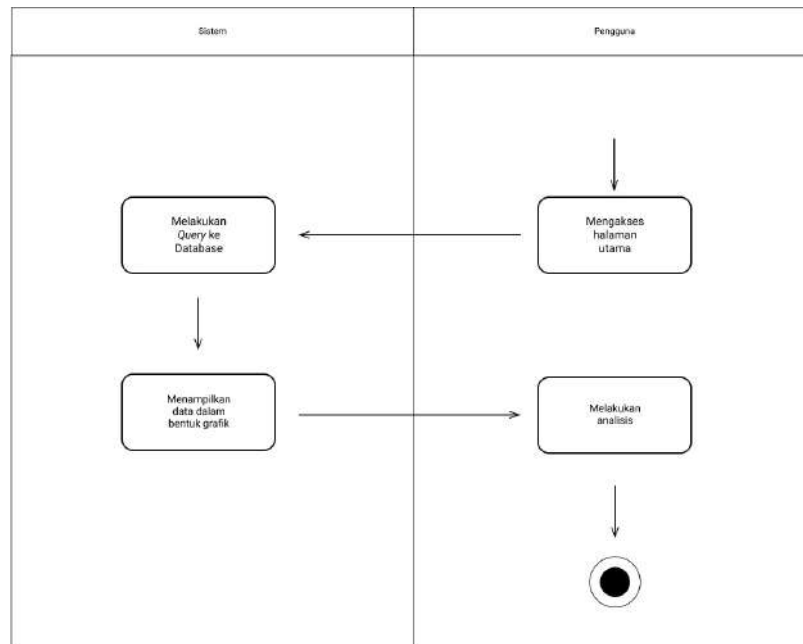


Gambar 3.4 Activity Diagram Lupa Password

4. Halaman Utama

Aktivitas yang dilakukan oleh pengguna dalam halaman utama yaitu pengguna mengakses halaman utama, yang kemudian sistem akan melakukan pengambilan data grafik dari database yang akan divisualisasikan dalam bentuk grafik, setelah data ditampilkan pengguna dapat melakukan analisis terhadap perkembangan harga komponen komputer pada halaman utama, grafik yang ditampilkan pada halaman utama adalah data yang dibagi berdasarkan subkategori dari ketiga marketplace.

Activity diagram halaman utama dapat dilihat pada Gambar 3.2 dibawah ini.

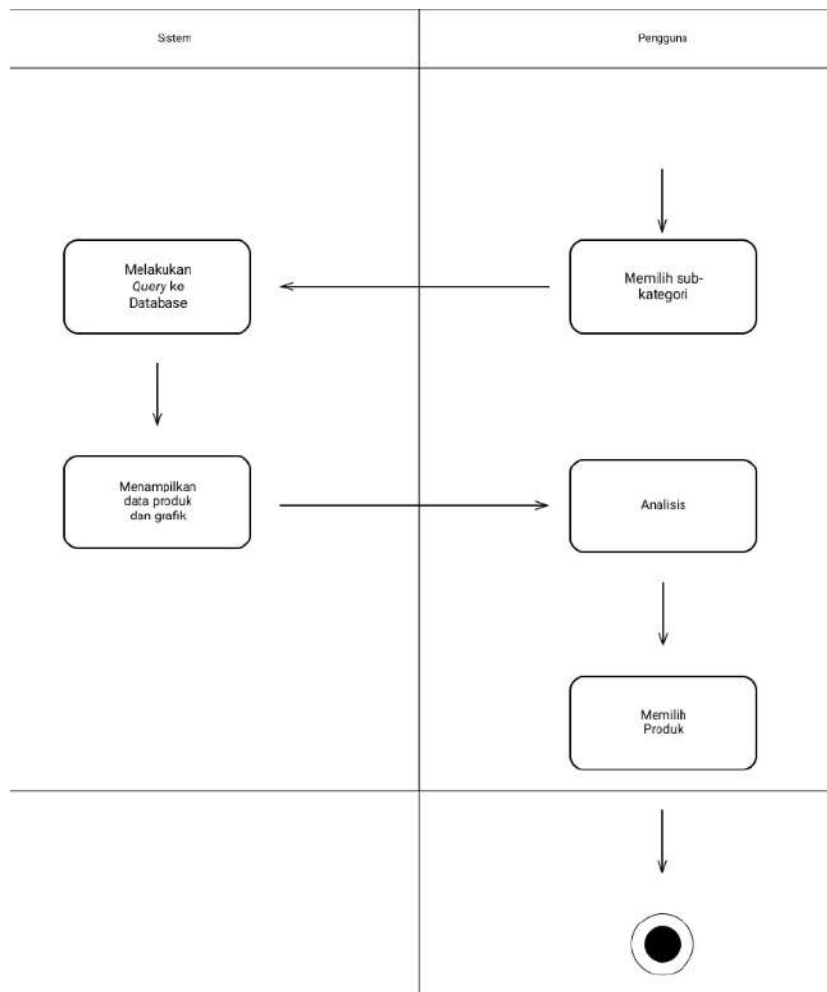


Gambar 3.5 *Activity Diagram* Halaman Utama

5. Sub-kategori

Pada halaman utama pengguna dapat memilih subkategori yang tersedia seperti GPU, CPU, *Motherboard*, *Case*, dan *Storage*. Apabila sub kategori telah dipilih sistem akan melakukan pengambilan data berdasarkan subkategori yang dipilih, kemudian sistem akan menampilkan data grafik yang membandingkan harga dari ketiga marketplace berdasarkan kategori yang dipilih dan sistem juga akan menampilkan produk terkait. Jika semua data sudah disajikan pengguna dapat melakukan analisis kemudian memilih produk yang ingin dilihat.

Activity diagram halaman sub kategori dapat dilihat pada Gambar 3.6 dibawah ini.



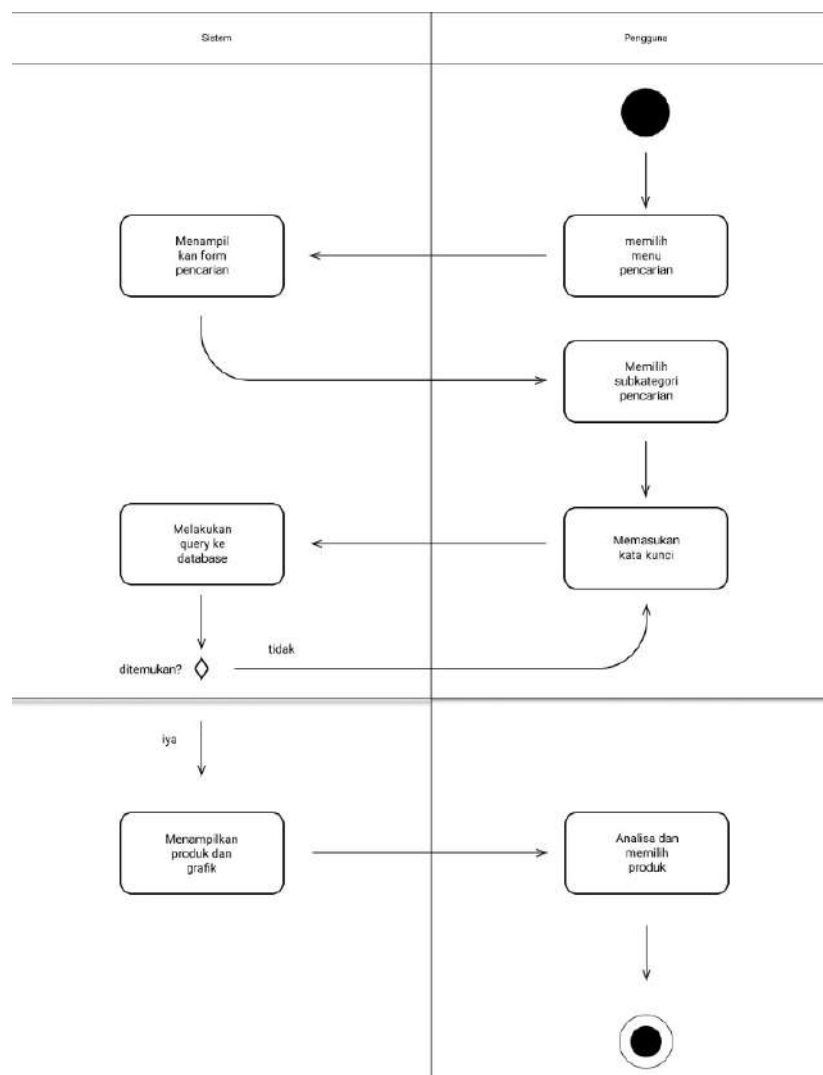
Gambar 3.6 Activity Diagram Halaman Sub Kategori

6. Pencarian

Pada sistem ini pengguna dapat melakukan pencarian produk berdasarkan subkategori dan kata kunci sesuai dengan yang pengguna inginkan, untuk melakukan pencarian pengguna terlebih dahulu harus memilih menu pencarian, kemudian sistem akan menampilkan pilihan sub kategori dan kolom masukan kata kunci. Pengguna dapat memilih subkategori dan memasukan kata kunci untuk melakukan pencarian, setelah dilakukannya pencarian sistem akan melakukan *query* kedalam database berdasarkan parameter pencarian, apabila tidak ditemukan pengguna diminta untuk memasukan kata kunci lain atau memilih sub kategori yang berbeda, dan jika hasil ditemukan maka sistem akan menampilkan grafik

dan produk sesuai dengan pencarian pengguna, setelah itu pengguna dapat melakukan analisis dan memilih produk dari hasil pencarian.

Activity Diagram dari pencarian produk dapat dilihat pada Gambar 3.7 dibawah ini.

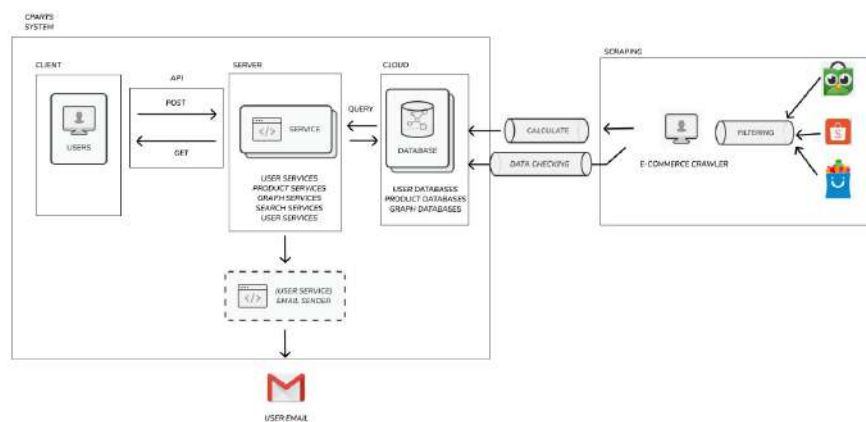


Gambar 3.7 *Activity Diagram* Pencarian

3.2.4 Software Architecture

Arsitektur cpart terdiri dari 4 bagian utama yaitu *Client*, *API*, *Server*, dan *Cloud Database* (MongoDB). komunikasi antara *Client* dengan server melalui sebuah *API* yang dibangun menggunakan *Python* (*Flask*). Seluruh permintaan atau

request ke *Server* akan diolah terlebih dahulu berdasarkan servis yang dituju, sistem cparts memiliki 4 servis utama yaitu User Services, Graph Services, Search Services, Product Services. Selain itu pada sistem ini memiliki sistem *Scraping* guna memenuhi data historical yang dibutuhkan oleh sistem utama Cparts, yang berjalan secara terpisah dari sistem utama dan *Scraping* dilakukan secara berkala pada tiap minggu. Struktur Sistem dapat dilihat pada Gambar 3.8 dibawah ini:



Gambar 3.8 Arsitektur Sistem Cparts

1. User Services

User Services menangani segala macam permintaan terkait dengan pengguna dan menjadi perantara yang menghubungkan data dari database ke *Client* (Pengguna), berikut beberapa hal yang ditangani oleh *User Services*:

- a. Melakukan verifikasi akun pengguna yang login dengan database.
- b. Melakukan pendaftaran akun pengguna baru kedalam database.
- c. Enkripsi kata sandi pengguna yang akan dimasukkan kedalam database.
- d. Mengirim email ke alamat email yang dikirimkan oleh pengguna apabila pengguna lupa password.
- e. Melakukan pergantian kata sandi lama dengan kata sandi baru.

Untuk mengakses *User Services* dapat mengakses beberapa *endpoint API* yang telah dibuat dapat dilihat pada *source code* dibawah ini:

```

1 from flask import Blueprint, request
2 from ..services.user_service import signup_service, login_service, send_email, change_user_password, reset_password
3
4 user_route = Blueprint('user_route', __name__)
5
6 @user_route.route("/api/user/signup", methods=['POST'])
7 def signup():
8     data = request.get_json()
9     return signup_service(data)
10
11
12 @user_route.route("/api/user/login", methods=['POST'])
13 def login():
14     data = request.get_json()
15     return login_service(data)
16
17 @user_route.route("/api/user/forgot-password", methods=['POST'])
18 def forgot_password():
19     data = request.get_json()
20     return send_email(data)
21
22 @user_route.route("/api/user/change-password", methods=["POST"])
23 def change_password():
24     data = request.get_json()
25     return change_user_password(data)
26
27 @user_route.route("/api/user/forgot-password/<id_key>", methods=["GET", "POST"])
28 def check_reset_pass(id_key):
29     if request.method == 'GET':
30         return reset_password(id_key, check_only=True)
31     data = request.get_json()
32     return reset_password(data)
33

```

Gambar 3.9 *User Service Endpoints*

Pada Gambar 3.9 terdapat 5 fungsi untuk mengakses masing-masing *endpoint* untuk menerima *request* dari sisi *client* yaitu:

- a. Fungsi *signup* (Baris kode ke-6)

Menangani data *request* dengan *method POST* pada *endpoint* */api/user/signup* yang akan diolah oleh *signup_service*

- b. Fungsi *login* (Baris kode ke-13)

Menangani data *request* dengan *method POST* pada *endpoint* */api/user/login* yang akan diolah oleh *login_service*

- c. Fungsi *forgot_password* (Baris kode ke-17)

Menangani data *request* dengan *method POST* pada *endpoint* */api/user/forgot-password* yang akan diolah oleh fungsi *send_email* yang merupakan bagian dari *user_service*

- d. Fungsi *change_password* (Baris kode ke-22)

Menangani data *request* dengan *method POST* pada *endpoint /api/user/change-password* yang akan diolah oleh fungsi *change_user_password* yang merupakan bagian dari *user_service*

e. Fungsi *check_reset_pass* (Baris kode ke-27)

Menangani data *request* dengan *method GET* dan *POST* pada *endpoint /api/user/forgot-password/<id_key>*, kedua *request* tersebut ditangani oleh fungsi *reset_password* yang dibedakan dengan parameter tambahan yaitu *check_only* untuk *method GET*.

```

12 def signup_service(user_credentials):    disastah007 [9 days ago] • add new endpoints
13     try:
14         user = User()
15         email_check = user.is_exist({'email': user_credentials['email']})
16         if email_check:
17             return {"status": 403, "message": "email already exists"}
18         else:
19             register = {}
20             register['firstName'] = user_credentials['firstName']
21             register['lastName'] = user_credentials['lastName']
22             register['email'] = user_credentials['email']
23             register['password'] = encrypt_password(user_credentials['password'])
24             register['number'] = user_credentials['number']
25
26             user.add_user(register)
27
28             return make_response({'message': 'successfully registered', 'status': 200}, 200)
29
30     except Exception as e:
31         return make_response({'message': str(e)}, 404)
32
33
34 def login_service(user_credentials):
35     try:
36         user = User()
37         email_check = user.is_exist({'email': user_credentials['email']})
38         account = user.find_user({'email': user_credentials['email']})
39         if email_check:
40             payload = {
41                 'email': account['email'],
42                 '_id': str(account['_id']),
43                 'exp': datetime.datetime.utcnow(
44                     ) + datetime.timedelta(days=1)}
45
46             secret = TOKEN_SECRET
47             if compare_passwords(user_credentials['password'], account['password']):
48                 token = generate_token(payload, secret)
49                 return make_response({'access_token': token}, 200)
50             else:
51                 return make_response({'message': 'Invalid password'}, 403)
52
53         else:
54             return make_response({'message': 'user not found!'}, 404)
55
56     except Exception as e:
57         return make_response({'message': str(e)}, 500)
58

```

Gambar 3.10 Source Code User Service

Untuk *Source Code* pada gambar 3.10 dari *User Service* terdapat dua fungsi pokok dalam menangani data user yaitu fungsi *signup_service* dan fungsi *login service* yang terdapat pada baris kode ke-12 dan ke-34.

Fungsi *login service* memuat data *model* dari user lalu melakukan pengecekan apakah data user yang dikirimkan sudah terdaftar pada *database*. Lalu pada baris kode ke-47 *login service* melakukan perbandingan masukan kata sandi oleh pengguna dengan data pengguna yang ada di *database*, apabila cocok maka user dapat untuk mengakses halaman utama *Cparts*, sedangkan fungsi *signup service* pada baris kode ke-15 melakukan pengecekan email dari *database* berdasarkan data yang dikirimkan pengguna apabila sudah terdaftar maka akan memberikan *response* email sudah terdaftar, apabila email belum terdaftar ada sistem maka *signup service* akan melakukan enkripsi kata sandi yang dilakukan pada baris kode ke-23 dan mendaftarkan akun kedalam sistem.

2. Product Services

Seluruh produk yang ditampilkan pada sisi *Client* atau Pengguna diolah oleh *Product Services*, pada servis ini sistem melakukan *Query* kedalam *database* sesuai dengan permintaan dari sisi *Client*. Berikut tanggung jawab *Product Service* pada sistem *Cpart*:

- a. Melakukan pengambilan data produk pada *database*
- b. Kalkulasi total halaman yang dihasilkan dari produk dalam *database*
- c. Melakukan paginasi
- d. Penyaringan data produk dari *database*

Untuk mengakses *Product Services* terdapat beberapa *endpoints* yang dapat diakses pada gambar dibawah ini:

```

1  from flask import Blueprint, request
2  from ..services.product_service import load_product, load_product_by_page
3
4  product_route = Blueprint('product_route', __name__)
5
6
7  @product_route.route("/api/products/<category>", methods=['GET'])
8  def get_products(category):
9      page_limit = request.args.get("page_limit")
10     page = request.args.get("page")
11     product_name = request.args.get('product_name', None)
12
13     if not page_limit:
14         if product_name:
15             return load_product(category, product_name)
16         return load_product(category)
17     if product_name:
18         return load_product_by_page(category, int(page), int(page_limit), product_name)
19     return load_product_by_page(category, int(page), int(page_limit))
20

```

Gambar 3.11 *Product Service Endpoints*

Pada gambar 3.11 *endpoint* `/api/products/<category>` memiliki *method* GET yang menerima parameter `category` dari url *endpoint* kemudian selain itu juga menerima argument-argument lain seperti `page_limit` untuk maksimal produk yang ditampilkan, `page` yang merupakan halaman produk saat ini dan nama produk untuk menyaring berdasarkan nama produk. Pada baris kode ke-13 dilakukan pengecekan apakah ada limitasi halaman jika tidak ada maka akan diolah oleh fungsi `load_product` dari `product_service`, jika ada akan diolah oleh fungsi `load_product_by_page`.

Source Code dari *Product Services* dapat dilihat pada gambar dibawah ini:

```

10 / services > product_service.py / ...
1  from ..model.product import Product
2  from flask import make_response
3
4
5  def load_product(category, product_name=None):
6      try:
7          database = Product()
8          if product_name:
9              products = database.get_product_by_name(category, str(product_name))
10             return make_response({'products': products}, 200)
11
12             products = database.get_product_by_category(category)
13
14             return make_response({'products': products}, 200)
15
16         except Exception as e:
17             return make_response({'message': str(e)}, 404)
18
19
20 def load_product_by_page(category, page:int, limit_page:int, product_name:str = None):
21     try:
22         database = Product()
23         first_index = (limit_page*page)-limit_page
24         last_index = limit_page*page
25         if product_name:
26             products, total_doc = database.get_by_page(category, first_index, last_index, str(product_name))
27         else:
28             products, total_doc = database.get_by_page(category, first_index, last_index)
29         pages = round(total_doc/limit_page)
30
31         return make_response({
32             'page_limit': limit_page,
33             'current_page': page,
34             'total_page': pages,
35             'products': products
36         }, 200)
37
38     except Exception as e:
39         return make_response({'message': str(e)}, 404)
40

```

Gambar 3.12 Source Code Product Service

Dapat dilihat pada Gambar 3.12 terdapat dua fungsi utama pada *product service* yaitu *load_product* dan *load_product_by_page*. Untuk fungsi *load_product*, fungsi ini menerima dua parameter yaitu nama kategori dan nama produk dengan nilai awal *None* atau kosong, pada baris kode ke-8 dan 9 sistem melakukan pengecekan apakah terdapat penyaringan berdasarkan nama produk atau tidak apa bila ada maka sistem akan melakukan *query* kedalam database berdasarkan nama produk, apabila parameter nama produk kosong maka sistem akan *query* kedalam database berdasarkan kategori dapat dilihat pada baris kode ke-12.

3. Graph Services

Servis yang berperan dalam menangani penyajian grafik kepada pengguna dilakukan oleh *Graph Services*, servis ini melakukan *Query* kedalam database kategori maupun database grafik bulanan yang telah di

olah terlebih dahulu oleh mesin scraping sebelum data tersebut dimasukkan ke dalam database.

Client dapat mengakses beberapa *endpoint* untuk menggunakan *Graph Services*, terdapat pada gambar pada dibawah ini:

```

1 from flask import Blueprint, request
2 from ..services.graph_service import get_main_graph, get_category_graph, get_main_graph_by_year
3
4 graph_route = Blueprint('graph_route', __name__)
5 category_route = Blueprint('category_route', __name__)
6
7 @graph_route.route("/api/graphs/main", methods=['GET'])
8 def get_main():
9     return get_main_graph()
10
11 @graph_route.route("/api/graphs/main/<year>", methods=['GET'])
12 def get_main_by_year(year):
13     return get_main_graph_by_year(year)
14
15 @graph_route.route("/api/graphs/category/<category>", methods=['GET'])
16 def get_category(category):
17     return get_category_graph(category)
18

```

Gambar 3.13 *Graph Service Endpoints*

Pada gambar 3.13 terdapat tiga *endpoint* grafik yaitu *get_main* pada baris kode ke-7 yang menangani penampilan data grafik utama tanpa adanya *filter* apa pun pada api *endpoint*, lalu *get_main_by_year* pada baris kode ke-11 dengan parameter atau argumen tahun pada api *endpoint* sebagai filter dalam pengambilan data grafik berdasarkan tahun, dan yang terakhir fungsi *get_category* pada baris kode ke-15 yang menangani penambilan data grafik berdasarkan parameter atau argument sub kategori yang diberikan. Pengolahan seluruh data grafik dilakukan oleh *graph service*.

Untuk *Source Code* dari *Graph Services* dapat dilihat pada gambar dibawah ini:

```

1 from ..model.graph import Graph
2 from flask import make_response
3
4
5 def get_main_graph():
6     try:
7         database = Graph()
8         data = database.get_graph_main()
9         result = [graph for graph in data]
10
11         return make_response({'graphs': result}, 200)
12
13     except Exception as e:
14         return make_response({'message': str(e)}, 404)
15
16 def get_main_graph_by_year(year):
17     try:
18         database = Graph()
19         data = database.get_graph_main_by_year(year)
20         result = [graph for graph in data]
21
22         return make_response({'graphs': result}, 200)
23
24     except Exception as e:
25         return make_response({'message': str(e)}, 404)
26
27 def get_category_graph(category):
28     try:
29         database = Graph()
30         data = database.get_graph_by_category(category)
31         result = [graph for graph in data]
32
33         return make_response({'graphs': result}, 200)
34
35     except Exception as e:
36         return make_response({'message': str(e)}, 404)
37

```

Gambar 3.14 Source code Graph Services

Dapat dilihat pada gambar 3.14 pada baris kode ke-5 terdapat fungsi *get_main_graphs* yang melakukan *query* atau pengambilan data grafik dari database yang kemudian dimasukan kedalam sebuah *list* yang menghasilkan *response* data grafik olahan, selain itu pada baris kode ke-16 terdapat sebuah fungsi *get_main_graph_by_year* yang melakukan pengambilan data dari database dengan parameter tahun sebagai *filter* data yang akan di ambil berdasarkan tahun, dan untuk fungsi *get_category_graph* pada baris kode ke-27 memiliki parameter sub-kategori yang berguna untuk penyaringan data dari database berdasarkan sub-kategori yang dipilih yang menghasilkan data grafik sub kategori.

4. Search Services

Pengguna dapat melakukan pencarian produk, tipe komponen tertentu pada platform *Cparts*, kegiatan pencarian pengguna dikerjakan oleh *Search Services*. Berikut beberapa hal yang dilakukan didalam *Search Services*:

- a. Melakukan query database berdasarkan sub-kategori yang dipilih
- b. Melakukan query kedalam database berdasarkan kata kunci yang dimasukan oleh pengguna

- c. Mengolah data produk kedalam beberapa data grafik yang siap disajikan kepada pengguna
- d. Menampilkan produk sesuai dengan sub-kategori yang dipilih dan kata kunci pencarian

Pada gambar dibawah ini dapat dilihat *Source Code* dan *Endpoints* dari *Search Services*:

```

1  from flask import Blueprint, request
2  from ..services.search_service import find_product
3
4  search_route = Blueprint('search_route', __name__)
5
6  @search_route.route("/api/products/search", methods=["POST"])
7  def search_product():
8      request_data = request.get_json()
9      category = request_data.get('category')
10     product_name = request_data.get('product')
11
12     return find_product(category, product_name)

```

Gambar 3.15 *Search Services Endpoints*

Pada gambar 3.15 baris kode ke-6 memperlihatkan sebuah fungsi yang menangani *api endpoint* dari pencarian produk dengan *method POST* yang terdiri dari data request yaitu nama kategori dan nama produk yang akan dicari, lalu data tersebut diolah oleh *search service*.

```

3  from flask import make_response
4
5  from ..model.search import Search
6  from ..utils.json_reformat import reformat_graph_from_aggregate
7
8
9  def find_product(category, search):
10     year_filter = str(datetime.now().year)
11     try:
12         database = Search()
13
14         aggregate_category_result = database.find_specific_product_category(category, search, year_filter)
15         aggregate_main_result = database.find_specific_product_main(category, search, year_filter)
16
17         graph_category = reformat_graph_from_aggregate(aggregate_category_result, year_filter, category=True)
18         graph_main = reformat_graph_from_aggregate(aggregate_main_result, year_filter)
19
20         # products = database.get_product_by_name(category, search)
21
22         return make_response({"graphCategory": graph_category, "graphMain": graph_main}, 200)
23
24     except Exception as e:
25         return make_response({'message': str(e)}, 404)
26

```

Gambar 3.16 *Search Services*

Pada gambar 3.16 terdapat sumber kode dari *search services* yang memiliki fungsi utama *find_product* yaitu mencari berdasarkan subkategori

dan kata kunci pencarian yang terdapat pada baris kode ke-9. Fungsi *find_product* melakukan pengambilan data dari *database* sebanyak dua kali yang pertama untuk pengambilan data grafik dari tiga *Marketplace* yang terdapat pada baris kode ke 14 lalu pengambilan data grafik utama yang berada pada baris kode ke-15. setelah melakukan *query* kedalam database hasil dari *query* atau data tersebut dilakukan pemformatan ulang agar data menjadi siap pakai yang dilakukan pada baris kode ke-17 dan 18.

3.2.5 Desain Database

Platform analysis Cparts menggunakan mongoDB sebagai database yang menyimpan data Produk. User dan Grafik. Pada database mongoDB terdiri dari nama database, nama koleksi, dan dokumen. Database pada mongoDB merupakan kumpulan dari beberapa koleksi, sedangkan koleksi merupakan kumpulan dari beberapa dokumen.

1. User

Pengguna yang melakukan registrasi kedalam sistem Cparts akan dimasukan kedalam database User, untuk kata sandi sebelum kata sandi dimasukan kedalam database, dilakukan hashing kedalam *binary base64* demi keamanan kata sandi pengguna. Struktur dokumen dapat dilihat pada Gambar 3.17 dibawah ini:.

```

{
  "_id": {
    "$oid": "61cbc6d4a92527591b74bb52"
  },
  "firstName": "user",
  "lastName": "dummy",
  "email": "user@email.com",
  "password": {
    "$binary": {
      "base64": "jDjiJDEyJGk3VWdVWXlTeTnyYjFYOGda0DNMLk9GZ0t1c0Z0dFZSMWJnUGFaaIp5YLZxVHhpSEpCY1BL",
      "subType": "00"
    }
  },
  "number": "085499993333"
}

```

Gambar 3.17 Struktur Dokumen Koleksi pada database User

2. Produk

Untuk data produk diisi dari *Scraping* data menggunakan *Selenium* yang dijalankan secara berkala pada tiap minggunya, pada database terdiri dari

beberapa koleksi yaitu *Cpu*, *Gpu*, *Ram*, *Memory*, *Motherboard*, *Case* dan *Storage*.

Sebelum data dimasukkan kedalam database dilakukan pengecekan dan penyaringan apakah data yang akan dimasukkan kedalam database termasuk kedalam sub-kategori yang ada, atau data yang akan dimasukkan sudah ada didalam database. Struktur dokumen dapat dilihat pada Gambar 3.18 dibawah ini:

```
{
  "_id": {
    "SoId": "6194521d38a3ef046b47fd44"
  },
  "marketplace": "shopee",
  "title": "casing pc cube gaming cabamesh putih m-atx free 2 fan rgb",
  "price": 495000,
  "sold": 2,
  "city": "KOTA BANJARMASIN - BANJARMASIN UTARA, KALIMANTAN SELATAN, ID",
  "link": "https://shopee.co.id/casing-pc-cube-gaming-cabamesh-putih-m-atx-free-2-fan-rgb-i.155154804.13624368179?sp_atk=59f3522a-dd83-45c2-af9a-
  "extraction_date": "2021-11-17"
}
```

Gambar 3.18 Struktur dokumen produk

3. Grafik

Pada database grafik terdiri dari dua koleksi dengan nama grafik_bulanan yaitu data grafik untuk data pada grafik utama kemudian koleksi grafik_kategori yang berisi data grafik berdasarkan sub-kategori yaitu *Cpu*, *Gpu*, *Memory*, *Motherboard*, *Storage* dan *Case*.

Data pada database ini didapatkan dengan mengolah data dari produk setelah pengambilan data selesai dilakukan pada tiap minggunya, setelah pengolahan data selesai data grafik akan diperbarui dengan data olahan yang baru. Struktur dokumen grafik bulanan dapat dilihat pada Gambar 3.19 dibawah:

```

{
  "_id": {
    "$oid": "62efb42a4dea39e051f0762b"
  },
  "year": "2022",
  "category": "cpu",
  "average": [
    4721135,
    4871335,
    4394707,
    5503611,
    5169322,
    5057141,
    5044776,
    0,
    0,
    0,
    0,
    0
  ],
  "minimum": [ ],
  "maximum": [ ]
}

```

Gambar 3.19 Struktur dokumen grafik bulanan

Pada koleksi database kategori data yang akan dimasukkan kedalamnya diolah terlebih dahulu dan dikategorikan berdasarkan *Marketplace* asal data tersebut, setelah itu dokumen lama akan diupdate dengan dokumen yang baru. Struktur dokumen dapat dilihat pada Gambar 3.20 dibawah:

```

{
  "_id": {
    "$oid": "62f0a7008aad20da8ca3474c"
  },
  "category": "memory",
  "sort_by": "average",
  "tokopedia": [
    659018,
    448909,
    536624,
    550750,
    567384,
    567384,
    538711,
    0,
    0,
    0,
    0,
    0
  ],
  "year": "2022",
  "shopee": [ ],
  "blibli": [ ]
}

```

Gambar 3.20 Struktur dokumen grafik kategori

3.2.6 Desain User Interface

Pada platform analisis Cparts memiliki 7 halaman utama yaitu *login*, registrasi, lupa *password*, *reset password*, halaman utama, halaman sub-kategori, dan pencarian. Berikut desain *user interface* dari ke-7 halaman tersebut:

1. Login

Halaman login memiliki dua buah form email dan *password* dengan tombol “sign in” untuk login, selain itu pada halaman ini pengguna juga dapat mengakses halaman registrasi dan lupa password.

Desain halaman login dapat dilihat pada Gambar 3.21

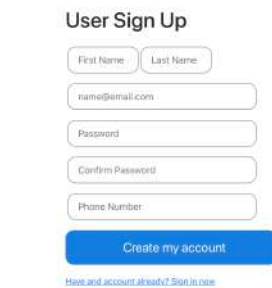


The image shows a user sign-in form. At the top center is a logo consisting of a stylized 'C' and 'P' inside a circle. Below the logo is the text 'User Sign in'. There are two input fields: the first is labeled 'Email' and the second is labeled 'Password'. Below these fields is a blue button with the text 'Sign In'. At the bottom of the form, there are two links: 'Does not have an account? Create now' and 'Forgot password?'.

Gambar 3.21 Desain Halaman Login

2. Registrasi

Halaman registrasi memiliki 6 buah formulir untuk diisi yaitu nama depan, nama belakang, alamat email, kata sandi, konfirmasi kata sandi, dan nomor telepon. Desain dari halaman registrasi dapat dilihat pada Gambar 3.22



The image shows a 'User Sign Up' form with the following fields: 'First Name' and 'Last Name' (two small input boxes), 'name@gmail.com' (email address), 'Password', 'Confirm Password', and 'Phone Number'. A blue 'Create my account' button is at the bottom, with a link 'Have an account already? Sign in now' below it.

Gambar 3.22 Desain Halaman Registrasi

3. *Lupa password*

Halaman lupa password memiliki 1 formulir yang harus diisi yaitu alamat email yang sudah terdaftar. Desain halaman lupa password dapat dilihat pada Gambar 3.21



The image shows a 'Cparts Forget Password' form with a single 'Email Address' input field containing 'user@gmail.com' and a blue 'Email My Password' button.

Gambar 3.23 Desain Halaman Lupa *Password*

4. *Reset password*

Halaman reset password memiliki dua kolom yang harus diisi yaitu kolom kata sandi baru dan konfirmasi kata sandi. Desain dapat dilihat pada Gambar 3.24



The image shows a 'PASSWORD RESET CPARTS' form with two input fields: 'New Password' and 'Confirm Password'. A blue 'Reset Password' button is at the bottom.

Gambar 3.24 Desain Halaman reset Password

5. *Halaman utama*

Pada halaman utama platform analisis cparts, pengguna dapat melihat data perkembangan harga komponen dalam bentuk grafik, selain itu

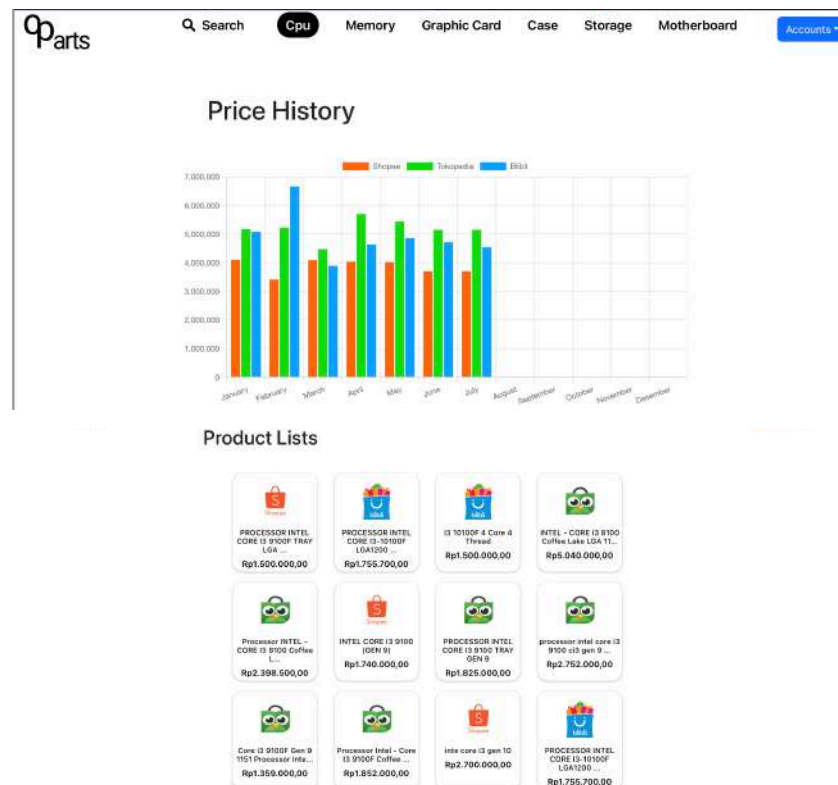
terdapat menu navigasi yang terdiri dari sub kategori, tombol akun, dan pencarian. Desain dari halaman utama dapat dilihat pada Gambar 3.25



Gambar 3.25 Desain Halaman Utama

6. Halaman sub-kategori

Untuk halaman sub-kategori memiliki persamaan pada bagian menu navigasi yang membedakan disini yaitu grafik pada halaman sub-kategori dibagi berdasarkan tiga *marketplace* yaitu Shopee, Tokopedia, Blibli beserta produk sesuai dengan sub kategori yang dipilih. Desain tampilan sub-kategori dapat dilihat pada Gambar 3.26



Gambar 3.26 Desain Halaman Sub Kategori

7. Pencarian

Halaman pencarian memiliki kolom pencarian dan menu sub kategori, yang dapat menampilkan dua buah grafik yaitu grafik perkembangan harga dari seluruh *marketplace* dan grafik berdasarkan *marketplace*, beserta produk sesuai dengan hasil pencarian yang ditemukan. Desain halaman pencarian dapat dilihat pada Gambar 3.27.



Gambar 3.27 Desain Halaman Pencarian

3.2.7 Pembangunan Prototype (Pengkodean)

Pada tahap ini penulisan kode dimulai, dengan menggunakan bahasa pemrograman *Python* dengan *library* Selenium sebagai scraper, untuk *Application Programming Interface* (API) menggunakan Flask sebagai *service backend*. Untuk pengembangan *frontend* bahasa yang digunakan yaitu *Javascript* dengan *framework* ReactJs yang dipasang kedalam NodeJs sebagai *server frontend*.

Sistem ini memiliki beberapa fungsi utama yaitu modul registrasi, *login*, pencarian, dan pengambilan data grafik, selain modul utama ada beberapa modul *utility* seperti enkripsi kata sandi dan modul untuk mengirim email.

1. Modul login

Modul ini memastikan pengguna yang *login* memiliki akun yang sudah terdaftar, dengan cara Ketika pengguna mengirimkan data *login* dengan akun yang dimasukan oleh pengguna, sistem akan membuka koneksi kedalam database mongoDB (baris kode ke-36), setelah itu sistem akan melakukan verifikasi terhadap akun pengguna (baris kode ke-37 hingga baris kode ke-46), setelah verifikasi berhasil sistem akan membuat token berdasarkan akun yang terverifikasi. *Source Code* dapat dilihat pada Gambar 3.28

```

36     user = User()
37     email_check = user.is_exist({'email': user_credentials['email']})
38     account = user.find_user({'email': user_credentials['email']})
39     if email_check:
40         payload = {
41             'email': account['email'],
42             '_id': str(account['_id']),
43             'exp': datetime.datetime.utcnow(
44                 ) + datetime.timedelta(days=1)}
45
46         secret = TOKEN_SECRET
47         if compare_passwords(user_credentials['password'], account['password']):
48             token = generate_token(payload, secret)

```

Gambar 3.28 *Source Code* Modul Login

2. Modul registrasi

Ketika pengguna melakukan registrasi pada platform analisis Cparts, modul registrasi mengatasi proses registrasi tersebut dengan tahapan pertama yaitu sistem akan membuka koneksi database pengguna (baris kode ke-14), lalu tahapan kedua sistem akan melakukan pengecekan email pengguna apakah akun dengan email yang pengguna daftarkan sudah ada di dalam database (baris kode ke-15). Setelah pengecekan berhasil selanjutnya sistem akan melakukan pendaftaran akun pengguna dengan memasukan informasi akun kedalam database (baris kode ke-16 hingga baris 26), dalam proses pendaftaran yang terjadi kata sandi akan dilakukan enkripsi terlebih dahulu (baris kode ke-23). *Soruce Code* dapat dilihat pada Gambar 3.29 dibawah ini.

```

14     user = User()
15     email_check = user.is_exist({'email': user_credentials['email']})
16     if email_check:
17         return {"status": 403, "message": "email already exists"}
18     else:
19         register = {}
20         register['firstName'] = user_credentials['firstName']
21         register['lastName'] = user_credentials['lastName']
22         register['email'] = user_credentials['email']
23         register['password'] = encrypt_password(user_credentials['password'])
24         register['number'] = user_credentials['number']
25
26     user.add_user(register)

```

Gambar 3.29 Source Code Modul registrasi

3. Modul pengambilan data grafik

Pada sistem ini terdapat dua buah tipe grafik yaitu grafik berdasarkan seluruh sub-kategori (Grafik Utama) dan grafik berdasarkan sub kategori yang dipilih dengan menampilkan data dari Shopee, Blibli dan Tokopedia (Grafik Sub-kategori).

Ketika sistem akan menampilkan grafik utama, sistem membuka koneksi database grafik (baris kode ke-18), setelah koneksi terbuka sistem akan melakukan *query* kedalam database dengan parameter penyaringan berdasarkan tahun (baris kode ke-19) setelah *query* berhasil dilakukan, sistem akan mengkonversi tipe data hasil *query* menjadi tipe data *list*. Source Code grafik utama dapat dilihat pada Gambar 3.30.

```

18     database = Graph()
19     data = database.get_graph_main_by_year(year)
20     result = [graph for graph in data]

```

Gambar 3.30 Source Code Grafik Utama

Grafik kedua yaitu grafik sub-kategori sistem menampilkan grafik tersebut dilakukan dengan cara yang sama dengan grafik utama hanya saja parameter *query* yang digunakan berdasarkan sub-kategori yang dipilih (baris kode ke-30). Source Code dapat dilihat pada Gambar 3.31

```

29     database = Graph()
30     data = database.get_graph_by_category(category)
31     result = [graph for graph in data]

```

Gambar 3.31 Source Code Grafik Sub-kategori

4. Modul enkripsi kata sandi

Pada saat pendaftaran akun pengguna sistem melakukan enkripsi kata sandi agar supaya data yang disimpan dapat terjaga kerahasiaannya. Enkripsi kata sandi dilakukan dengan pustaka *Bcrypt* yang dimiliki oleh bahasa pemrograman *Python*. Modul ini bekerja dengan menerima parameter kata sandi yang kemudian diolah oleh pustaka *Bcrypt* yang menghasilkan kata sandi yang sudah terenkripsi. *Source Code* dapat dilihat pada Gambar 3.32

```

4 def encrypt_password(password):
5     encrypted = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
6     return encrypted
7

```

Gambar 3.32 Modul Enkripsi

5. Modul pengiriman email

Sistem *cpart* memiliki fitur lupa password yang bekerja dengan cara mengirimkan *URL* khusus ke email pengguna yang terdaftar. *Source Code* dapat dilihat pada Gambar 3.33.

```

1 import os
2 import smtplib
3 import ssl
4
5 SMTP_SERVER = "smtp.gmail.com"
6 PORT = 587 # For starttls
7 SENDER_EMAIL = os.getenv('EMAIL_ADDRESS')
8 PASSWORD = os.getenv('EMAIL_PASSWORD')
9
10
11 def send_to_user(receiver_email, secret_id):
12     def compose_url():
13         return f'https://cparts.netlify.app/user/forget-password/{secret_id}'
14         # return f'http://localhost:5000/user/forget?key={secret_id}'
15
16     def compose_message(url):
17         receiver_email = "your@gmail.com"
18         message = f"""From: From Person {SENDER_EMAIL}
19 To: To Person {receiver_email}
20 MIME-Version: 1.0
21 Content-type: text/html
22 Subject: Forgot Password CPARTS
23
24 {url}
25
26 <b>Please Visit This <a href="{url}">Link</a> to reset password</b>
27 """
28     return message
29
30 # Create a secure SSL context
31 context = ssl.create_default_context()
32
33 url = compose_url()
34 message = compose_message(url)
35
36 # Try to log in to server and send email
37 try:
38     server = smtplib.SMTP(SMTP_SERVER, PORT)
39     server.hello() # Can be omitted
40     server.starttls(context=context) # Secure the connection
41     server.hello() # Can be omitted
42     server.login(SENDER_EMAIL, PASSWORD)
43     # TODO: Send email here
44     server.sendmail(SENDER_EMAIL, receiver_email, message)

```

Gambar 3.33 *Source Code* Kirim Email

Pengiriman email dilakukan dengan memanfaatkan pustaka yang tersedia dalam *Python* yaitu dengan pustaka *smtplib* dan *ssl*, modul *smtplib* berguna untuk melakukan koneksi terhadap server email Google sedangkan modul *SSL* berfungsi untuk membuat koneksi ke server menjadi lebih aman, modul pengiriman email ke pengguna memiliki parameter email pengguna dan *ID* rahasia pengguna yang digunakan untuk membuat link reset kata sandi yang akan digunakan oleh pengguna. (baris kode ke-12 dan 13), modul ini juga dapat membuat pesan yang akan dikirimkan kepada pengguna (baris kode ke-16 hingga baris kode ke-28). Sebelum sistem mampu mengirim email kepada pengguna sistem perlu melakukan *login* kedalam server dengan akun Google dan membuat koneksi dengan *SSL*, setelah sistem berhasil terkoneksi maka sistem dapat mengirim email kepada pengguna (baris kode ke-31 hingga baris kode ke-44)

3.2.8 Pengujian dan Evaluasi

Tahap ini merupakan tahap pengujian prototype yang telah dibuat, dengan memberikan aplikasi untuk diuji kepada pengguna apakah fitur secara fungsionalitas telah sesuai, apakah dari sisi tampilan antar dan pengalaman pengguna sudah baik, apakah diperlukannya fitur tambahan dan fitur yang ada sudah berjalan dengan baik. Pada tahap ini metode pengujian menggunakan metode *blackbox*, pengguna tidak perlu mengetahui bagaimana logika sistem bekerja namun hanya melihat bagaimana secara fungsi sudah berjalan dengan baik atau belum (Amalia et al., 2021).

3.2.9 Memperbaiki Prototype

Setelah melalui tahap pengujian dan evaluasi, pada tahap ini aplikasi yang dibuat disempurnakan lagi sesuai dengan evaluasi dari pengguna. Fitur yang ada bisa saja berkurang maupun bertambah bergantung dengan kebutuhan dari pengguna. Selain itu peningkatan dari tampilan antar muka dan pengalaman pengguna ditingkatkan lagi demi memenuhi kepuasan pengguna dan mudah digunakan.

3.2.10 Implementasi dan Pemeliharaan

Pada tahap ini aplikasi telah disempurnakan dengan menjadikan prototype sebuah produk yang siap untuk dipublikasi, perangkat lunak mulai dibuka untuk umum dan siap digunakan. Selain itu proses pemeliharaan terus berjalan, agar perangkat lunak dapat berjalan dengan baik.