

## BAB 4

### HASIL DAN PEMBAHASAN

#### 4.1 Implementasi Aplikasi

##### 4.1.1 Spesifikasi Sistem

###### 4.1.1.1 Spesifikasi *Development*

Pengembangan aplikasi *website* ini menggunakan Visual Studio Code sebagai IDE (*Integrated Development Environment*). Berikut spesifikasi minimal yang dibutuhkan untuk *development* aplikasi ini.

Tabel 4.1. Spesifikasi Perangkat untuk Menjalankan Visual Studio Code (Visual Studio Code, 2023)

Spesifikasi	<i>Supported Specs</i>	<i>Actual Device</i>
<i>Operating System</i>	<ul style="list-style-type: none"><li>- Windows 8.0, 8.1 and 10, 11 (32-bit dan 64-bit)</li><li>- OS X High Sierra (10.13+)</li><li>- Linux (Debian): Ubuntu Desktop 16.04, Debian 9</li><li>- Linux (Red Hat): Red Hat Enterprise Linux 7, CentOS 7, Fedora 34</li></ul>	Windows 10 (64-bit)
RAM	1GB atau lebih	8GB

<i>Processor</i>	1.6GHz atau lebih	Intel(R) Core(TM) i7-8550U CPU @ 1,80GHz, 1992 Mhz, 4 Core(5), 8 Logical Processor(s)
------------------	-------------------	---

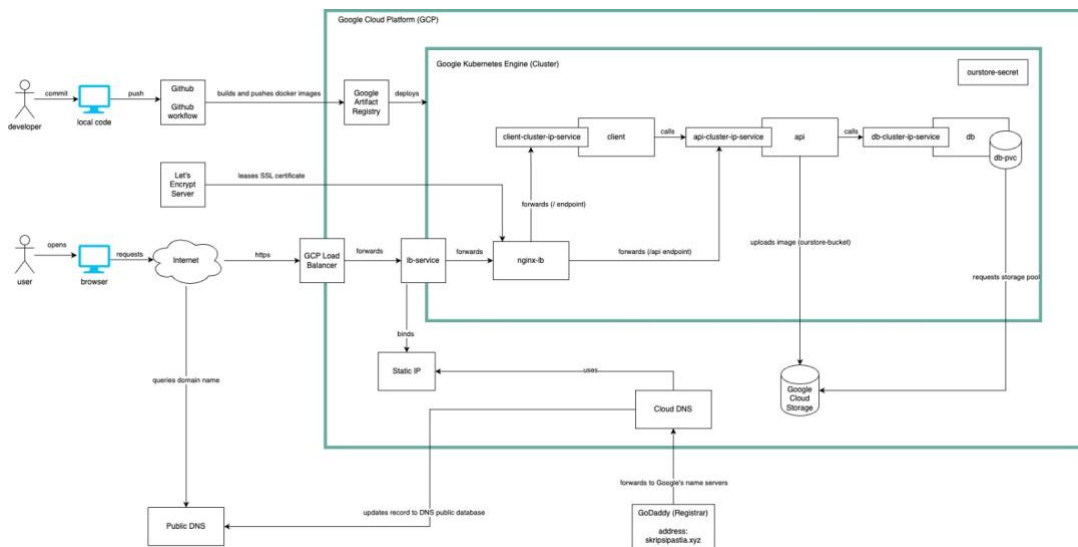
#### 4.1.1.2 Spesifikasi Pengguna

Berikut spesifikasi minimal yang dibutuhkan pengguna untuk menjalankan aplikasi ini.

Tabel 4.2. Spesifikasi Perangkat untuk Menjalankan Aplikasi

<b>Spesifikasi</b>	<b><i>Supported Specs</i></b>
<i>Operating System</i>	Windows 7 atau lebih macOS 10.9 iOS 9 Android 4.4
RAM	128MB atau lebih
<i>Processor</i>	1.6GHz atau lebih
<i>Screen Resolution</i>	1024 x 768
<i>Browser</i>	<ul style="list-style-type: none"> <li>- Internet Explorer 11</li> <li>- Firefox 60 atau lebih</li> <li>- Firefox ESR</li> <li>- Chrome 60 atau lebih</li> <li>- Edge 16 atau lebih</li> <li>- Safari 12 atau lebih</li> <li>- Android 2.1 atau lebih</li> </ul>

## 4.2 Arsitektur Aplikasi



Gambar 4.1. Arsitektur Aplikasi

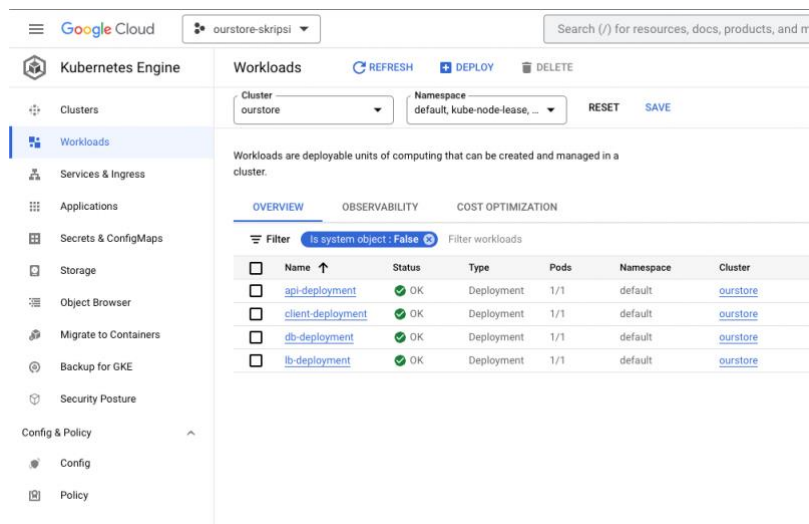
Komponen yang terdapat dan berkontribusi di dalam arsitektur aplikasi ini adalah sebagai berikut.

- Github & Github workflow → sebagai SCM dan *worker* CICD pipeline
- Google Artifact Registry → sebagai *repository* Docker image
- Let's Encrypt Server → *third party server* untuk memberikan *certificate*
- GCP Load Balancer → LB dari GCP untuk menerima *load* dari luar
- Static IP → *service* GCP untuk *assign static IP*
- Cloud DNS → *service* GCP untuk melakukan konfigurasi DNS
- Public DNS → *third party DNS server* yang diakses oleh semua orang
- GoDaddy → sebagai *domain name registrar*
- GCS → *service storage* GCP untuk menyimpan *binary large object*
- GKE → *service* GCP yang menjalankan Kubernetes cluster
- Lb-service → *service* pada GKE untuk menerima *load* dari luar cluster
- Nginx-lb → *deployment* NGINX sebagai *load balancer*
- Client → *deployment frontend* (ReactJS)
- Api → *deployment backend* (NodeJS & ExpressJS)
- Db → *deployment database* (MongoDB)
- Db-pvc → PVC pada GKE untuk *persist data* MongoDB
- Ourstore-secret → *object* untuk menyimpan konfigurasi *confidential*
- Client-cluster-ip-service → *service* untuk *expose deployment* "client"
- Api-cluster-ip-service → *service* untuk *expose deployment* "api"
- Db-cluster-ip-service → *service* untuk *expose deployment* "db"

#### 4.2.1 Google Container Platform (GCP)

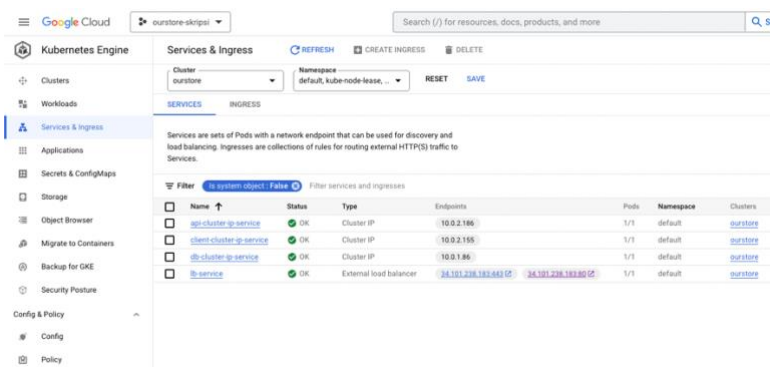
Vendor *cloud computing* yang digunakan di aplikasi ini adalah Google Cloud Platform (GCP). Di dalam GCP, ada berbagai macam *service* yang dapat digunakan mulai dari Kubernetes *engine*, *networking*, *artifact registry* dan lain-lain. *Service* GCP yang digunakan di proyek ini telah dijabarkan pada bab 2.18. Beberapa objek atau konfigurasi yang ada dalam Google Kubernetes Engine (GKE) adalah sebagai berikut.

- db-deployment → *deployment database* MongoDB
- api-deployment → *deployment* NodeJS *web server*
- client-deployment → *deployment* ReactJS *frontend*
- lb-deployment → *deployment* NGINX *load balancer*



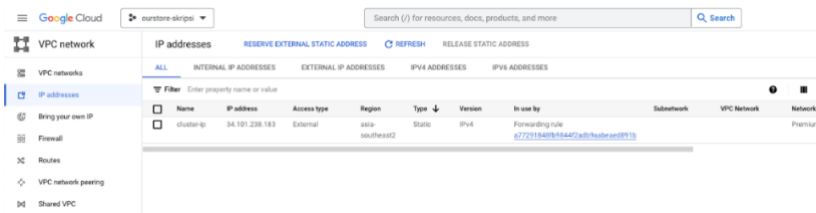
Gambar 4.2. *Deployment Objects* pada GKE

- db-cluster-ip-service → untuk *expose* db-deployment
- api-cluster-ip-service → untuk *expose* api-deployment
- client-cluster-ip-service → untuk *expose* client-deployment



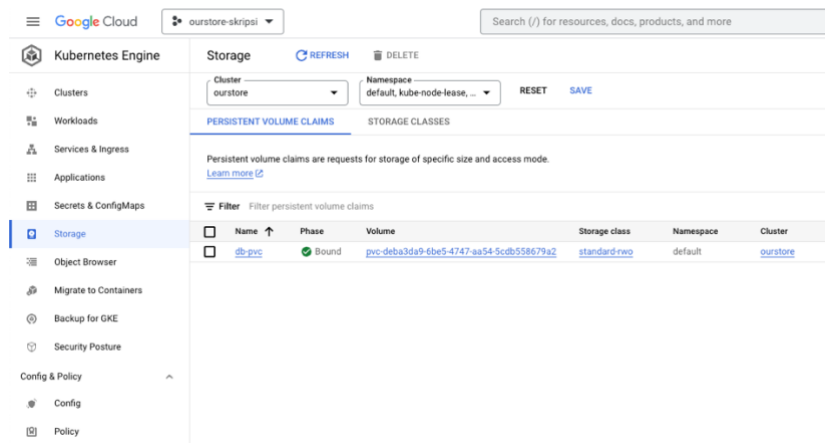
Gambar 4.3. *Service Objects* pada GKE

- lb-service → untuk expose lb-deployment ke luar cluster



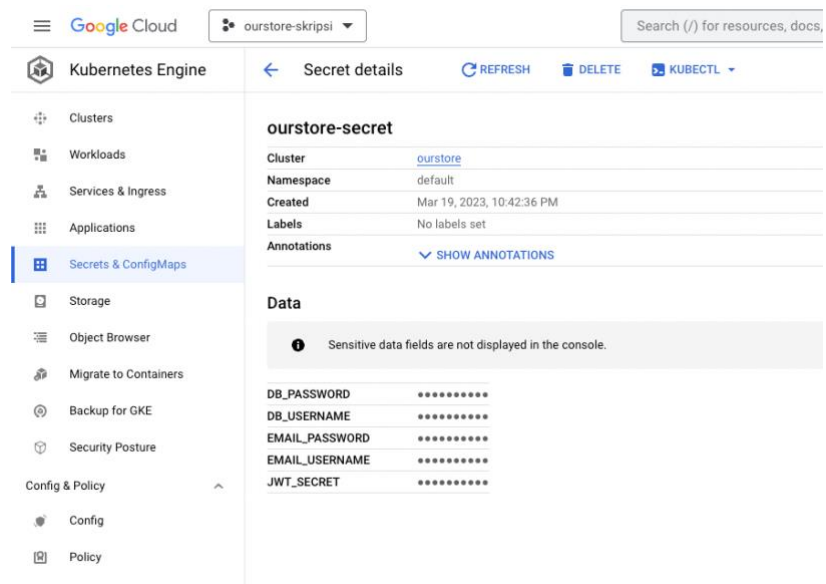
Gambar 4.4. Static IP pada GCP Console

- db-pvc → untuk *persist* data pada MongoDB



Gambar 4.5. PVC pada GKE

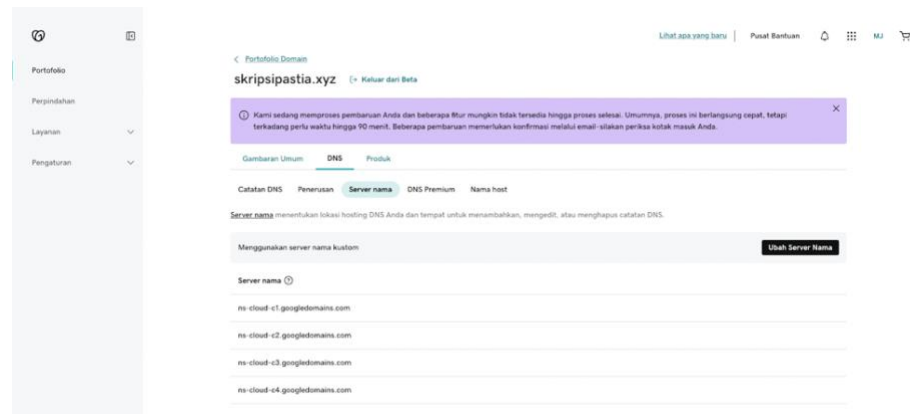
- ourstore-secret → untuk menyimpan konfigurasi *environment variable* yang konfidensial.



Gambar 4.6. Secret pada GKE

#### 4.2.2 Registrar

Aplikasi *web* ini menggunakan nama *domain* skripsipastia.xyz yang didaftarkan oleh *registrar* GoDaddy. Di dalam *registrar*, dilakukan konfigurasi *name server* milik Google sehingga GoDaddy dapat meneruskan *request user* terhadap nama *domain* kepada DNS milik Google



Gambar 4.7. Dashboard Domain Registrar GoDaddy

#### 4.2.3 Github

Github digunakan sebagai *code repository* dan juga *CICD pipeline* yang dijelaskan pada bab III.

#### 4.2.4 SSL Certificate

Nama *domain* perlu mendapatkan *SSL certificate* agar dapat menggunakan protokol HTTPS. Salah satu *certificate authority* yang gratis adalah Let's Encrypt. Protokol yang digunakan adalah *ACME protocol*. Cara kerjanya adalah Let's Encrypt mengirimkan *challenge* kepada *web server* dengan URL sesuai dengan nama *domain* yang ingin di-sign *certificate* nya untuk validasi *domain*, kemudian *web server* akan menjawab *challenge* tersebut melalui *certbot agent*. Setelah *challenge* sesuai, maka Let's Encrypt akan membuat *SSL certificate* untuk nama *domain* yang diminta.

#### 4.2.5 Flow Arsitektur

Arsitektur yang disediakan secara garis besar memiliki dua *flow* yaitu sebagai berikut.

##### 4.2.5.1. Flow Deployment

*Flow* ini berjalan saat *developer* melakukan perubahan pada aplikasi *web*. Dimulai pada saat *developer* melakukan *commit* pada *branch* “main” di Github. Untuk *flow* lebih detailnya dijelaskan pada bab III. Hasil dari *flow* ini yaitu *object deployment* di dalam GKE diubah dan *Pods* dilakukan *restart*. *Flow* lain yaitu saat *web server* menerima *SSL certificate* dari Let’s Encrypt yang dijelaskan pada subbab 4.2.4.

##### 4.2.5.2. Flow User

*Flow* ini berjalan saat *user* (*admin* dan *customer*) memakai aplikasi *web e-commerce*. *Flow* dimulai saat *user* memasukkan nama *domain* skripsipastia.xyz pada *browser*. *Browser* akan melakukan *resolve* nama *domain* ke sebuah *IP address* melalui *local cache* terlebih dahulu. Apabila masih belum dapat *resolve*, *browser* akan berkomunikasi dengan *public DNS server* untuk mendapatkan *IP address* tersebut. *Flow public DNS server* dijelaskan di bab 4.2.2. Ketika *IP address* sudah *resolved*, maka *browser* akan melakukan *request* pada *IP address* tersebut dan diterima oleh GCP Load Balancer yang diteruskan ke Kubernetes *cluster* melalui *lb-service*. *Service lb-service* ini meneruskan *request* kepada *web server / load balancer NGINX* (*nginx-lb*). *Web server* ini meneruskan *request* kepada *deployment “client”* atau “api” sesuai dengan *path* nya (“/” ke “client” dan “/api” ke “api”) melalui *Cluster IP service* masing-masing *deployment*. Ketika *user* melakukan *query/update* pada *database* atau *backend* (*api*) melakukan validasi *user*, *deployment “api”* juga melakukan komunikasi dengan *database* (*db*) melalui *ClusterIP service*. *Database* ini menggunakan *PersistentVolumeClaim* (*PVC*) yang disediakan oleh Google Cloud Storage agar data tidak hilang saat dilakukan *restart* pada

*pods database*. Terakhir, “api” juga melakukan *fetch* dan *put* ke dalam *bucket* pada Google Cloud Storage apabila *browser* ingin menampilkan gambar-gambar atau saat *admin* melakukan *upload* gambar-gambar.

#### 4.2.6 Dependencies

##### A. Backend

- @google-cloud/storage → SDK GCP untuk *upload* gambar
- Bcryptjs → untuk melakukan *hashing* pada *password* yang disimpan di *database*
- Cors → untuk *setup Cors Origin Resource Sharing* (CORS)
- Dotenv → untuk *setup environment variable* dari file *.env*
- Express → sebagai *web server framework*
- Express-mongo-sanitize → *library* untuk melakukan sanitasi data yang masuk ke *database* agar lebih aman
- Express-rate-limit → *library* untuk melakukan *rate limiting* pada *web server*
- Helmet → *library* untuk menambah pengamanan pada *HTTP header*
- Hpp → *library* untuk menjaga untuk mencegah serangan *HTTP Parameter Pollution* (HPP)
- Jsonwebtoken → *library* untuk membuat dan memvalidasi *JWT token*
- Mongoose → sebagai *driver* antara NodeJS dengan MongoDB *database*
- Morgan → *Express middleware* untuk melakukan *logging*
- Multer → *library* untuk melakukan pemrosesan gambar yang akan di-*upload*
- Nodemailer → *library* untuk mengirimkan *email* melalui protokol SMTP
- Nodemon → *dependency* untuk *development* agar *runtime* NodeJS melakukan *restart* setiap kali *developer* melakukan *save*
- Sharp → *library* untuk melakukan pemrosesan gambar yang akan di-*upload*



- Stripe → *library* untuk melakukan pembayaran dengan melakukan koneksi pada *Stripe payment gateway*
- Validator → *library* untuk melakukan validasi terhadap data yang akan disimpan dalam MongoDB
- Xss-clean → *library* untuk menjaga untuk mencegah serangan *Cross Site Scripting (XSS)*

#### B. Frontend

- Font awesome → *library* untuk mendapatkan *icon*
- Stripe → *library* untuk menghubungkan dengan Stripe
- Axios → *library HTTP client* untuk melakukan *request* dan *response* terhadap API
- React-bootstrap → *library bootstrap* untuk React
- React-router-dom → *library* yang menyediakan *routing* untuk aplikasi
- React-redux → *library* yang digunakan untuk mengelola *state*
- React-color → *library* yang menyediakan komponen untuk memilih warna
- React-toastify → *library* yang digunakan untuk menampilkan notifikasi
- React-dom → *library* yang digunakan untuk menghubungkan antara React dan *Document Object Model (DOM)*
- Dotenv → *library* untuk mengatur *variable environment* file *.env*
- Js-Cookie → *library* untuk mengelola *cookie*
- Query-string → *library* untuk mengelola *parameter URL*

#### 4.2.7 Third Party Services

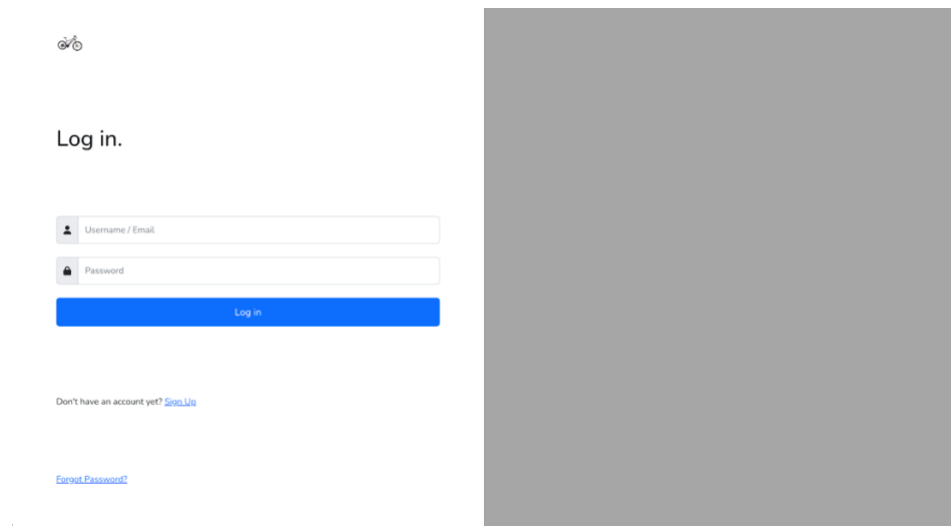
*Third party service* yang digunakan di aplikasi ini adalah Mailtrap dan Stripe. Mailtrap digunakan sebagai *inbox* saat aplikasi mengirimkan *email* kepada *end user*. Stripe digunakan sebagai *payment gateway service* yang mudah untuk diimplementasikan.

### 4.3 Prosedur Penggunaan Aplikasi

Saat pertama kali membuka *web page* “https://skripsipastia.xyz”, maka *browser* akan melakukan *redirect* ke halaman “/login”.

#### a. Halaman *Login*

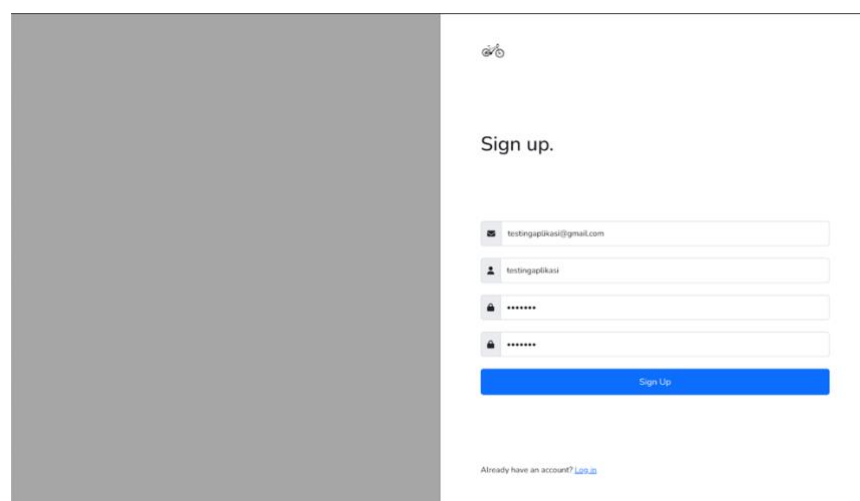
Di halaman ini terdapat *logo* di kiri atas, *form* yang berisi *field username* dan *password*, kemudian *link* untuk melakukan *sign up* dan *forgot password*.



Gambar 4.8. Halaman *Login*

#### b. Halaman *Sign Up*

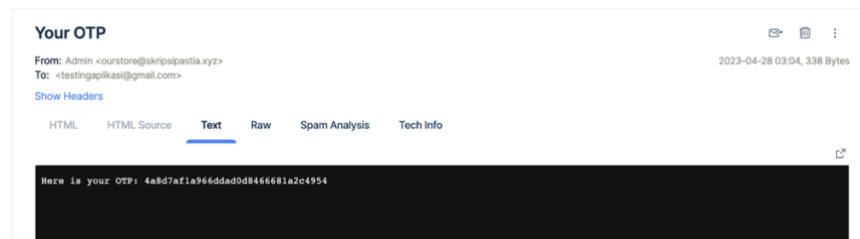
Apabila *user* menuju ke halaman “/signup”, *user* akan mengisi *form* yang berisi *field email*, *username*, *password*, dan *confirm password*. Di halaman ini juga terdapat *link* untuk menuju ke halaman “/login”.



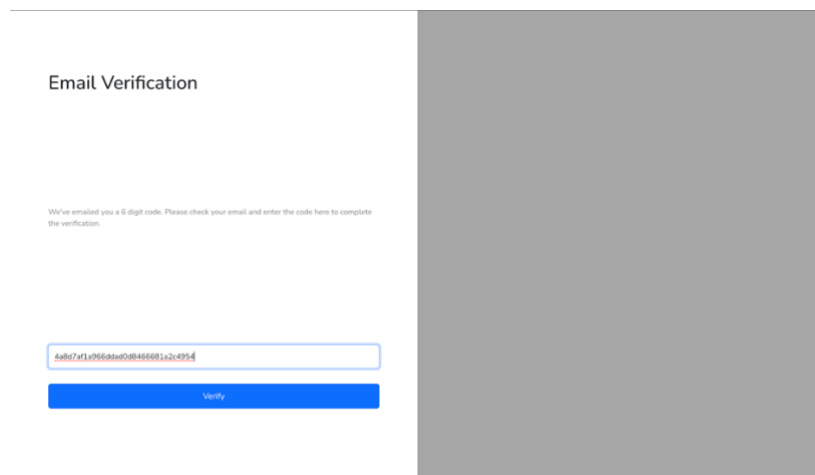
Gambar 4.9. Halaman *Sign Up*

### c. Halaman OTP

Setelah melakukan klik pada tombol “sign up” di halaman “/signup”, *user* akan diarahkan menuju halaman “/otp”. Di halaman ini terdapat *form* yang berisi sebuah *field* OTP yang dikirimkan melalui *email*.



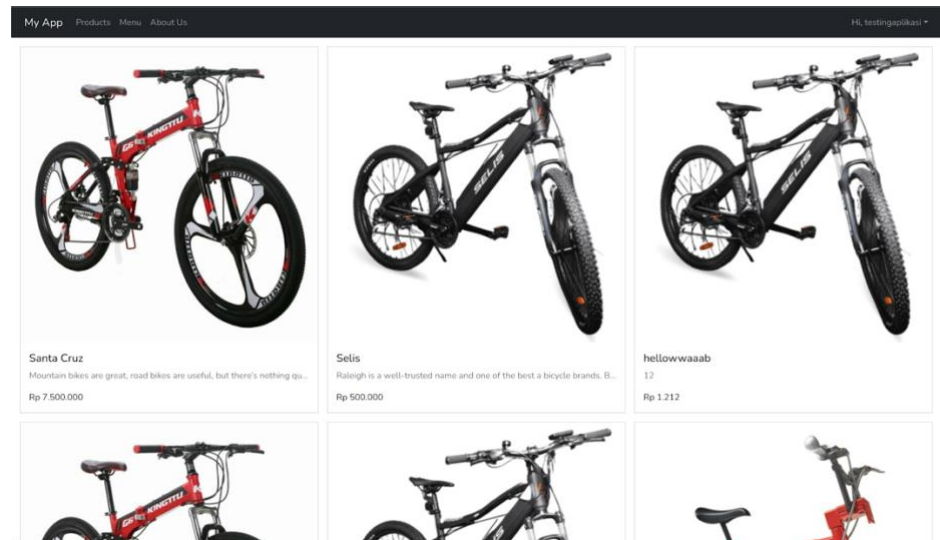
Gambar 4.10. OTP yang Dikirim melalui *Email*



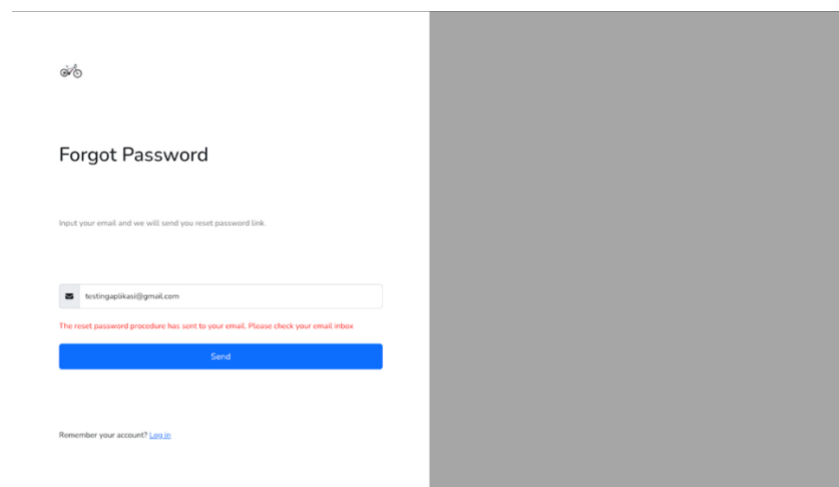
Gambar 4.11. Halaman OTP

### d. Halaman *Products*

Setelah melakukan verifikasi *email*, *user* langsung diarahkan ke halaman “/products”, *user* dapat melihat beberapa produk yang ditawarkan oleh penjual di halaman ini. Selain itu, juga terdapat *header* dan *footer* pada halaman ini.

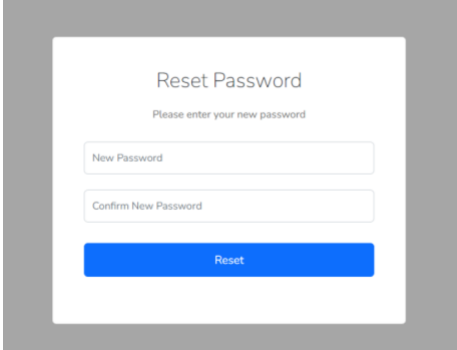
Gambar 4.12. Halaman *Products*e. Halaman *Forgot Password*

Apabila *user* menuju halaman “/forgot” pada halaman “/login”, *user* akan mengisi *email*. Sistem akan mengirimkan *link reset password* melalui *email* *user* tersebut.

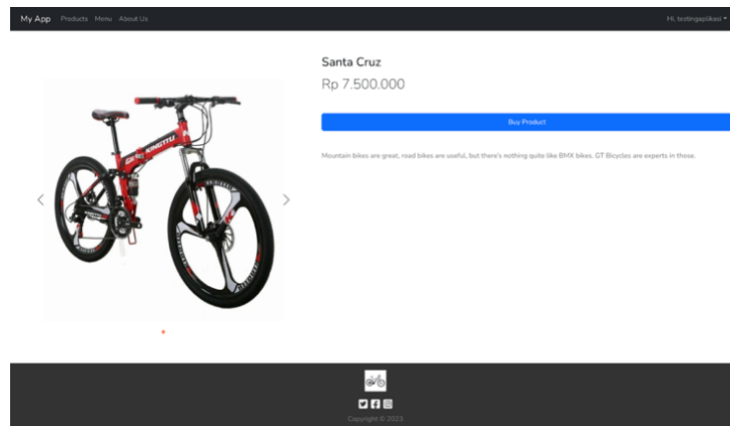
Gambar 4.13. Halaman *Forgot Password*Gambar 4.14. *Reset Token* yang Dikirim melalui *Email*

f. Halaman *Reset Password*

Ketika *user* menuju *link* yang dikirim melalui *email*, *user* akan menuju ke halaman *Reset Password*. *User* mengisi *field password* dan *confirm password*. Setelah itu, *user* akan diarahkan menuju halaman “/login”.

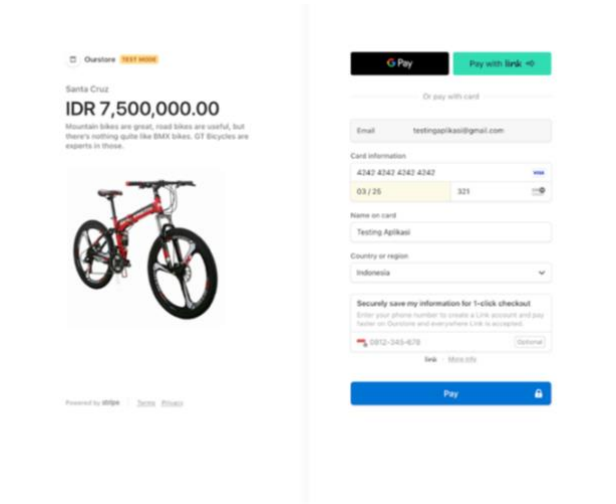

Gambar 4.15. Halaman *Reset Password*g. Halaman *Single Product*

Ketika salah satu *item* pada halaman “/products” diklik, maka *user* akan melihat detail dari produk tersebut dan menuju ke halaman *Single Product*. Di halaman ini terdapat beberapa gambar yang dapat di-*slide*, nama produk, harga, dan tombol “*Buy Product*”.

Gambar 4.16. Halaman *Single Product*h. Halaman *Buy Product* (Stripe)

Saat *user* ingin membeli suatu produk, *user* akan melakukan klik pada tombol “*Buy Product*” pada halaman *Single Product*. Kemudian *user* diarahkan menuju halaman *Checkout Stripe*. Di halaman ini *user* mengisi

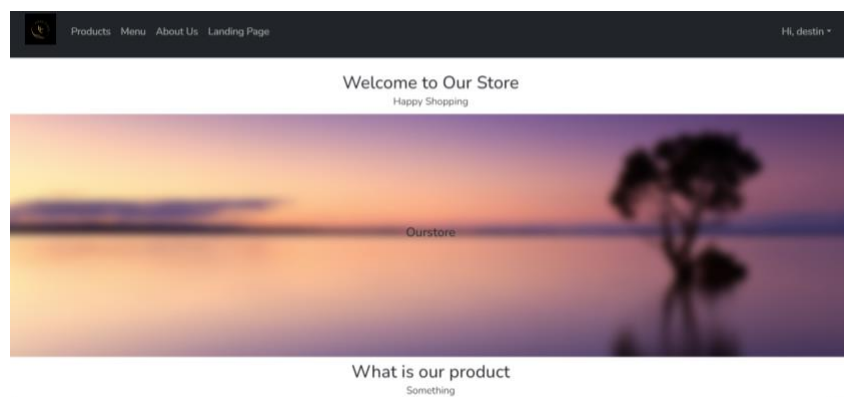
data-data pembayaran seperti nomor kartu, bulan dan tahun *expiry*, CVV, nama, dan *country*. Setelah itu *user* klik tombol “Pay”. Setelah itu muncul pesan sukses dan *user* diarahkan menuju halaman “/products”.



Gambar 4.17. Halaman *Checkout* Stripe

i. Halaman *Custom Page*

Selain melihat produk-produk yang ditawarkan di halaman “/products”, *user* juga dapat melihat halaman lain yang telah disediakan oleh penjual seperti *landing page*, halaman *about us*, halaman *terms and condition*, dan lain-lain. *User* dapat melakukan klik pada salah satu menu pada *header*.

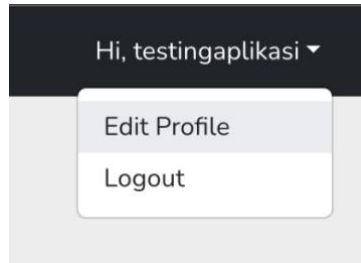


Gambar 4.18. Halaman *Custom Page*

j. Halaman *Edit Profile*

*User* juga dapat melakukan *edit profile* dengan melakukan klik menu pada kanan atas, kemudian pilih menu *Edit Profile*. Di halaman ini terdapat *field*

*username*, *current password*, *new password*, dan *confirm password* untuk melakukan *edit profile* atau *change password*. Setelah itu, user klik tombol “*Submit*”.



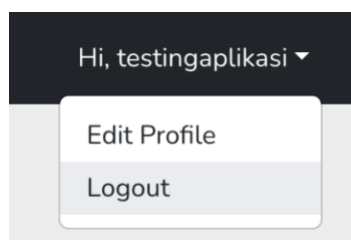
Gambar 4.19. Menu *Edit Profile*

The page has a dark header with "My App" and navigation links "Products", "Menu", and "About Us". On the right of the header is "Hi, testingaplikasi". The main content area is titled "Your Profile" and contains a form with the following fields: "Email" (pre-filled with "testingaplikasi@gmail.com"), "Username" (pre-filled with "testingaplikasi"), a "Change Password" toggle switch (checked), "Current Password", "New Password", and "Confirm New Password". A blue "Submit" button is at the bottom of the form. The footer is dark and contains a logo, social media icons, and the text "Copyright © 2023".

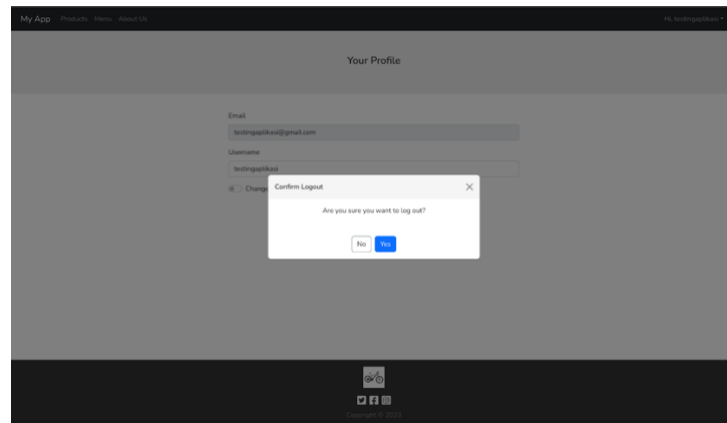
Gambar 4.20. Halaman *Edit Profile*

k. Logout

User juga dapat melakukan *logout* yang terletak pada menu bagian kanan atas. Setelah itu *pop-up* akan muncul dan user klik tombol “*Yes*”.



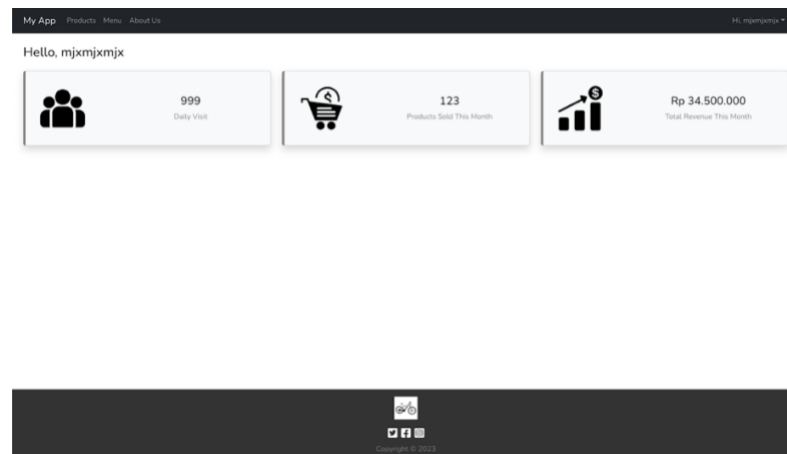
Gambar 4.21. Menu *Logout*



Gambar 4.22. *Logout Confirmation Dialog*

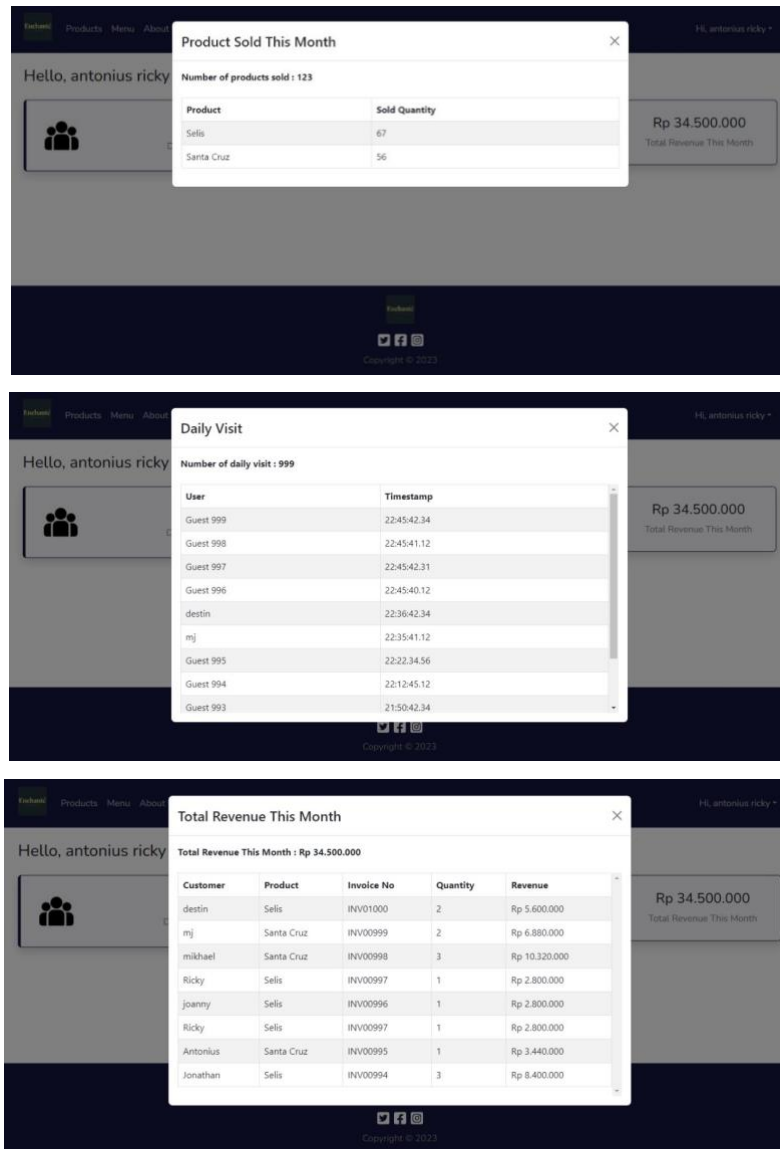
### 1. Halaman *Dashboard*

Aplikasi ini juga dapat digunakan oleh admin. Di mana setelah *login*, admin akan diarahkan menuju halaman “/dashboard”. Di menu *dashboard* ini, admin diharapkan mendapatkan *insight* mengenai performa toko *online* nya, mulai dari *daily active user*, produk yang terjual di bulan ini, dan juga pendapatan total di bulan ini. Masing-masing *metrics* dapat diklik dan ditampilkan tabel detail terkait *metric* tersebut.

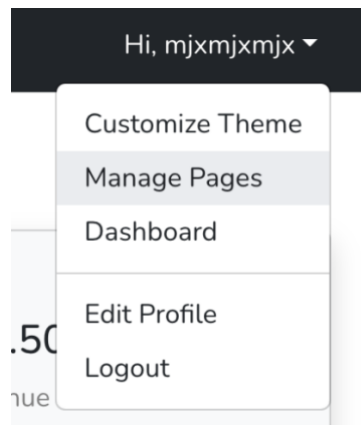
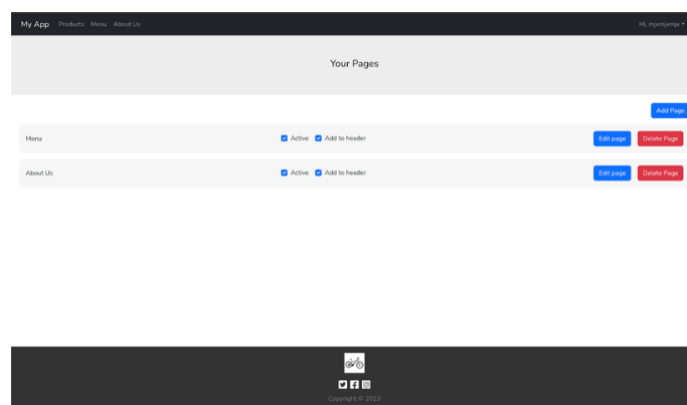


Gambar 4.23. Halaman *Dashboard*

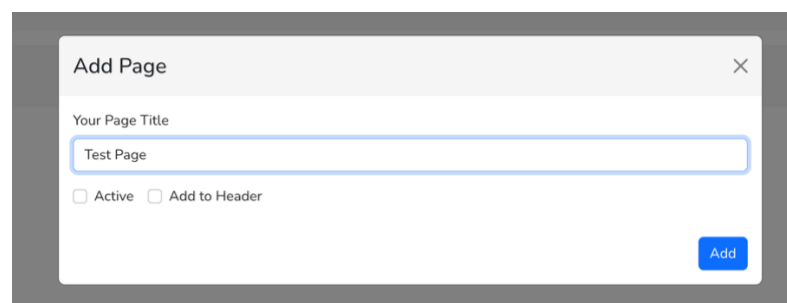


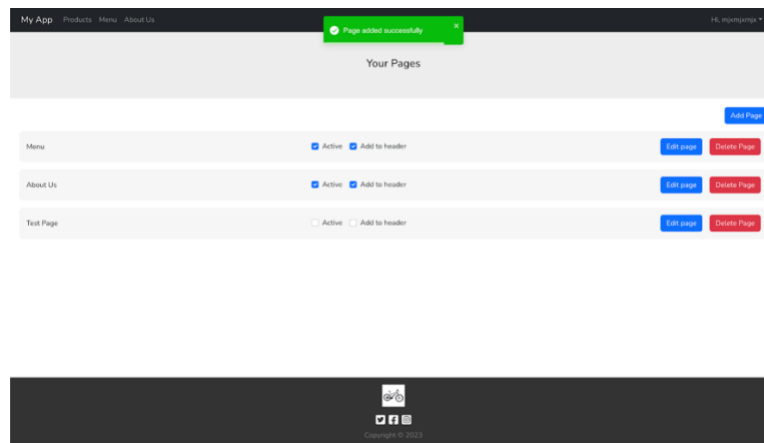
Gambar 4.26. Tabel Detail *Dashboard*m. Halaman *Manage Pages*

Admin juga dapat mengatur halaman-halaman yang ingin ditampilkan pada *customer* maupun calon *customer*. Admin dapat klik pada menu “*Manage Pages*” pada kanan atas.

Gambar 4.24. Menu *Manage Pages*Gambar 4.25. Menu *Manage Pages*

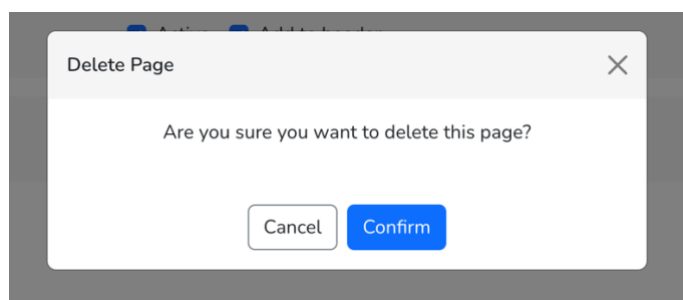
Di halaman ini terdapat daftar halaman-halaman yang dibuat oleh admin. Terdapat tombol “*Add Page*” untuk menambahkan suatu halaman. Ketika admin klik tombol tersebut, sebuah *pop-up* “*Add Page*” akan muncul. Admin kemudian mengisi nama halaman dan mengisi *option* “*Active*” atau “*Add to Header*”. Terakhir, admin klik tombol “*Add*”. Setelah itu pesan sukses akan muncul dan halaman baru ditambahkan di daftar halaman.

Gambar 4.26. *Pop-Up Add Page*

Gambar 4.27. Pesan Sukses *Add Page*

*Option “Active”* untuk memberi “*flag*” apakah halaman sudah dapat diakses oleh *customer* atau masih dalam proses pembuatan oleh admin. Ketika halaman-halaman sudah mulai banyak, maka *header* menjadi tidak cukup. Oleh karena itu, terdapat *option “Add to header”* juga untuk membedakan halaman mana saja yang dapat langsung diakses oleh *customer* pada *header*. Halaman yang tidak ada pada *header* tetap bisa diakses oleh *customer* asalkan *option “Active”* tercentang.

Admin juga dapat menghapus halaman dengan klik tombol “*Delete Page*” pada setiap *item* halaman. Setelah melakukan klik, *pop-up* akan muncul dan klik tombol “*Confirm*”. Terakhir, pesan sukses akan muncul dan halaman tersebut terhapus dari daftar.

Gambar 4.28. *Delete Page Confirmation Dialog*

#### n. Halaman *Custom Page* (Admin)

Admin dapat mengubah isi *page* dengan klik tombol “*Edit page*” pada setiap *page item* di halaman *Manage Pages*. Admin akan diarahkan kepada

*page* tersebut. Satu halaman memiliki beberapa *section* dan beberapa *section* terdiri dari beberapa kolom. Suatu kolom bisa berupa satu dari empat tipe, yaitu *text* untuk menampilkan tulisan, *image* untuk menampilkan gambar, *image overlay* untuk menampilkan tulisan di atas gambar, dan *product* untuk menampilkan salah satu produk yang ditawarkan. Admin dapat klik menu “Add section” di kanan bawah untuk menambahkan *section*. Setelah itu akan muncul *form* “Add Section”.



Gambar 4.29. Halaman *Custom Page (Admin)*

Gambar 4.30. *Form Add Section (Text)*

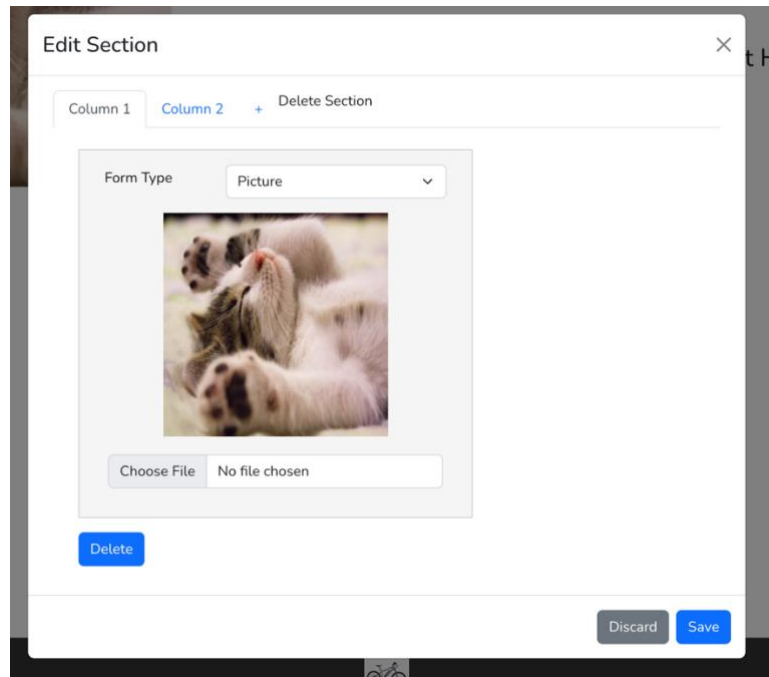
Gambar 4.31. *Form Add Section (Picture Overlay)*

Di *form* ini terdapat beberapa *field* di mana berbeda sesuai dengan *form type* nya. Untuk menambahkan kolom, admin dapat klik tombol “+” di atas *form*. Admin juga dapat menghapus suatu kolom dengan klik tombol “Delete” di kiri bawah”. Apabila *section* sudah oke, admin klik tombol “Add” di kanan bawah.

Untuk melakukan *edit section*, admin hanya perlu klik *section* yang ingin di-*edit*. Setelah itu *form edit section* akan muncul.



Gambar 4.32. Halaman *Custom Page (Filled by Admin)*

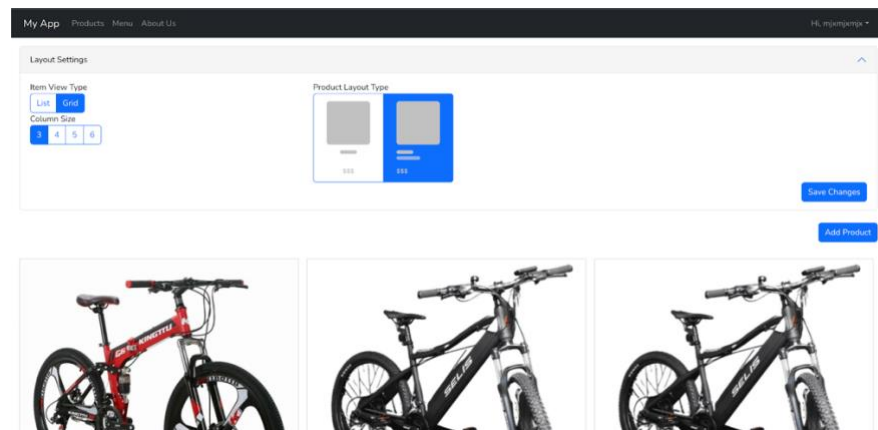


Gambar 4.33. *Form Edit Section*

Admin dapat mengubah *value* yang ada di *form*. Setelah sudah oke, admin dapat klik tombol “Save” di kanan bawah. Admin juga dapat menghapus *section* dengan klik tombol “Delete Section” di kanan atas *form*.

o. Halaman *Products* (Admin)

Tampilan halaman “/products” yang dilihat oleh admin berbeda dengan *customer* pada poin D. Di halaman “/products” dengan admin, terdapat menu tambahan untuk melakukan kustomisasi *layout* halaman “/products”. Admin dapat memilih *item view type* apakah mau berbentuk *list* atau *grid*. Apabila *grid*, admin juga dapat memilih berapa banyak kolom per satu baris. Selain itu, admin juga dapat memilih tipe *layout* yang telah disediakan sistem. Terakhir, admin klik tombol “save changes”.

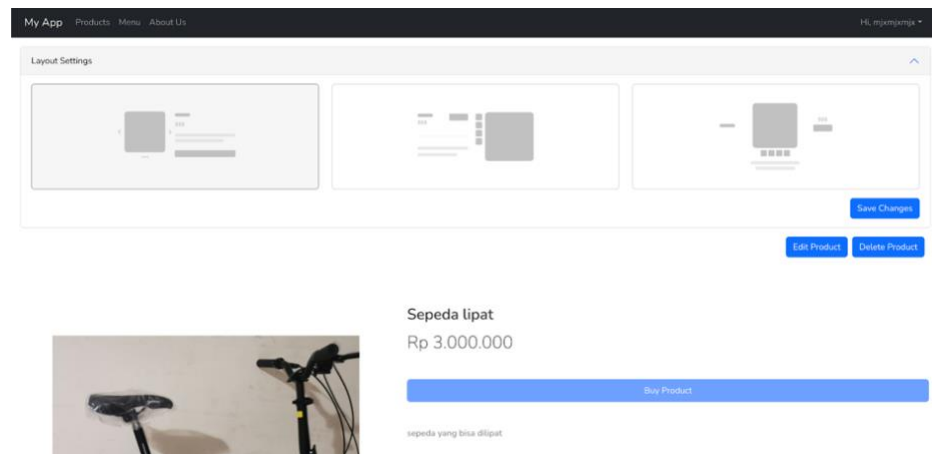
Gambar 4.34. Halaman *Products* (Admin)

Untuk menambah produk, admin dapat klik tombol “*Add Product*” dan form “*Add New Product*” akan muncul. Form ini memiliki *field* nama produk, harga produk (dalam rupiah), deskripsi produk, dan juga *field* untuk menambah gambar. Apabila sudah oke, admin klik tombol “*Add*” di kanan bawah form. Setelah itu pesan sukses akan muncul.

Gambar 4.35. Form *Add Product*

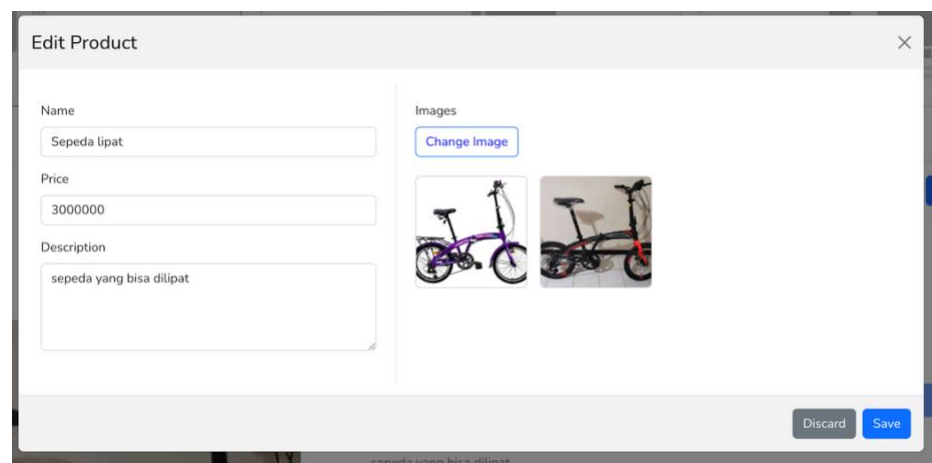
p. Halaman *Single Product* (Admin)

Halaman ini juga memiliki tampilan yang sedikit berbeda antara admin dan *customer*. Untuk admin, tombol “*Buy Product*” disabled, dan ada beberapa menu tambahan. Terdapat menu *layout setting* untuk melakukan kustomisasi *layout* di halaman *Single Product*. Admin dapat memilih *layout setting* yang disediakan oleh sistem, setelah itu admin dapat klik tombol “*Save Changes*” dan pesan sukses akan muncul.



Gambar 4.36. Halaman *Single Product (Admin)*

Admin juga dapat mengubah *product* dengan klik tombol “*Edit Product*”, setelah itu *form “Edit Product”* akan muncul. Setelah itu admin dapat klik tombol “*Save*” saat selesai mengubah *product*.



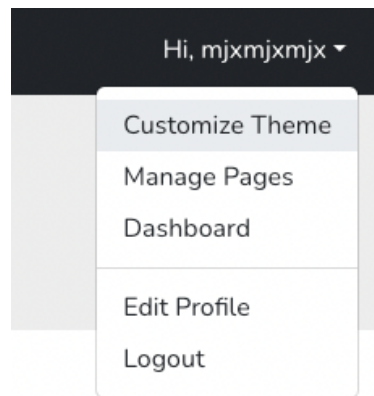
Gambar 4.37. *Form Edit Product*

Untuk menghapus *product*, admin dapat klik tombol “*Delete Product*” dan form konfirmasi akan muncul. Klik tombol “*Confirm*” untuk melanjutkan. Setelah itu, pesan sukses akan muncul dan admin diarahkan menuju halaman “/products”.

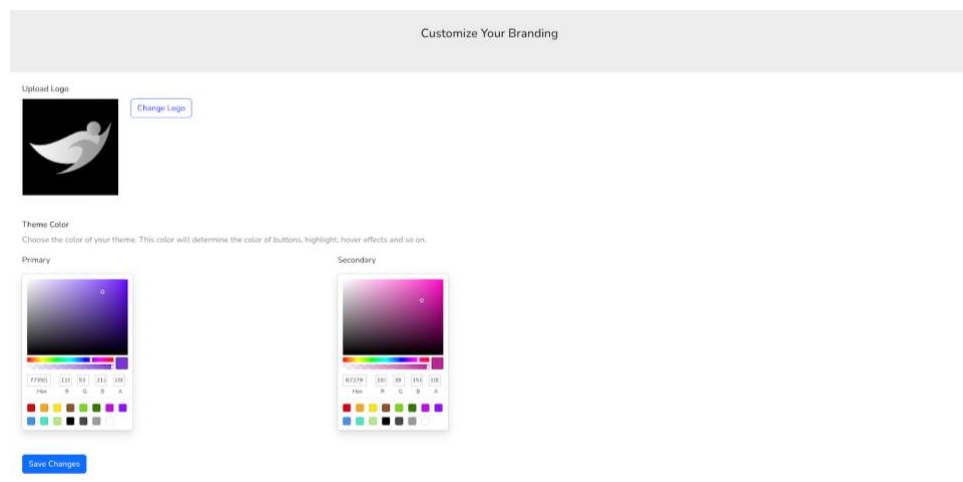
q. Halaman *Customize Theme*

Halaman terakhir yaitu halaman *Customize Theme*. Admin dapat klik tombol “*Customize Theme*” pada menu di kanan atas.



Gambar 4.38. Menu *Customize Theme*

Halaman ini dapat dipakai oleh admin untuk mengubah tema atau warna dari tampilan *web*. Terdapat dua warna yaitu *primary* dan *secondary*, disarankan agar kedua warna ini saling berkomplemen. Admin dapat memilih warna melalui *color picker* yang sudah disediakan. Selain itu, admin juga dapat mengubah *logo* dengan klik pada tombol "*Change Logo*". Setelah semua sudah oke, admin dapat klik tombol "*Save Changes*" untuk menyimpan perubahan tema. Pesan sukses akan segera muncul.

Gambar 4.39. Halaman *Customize Theme*

## 4.4 Testing

### 4.4.1 Unit Testing

Berikut hasil *unit testing* (*white-box testing*) yang telah dilakukan dengan menggunakan *framework* Jest .

Tabel 4.3. Hasil *Unit Testing*

Class	Function	Evaluation	Test
ThemeController	getTheme	Should have a getTheme function	Passed
		Should call themeModel.find	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
	addTheme	Should have an addTheme function	Passed
		Should call themeModel.save	Passed
		Should return 201 response code	Passed
		Should return JSON body in response	Passed
	editTheme	Should have an editTheme function	Passed
		Should call themeModel.find	Passed
		Should call themeModel.save	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
MetricController	getMetric	Should have a getMetric function	Passed
		Should call metricModel.find	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
	addMetric	Should have an addMetric function	Passed
		Should call metricModel.save	Passed
		Should return 201 response code	Passed
		Should return JSON body in response	Passed
	editMetric	Should have an editMetric function	Passed

		Should call metricModel.find	Passed
		Should call metricModel.findByIdAndUpdate	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
UserController	getUser	Should have a getUser function	Passed
		Should call userModel.findById	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
	editUser	Should have an editUser function	Passed
		Should call userModel.findByIdAndUpdate	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
AuthController	login	Should have a login function	Passed
		Should call userModel.findOne	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
	signup	Should have a signup function	Passed
		Should call userModel.save	Passed
		Should return 201 response code	Passed
		Should return JSON body in response	Passed
PageController	getAllPages	Should have a getAllPage function	Passed
		Should call pageModel.find	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
	addPage	Should have an addPage function	Passed
		Should call pageModel.save	Passed
		Should return 201 response code	Passed
		Should return JSON body in response	Passed
	editPage	Should have an editPage function	Passed

		Should call pageModel.findById	Passed
		Should call pageModel.save	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
	deletePage	Should have a deletePage function	Passed
		Should call pageModel.findByIdAndDelete	Passed
		Should return 204 response code	Passed
ProductController	getAllProducts	Should have a getAllProducts function	Passed
		Should call productModel.find	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
	addProduct	Should have an addProduct function	Passed
		Should call productModel.save	Passed
		Should return 201 response code	Passed
		Should return JSON body in response	Passed
	editProduct	Should have an editProduct function	Passed
		Should call productModel.findById	Passed
		Should call productModel.save	Passed
		Should return 200 response code	Passed
		Should return JSON body in response	Passed
	deleteProduct	Should have a deleteProduct function	Passed
		Should call productModel.findByIdAndDelete	Passed
		Should return 204 response code	Passed

#### 4.4.2 Integration Testing

Berikut hasil *integration testing (black-box testing)* yang telah dilakukan ke seluruh API dengan menggunakan Postman.

Tabel 4.4. Hasil *Integration Testing*

Document Group	HTTP Method	Endpoint	Required Header	Request Body	Response Code	Response Body	Test
Metric	GET	/api/v1/metrics/	Authorization: Bearer {{jwt}}	-	200	{ "status": "success", "data": [ { "_id": "6436b482b9e8250be928a57a", "productSoldPerMonth": 123, "revenuePerMonth": 34500000, "dailyActiveUser": 123, "__v": 0 }] }	Passed

						] }	
	POST	/api/v1/metrics/	Authorization: Bearer {{jwt}}	{ "productSoldPerMonth": 123, "revenuePerMonth": 34500000, "dailyActiveUser": 123 }	201	{ "status": "success", "data": { "productSoldPerMonth": 123, "revenuePerMonth": 34500000, "dailyActiveUser": 123, "_id": "6436b482b9e8250be928a57a", "_v": 0 } }	Passed
	PATCH	/api/v1/metrics/	Authorization: Bearer {{jwt}}	{ "dailyActiveUser": 999 }	200	{ "status": "success", "data": { "_id": "6436b482b9e8250be928a57a", "productSoldPerMonth": 123, "revenuePerMonth": 34500000, "dailyActiveUser": 999, "_v": 0 } }	Passed
Page	POST	/api/v1/pages/	Authorization: Bearer {{jwt}}	{ "name": "Menu", "isActive": true, "isInHeader": }	201	{ "status": "success", "data": { "name": "Menu", "isActive": true, }	Passed

				<pre> true } </pre>		<pre> "isInHeader": true, "_id": "6436b4f9b9e8250be928a582", "sections": [], "__v": 0 } } </pre>	
	GET	/api/v1/pages/	-	-	200	<pre> {   "status": "success",   "data": [     {       "_id": "6436b5c0b9e8250be928a58e",       "name": "Menu",       "isActive": true,       "isInHeader": true,       "sections": [],       "__v": 0     }   ] } </pre>	Passed
	GET	/api/v1/pages/:id/	-	-	200	<pre> {   "status": "success",   "data": {     "_id": "6436b5c0b9e8250be928a58e",     "name": "Menu",     "isActive": true,     "isInHeader": true,     "sections": [],     "__v": 0   } } </pre>	Passed

						} }	
	PATCH	/api/v1/pages/:id/ /	Authorization: Bearer {{jwt}}	{ "sections": [ { "columns": [ { "columnType": "picture", "imageLink": "https://picsum. photos/200" }, { "columnType": "text", "header": "This is a really good product", "content": "This product is good because it's not bad" }, { "columnType": "picture", "imageLink":	200	{ "status": "success", "data": { "_id": "6436b5c0b9e8250be928a58e", "name": "Menu", "isActive": true, "isInHeader": true, "sections": [ { "columns": [ { "columnType": "picture", "imageLink": "https://picsum.photos/200", "_id": "6436b680b9e8250be928a596" }, { "columnType": "text", "header": "This is a really good product", "content": "This product is good because it's not bad", "_id": "6436b680b9e8250be928a597" }, { "columnType": "picture", "imageLink": "https://picsum.photos/200", "_id": "6436b680b9e8250be928a598" } ] } ] }	Passed



				<pre> "https://picsum. photos/200" } ] }}, { "columns": [ { "columnType": "picture", "imageLink": "https://picsum. photos/200/300" }}, { "columnType": "text", "header": "This is a good product", "content": "This product is good because it's not bad" } ] } </pre>	<pre> } ], "_id": "6436b680b9e8250be928a595" }}, { "columns": [ { "columnType": "picture", "imageLink": "https://picsum.photos/200/300", "_id": "6436b680b9e8250be928a59a" }}, { "columnType": "text", "header": "This is a good product", "content": "This product is good because it's not bad", "_id": "6436b680b9e8250be928a59b" } ], "_id": "6436b680b9e8250be928a599" } ], "__v": 1 } } </pre>	
--	--	--	--	--	--	--

				]			
	DELETE	/api/v1/pages/:id/	Authorization: Bearer {{jwt}}	-	204	-	Passed
	POST	/api/v1/pages/images/	Authorization: Bearer {{jwt}}	Form-Data Key: image Value: [file png]	201	{ "status": "success", "data": "https://storage.googleapis.com/ourstore-bucket/page-1681308015082.jpeg" }	Passed
Products	POST	/api/v1/products/	Authorization: Bearer {{jwt}}	Form-Data  Key: price Value: 1000  Key: name Vlaue: seped  Key: description Value: Sepeda lipat  Kwy: images Value: [file jpeg]  Key: images Value: [file png]	201	{ "status": "success", "data": { "name": "sepeda", "price": 1000, "description": "Sepeda lipat", "images": [ { "imageLink": "https://storage.googleapis.com/ourstore-bucket/product-undefined-1681318152753-1.jpeg", "_id": "6436e108cc759fa1f6a37d05" }, { "imageLink": "https://storage.googleapis.com/ourstore-bucket/product-undefined-1681318152756-	Passed

						<pre> 2.jpeg",   "_id": "6436e108cc759fa1f6a37d06" }, {   "_id": "6436e108cc759fa1f6a37d04",   "__v": 0 } ] } } </pre>	
	GET	/api/v1/products/	-	-	200	<pre> {   "status": "success",   "data": [     {       "_id": "641c237b71c41294efb00576",       "name": "Roadmaster Gen 2",       "price": 150000,       "description": "Here's one of the best bicycle brands for engineering nerds (and we mean that affectionately).",       "images": [         {           "imageLink": "https://storage.googleapis.com/ourstore-bucket/product-undefined-1679565691912-1.jpeg",           "_id": "641c237b71c41294efb00577"         }       ]     }   ],   "__v": 0 } </pre>	Passed

						<pre> }, {   "_id": "641c250771c41294efb007cd",   "name": "Trek Bikes",   truncated... </pre>	
	GET	/api/v1/products/:id/	-	-	200	<pre> {   "status": "success",   "data": {     "_id": "641c237b71c41294efb00576",     "name": "Roadmaster Gen 2",     "price": 150000,     "description": "Here's one of the best bicycle brands for engineering nerds (and we mean that affectionately).",     "images": [       {         "imageLink": "https://storage.googleapis.com/ourstore-bucket/product-undefined-1679565691912-1.jpeg",         "_id": "641c237b71c41294efb00577"       }     ],     "__v": 0   } } </pre>	Passed
	PATCH	/api/v1/products/:id/	Authorization: Bearer {{jwt}}	Form-Data	200	<pre> {   "status": "success", </pre>	Passed

				Key: price Value: 200000		<pre> "data": {   "_id": "641c237b71c41294efb00576",   "name": "Roadmaster Gen 2",   "price": 200000,   "description": "Here's one of the best bicycle brands for engineering nerds (and we mean that affectionately).",   "images": [     {       "imageLink": "https://storage.googleapis.com/ourstore- bucket/product-undefined-1679565691912- 1.jpeg",       "_id": "641c237b71c41294efb00577"     }   ],   "__v": 0 } </pre>	
	DELETE	/api/v1/products /:id/	Authorization: Bearer {{jwt}}	-	204	-	Passed
	GET	/api/v1/products /:id/checkout- session/	-		200	<pre> {   "status": "success",   "session": {     "id": "cs_test_a1sQ2Oc9OWUlVBHKYEDT0Jhpm F63dnwHTwdln0M4Giv3mupXPHZwz1BE Cq", </pre>	Passed

					<p>"object": "checkout.session",  "after_expiration": null,  "allow_promotion_codes": null,  "amount_subtotal": 20000000,  "amount_total": 20000000,  "automatic_tax": {  "enabled": false,  "status": null  },  truncated...</p> <p>"customer_email": "sda fdsaf@gmail.com",  "expires_at": 1681405020,  truncated...</p> <p>"url":  <a href="https://checkout.stripe.com/c/pay/cs_test_a1sQ2Oc9OWUl vBHKYEDT0JhpmF63dnwHTwdIn0M4Giv3mupXPHZwz1BECq#fidkdWxOYHwnPyd1blpxYHZxWjA0SFV0dnNBskJkUIVKVFV3YV1oMWZPajRzUUBdSnU1UkdiTX13XFJ8NkJPdFyVnZKRf1yRk5SUEBXNGhXaG1IQEBwYjJ3VnNTa1dgNIJoT1RdTl9qc3FfNTVJU9BdX9%2FRCCpJ2N3amhWYHdzYHcnP3F3cGApJ2lkfGpwcVF8dWAnPyd2bGtiaWBabHFgaCcpJ2BrZGd">https://checkout.stripe.com/c/pay/cs_test_a1sQ2Oc9OWUl vBHKYEDT0JhpmF63dnwHTwdIn0M4Giv3mupXPHZwz1BECq#fidkdWxOYHwnPyd1blpxYHZxWjA0SFV0dnNBskJkUIVKVFV3YV1oMWZPajRzUUBdSnU1UkdiTX13XFJ8NkJPdFyVnZKRf1yRk5SUEBXNGhXaG1IQEBwYjJ3VnNTa1dgNIJoT1RdTl9qc3FfNTVJU9BdX9%2FRCCpJ2N3amhWYHdzYHcnP3F3cGApJ2lkfGpwcVF8dWAnPyd2bGtiaWBabHFgaCcpJ2BrZGd</a></p>	
--	--	--	--	--	---	--

						pYFVpZGZgbWppYWB3dic%2FcXdwYHgl "" } }	
Themes	POST	/api/v1/themes/	-	{ "logoLink": "https://storage. googleapis.com/ ourstore- bucket/logo- 1681314620112 .jpeg", "primary": "#BC3D3D", "secondary": "#6A4545", "viewType": "grid", "templateId": "1", "columnSize": "3" }	201	{ "status": "success", "data": { "primary": "#BC3D3D", "secondary": "#6A4545", "templateId": "1", "viewType": "grid", "columnSize": 3, "logoLink": "https://storage.googleapis.com/ourstore- bucket/logo-1681314620112.jpeg", "_id": "643ff517f4302c43043e926f", "_v": 0 } }	Passed
	GET	/api/v1/themes/	-	-	200	{ "status": "success", "data": [ { _id": "643ff517f4302c43043e926f",	Passed

						<pre> "primary": "#BC3D3D", "secondary": "#6A4545", "templateId": "1", "viewType": "grid", "columnSize": 3, "logoLink": "https://storage.googleapis.com/ourstore- bucket/logo-1681314620112.jpeg", "__v": 0 } ] } </pre>	
	PATCH	/api/v1/themes/	Authorization: Bearer {{jwt}}	<pre> { "button": "#000000", "primary": "#000000", "secondary": "#122232" } </pre>	200	<pre> { "status": "success", "data": { "_id": "643ff517f4302c43043e926f", "primary": "#000000", "secondary": "#122232", "templateId": "1", "viewType": "grid", "columnSize": 3, "logoLink": "https://storage.googleapis.com/ourstore- bucket/logo-1681314620112.jpeg", "__v": 0 } } </pre>	Passed



	POST	/api/v1/themes/logo/	Authorization: Bearer {{jwt}}	Form-Data Key: image Value: [file png]	201	{ "status": "success", "data": "https://storage.googleapis.com/ourstore-bucket/logo-1681913245497.jpeg" }	Passed
Users	POST	/api/v1/users/	-	{ "email": "mikhaeljonathan15@gmail.com", "username": "mjxxx", "password": "mjx15mjx" }	201	{ "status": "success", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0M2ZmNjQxZjQzMDJjNDMwNDNIOTI3NyIsImhhdCI6MTY4MTkxMzQxNiwiZXhwIjojNjg5Njg5NDE2fQ.DSDFp-ZJEC3-KER40sI24UstTtE6ILS9w1w_Ok1OLq4", "data": { "user": { "email": "mikhaeljonathan15@gmail.com", "username": "mjxxx", "role": "customer", "passwordChangedAt": "2023-04-19T14:03:06.958Z", "verified": false, "isActive": true, "_id": "643ff641f4302c43043e9277", "otp": "a8f8e44146c873825f777cb2247d7fbd", "_v": 0 } }	Passed

						} } }	
	POST	/api/v1/users/login/	-	{ "email": "mikhaeljonathan15@gmail.com", "password": "mjx15mjx" }	200	{ "status": "success", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJpZCI6IjY0M2ZmNjQxZjQzMjNDMDJjNDMw NDNIOTI3NyIsImhhdCI6MTY4MTkxMzQ2 MCwiZXhwIjoxNjg5Njg5NDYwfQ.N8dRYr vteOtir602BWCR- gZNE3jcQnCAkw97FtmIm0k", "data": { "user": { "_id": "643ff641f4302c43043e9277", "email": "mikhaeljonathan15@gmail.com", "username": "mjxxx", "role": "customer", "passwordChangedAt": "2023-04- 19T14:03:06.958Z", "verified": false, "otp": "a8f8e44146c873825f777cb2247d7fbd", "__v": 0 } } }	Passed

	GET	/api/v1/users/	Authorization: Bearer {{jwt}}	-	200	{ "status": "success", "data": [ { "verified": false, "_id": "63d6366aaf12a7d43916603b", "email": "sda fdsaf@gmail.com", "username": "adsfdf", "role": "admin", "passwordChangedAt": "2023-01-28T10:53:40.572Z", "__v": 0 }, { "verified": false, "_id": "63d6752844b42f978a1a1d79", "email": "makan@skripsi.xyz", "username": "makan", "role": "customer", "passwordChangedAt": "2023-01-29T12:56:32.997Z", "__v": 0 }, truncated...	Passed
	GET	/api/v1/users/:id/ /	Authorization: Bearer {{jwt}}	-	200	{ "status": "success", "data": { "verified": false,	Passed

						<pre> {   "_id": "63d6752844b42f978a1a1d79",   "email": "makan@skripsi.xyz",   "username": "makan",   "role": "customer",   "passwordChangedAt": "2023-01-29T12:56:32.997Z",   "__v": 0 } </pre>	
	PATCH	/api/v1/users/:id/	Authorization: Bearer {{jwt}}	{ "username": "admin123" }	200	<pre> {   "status": "success",   "data": {     "verified": false,     "_id": "63d6752844b42f978a1a1d79",     "email": "makan@skripsi.xyz",     "username": "admin123",     "role": "customer",     "passwordChangedAt": "2023-01-29T12:56:32.997Z",     "__v": 0   } } </pre>	Passed
	DELETE	/api/v1/users/:id/	Authorization: Bearer {{jwt}}	-	204	-	Passed
	POST	/api/v1/users/pas/forgot/	-	{ "email": "ricky1@gmail." }	200	<pre> {   "status": "success",   "message": "Token sent to email!" } </pre>	Passed

				com" }			
	PATCH	/api/v1/users/pa ss/reset/:token/	-	{ "password": "supersecret123" }	200	{ "status": "success", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJpZCI6IjYzZGE5ZmU0YWYxYjIwYWE5 YWUzNTM5ZSI6ImhhdCI6MTY4MjU5MD kzOCwiZXhwIjoxNjkwMzY2OTM4fQ.rGeq FFxlHz8Ils93fCJBFA- JOkzwzv6EKiLVYCE2k8", "data": { "user": { " _id": "63da9fe4af1b20aa9ae3539e", "email": "ricky1@gmail.com", "username": "ricky1", "role": "customer", "passwordChangedAt": "2023-04- 27T10:22:17.064Z", " __v": 0, "verified": false } } }	Passed
	PATCH	/api/v1/users/pa ss/	Authorization: Bearer {{jwt}}	{ "password": "aminamin", "newpassword":	200	{ "status": "success", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.	Passed

				"onlysecret321" }		eyJpZCI6IjYzZDYzNjZhYWYxMmE3ZDQzOTE2NjAzYiIsImIhdCI6MTY4MjU5MTA2NCwiZXhwIjoxNjkwMzY3MDY0fQ.upawB5atMY1AwWPgd-ldKnO389OutX3gzyHZEEr4z8w", "data": { "user": { "verified": false, "_id": "63d6366aaf12a7d43916603b", "email": "sda fdsaf@gmail.com", "username": "adsfdf", "role": "admin", "passwordChangedAt": "2023-04-27T10:24:23.289Z", "__v": 0 } } }	
	PATCH	/api/v1/users/	Authorization: Bearer {{jwt}}	{ "username": "mjxmjxmjx" }	200	{ "status": "success", "data": { "_id": "63d6366aaf12a7d43916603b", "email": "sda fdsaf@gmail.com", "username": "mjxmjxmjx", "role": "admin", "passwordChangedAt": "2023-04-27T10:26:41.693Z", "__v": 0, }	Passed

						"verified": false } }	
	DELETE	/api/v1/users/	Authorization: Bearer {{jwt}}	-	204	-	Passed
	GET	/api/v1/users/pa ss/reset/token/ch eck/:token/	-	-	200	OK	Passed
	GET	/api/v1/users/me	Authorization: Bearer {{jwt}}	-	200	{ "_id": "63d6366aaf12a7d43916603b", "email": "sdafdsaf@gmail.com", "username": "mjxmjxmjx", "role": "admin", "passwordChangedAt": "2023-04- 27T10:26:41.693Z", "__v": 0, "verified": false }	Passed

#### 4.4.3 End-to-End Testing

Berikut hasil *end-to-end testing (black-box testing)* yang telah dilakukan dengan menggunakan *browser*.

Tabel 4.5. Hasil *End-to-End Testing*

ID	Use Case	Test	Expected Result	Comment
UC01	Register	Passed	User berhasil melakukan registrasi	Use Mailtrap to see the email
UC02	Login	Passed	User berhasil masuk ke dalam aplikasi	-
UC03	Forgot Password	Passed	User berhasil melakukan reset password	Use Mailtrap to see the email
UC04	Logout	Passed	User berhasil keluar dari aplikasi	-
UC05	Edit User	Passed	User berhasil melakukan edit profile	-
UC06	View Single Page	Passed	Aplikasi berhasil menampilkan halaman <i>custom page</i>	-
UC07	Add Section	Passed	Admin berhasil menambahkan section pada halaman <i>custom page</i>	-
UC08	Edit Section	Passed	Admin berhasil melakukan modifikasi pada section halaman <i>custom page</i>	-
UC09	Delete Section	Passed	Admin berhasil menghapus section pada halaman <i>custom page</i>	-
UC10	Manage Pages	Passed	Admin berhasil melakukan modifikasi pages seperti 'add to header' dan set 'active'	-



UC11	Add Page	Passed	Admin berhasil menambahkan halaman baru pada aplikasi	-
UC12	Delete Page	Passed	Admin berhasil menghapus halaman dari aplikasi	-
UC13	View Products	Passed	Aplikasi menampilkan list produk	-
UC14	Add Product	Passed	Admin berhasil menambahkan produk baru	-
UC15	View Single Product	Passed	Aplikasi menampilkan sebuah produk	-
UC16	Edit Product	Passed	Admin berhasil melakukan modifikasi pada produk	-
UC17	Delete Product	Passed	Admin berhasil menghapus produk	-
UC18	Buy Product	Passed	User berhasil melakukan pembelian	The payment is set to test mode, so it doesn't debit customer's real money
UC19	Customize Theme	Passed	Admin berhasil melakukan modifikasi tema aplikasi	-
UC20	View Dashboard	Passed	Aplikasi menampilkan metrik aplikasi	-

## 4.5 Evaluasi

### 4.5.1 *Twelve Factor App*

Berikut merupakan hasil evaluasi berdasarkan prinsip *twelve factor app*.

#### 1. *Codebase*

*Codebase* yang digunakan di aplikasi *web* ini hanya satu. Di dalam satu *codebase* ini terdiri dari *code frontend*, *code backend*, konfigurasi NGINX *load balancer*, dan juga *kubernetes yml file*. *Codebase* ini juga ditaruh pada *Source Code Management Github* sebagai *revision tracking system*.

#### 2. *Dependencies*

Node Package Manager (NPM) digunakan untuk mengatur *dependencies* pada *project NodeJS*. Daftar *dependency* tercantum pada *file package.json* dan *package-lock.json*. Untuk meng-*install dependencies* tersebut diperlukan *command* “*npm install*”. *File-file dependencies* yang ter-*install* terkumpul pada *folder* “*node\_modules*”.

#### 3. *Config*

Konfigurasi-konfigurasi yang digunakan tidak di-*hardcode*, tetapi ditaruh di *file* terpisah, yaitu *file .env* untuk *development* dan juga *deployment .yaml file* untuk *production*.

#### 4. *Backing services*

*Backing service* yang digunakan adalah hubungan antara *code backend* dengan MongoDB.

```
13 const DB_HOST = process.env.DB_HOST || 'db';
14 const DB_PORT = process.env.DB_PORT || 27017;
15 const DB_USERNAME = process.env.DB_USERNAME;
16 const DB_PASSWORD = process.env.DB_PASSWORD;
17
18 const CONNECTION_URL = `mongodb://${DB_USERNAME}:${DB_PASSWORD}@${DB_HOST}:${DB_PORT}/ourstore?retryWrites=true&majority`;
```

Gambar 4.40. Evaluasi *Backing Services*

Koneksi ke MongoDB dapat dikonfigurasi pada *environment variable* DB\_HOST, DB\_PORT, DB\_USERNAME, dan DB\_PASSWORD tanpa harus mengubah *code backend*. Membuat *code* dengan *backing service* yang sangat fleksibel tidak mudah karena

masing-masing *third party service* biasanya membutuhkan *driver* yang spesifik.

### 5. Build, Release, Run

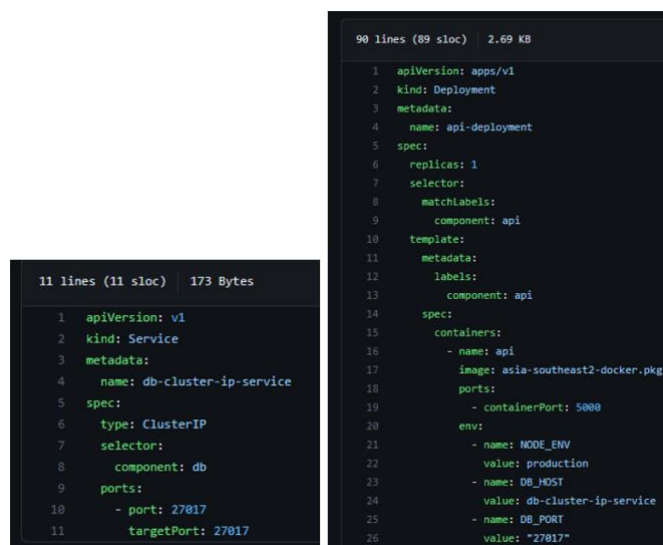
Aplikasi ini memiliki *CICD pipeline* yang memenuhi tahapan *build*, *release*, dan *run*. *Build* ketika *developer* melakukan *push* ke Github *repository*, kemudian *github workflow* akan melakukan “docker build” berdasarkan *file* konfigurasi *Dockerfile*. Tahapan *release* dan *run* digabung menjadi satu proses dengan *file* konfigurasi *.yaml* di mana *file* ini mengambil *image* yang telah di-*build* dan memasang *environment variable* untuk di *environment production*.

### 6. Processes

Proses di aplikasi ini seperti *frontend* maupun *backend* tidak menyimpan data *user* sehingga proses berjalan dengan *stateless*. Semua data disimpan di dalam MongoDB *database* maupun Google Cloud Storage.

### 7. Port binding

Kubernetes memakai *port binding* untuk meneruskan *request* dari luar Kubernetes *cluster* ke dalam *cluster*, demikian juga untuk *request* dari luar *Pods* ke dalam *Pods*. Oleh karena itu, *service-service* di Kubernetes seperti *ClusterIP* dan *LoadBalancer* diperlukan sebagai *port binding*.



Gambar 4.41. Evaluasi Port Binding

### 8. Concurrency

Kubernetes juga mendukung *concurrency process* dengan menggunakan *horizontal scaling*. Jumlah proses ini dapat diatur dengan mengubah *value spec.replicas* pada *yaml deployment file*.

### 9. Disposability

Sama seperti poin 8, Kubernetes melakukan pengawasan terhadap jumlah *Pods* berdasarkan *value* pada *spec.replicas* di *yaml deployment file*. Pengawasan ini dilakukan oleh *controller manager*, *scheduler*, dan *etcd database* yang terletak pada *control plane* Kubernetes.

### 10. Dev/Prod Parity

*Environment development* dan *production* di aplikasi ini dibuat semirip mungkin. *Backing service development* dan *production* juga hampir tidak ada beda, kecuali *database* MongoDB yang digunakan. Kedua *environment* ini sama-sama memakai *Docker container* saat *running software*. Perbedaannya adalah di *development* memakai *docker-compose* sedangkan di *production* sudah memakai Kubernetes. Selain itu, perbedaan juga ada pada *return error response* di *development* yang lebih *verbose*.

### 11. Logs

*Logging* terdapat di setiap *Pods* yang berjalan pada Kubernetes. Di dalam *code backend* juga terdapat *dependency* Morgan yang melakukan *logging* ketika ada *request* ke sebuah *endpoint* tertentu. *Centralized logging* dilakukan oleh *GCP logging service* di mana mengambil semua *log* terhadap *resource* GCP termasuk GKE.

### 12. Admin Processes

Di dalam GCP, disediakan *terminal* untuk melakukan konfigurasi dan administrasi *resource* di dalam GCP. *Command* “*gcloud*” digunakan untuk melakukan *management resource* GCP. Sedangkan *command* “*kubectl*” sudah *pre-installed* dan digunakan sebagai *interface* kepada Kubernetes *cluster*, seperti menambah atau menghapus *deployment* dan *service*. Terlebih lagi, terdapat *command* “*kubectl exec -it <Pods name> – bash*” untuk masuk ke dalam *terminal*

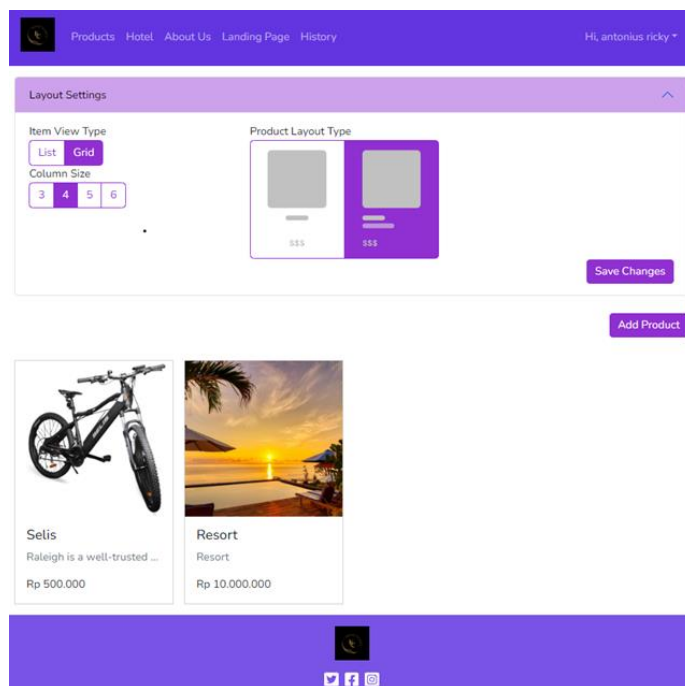
*pods* yang sedang berjalan. Hal ini digunakan untuk meminta *Certificate Signing Request* (CSR) kepada Let's Encrypt server.

#### 4.5.2 *Eight Golden Rules*

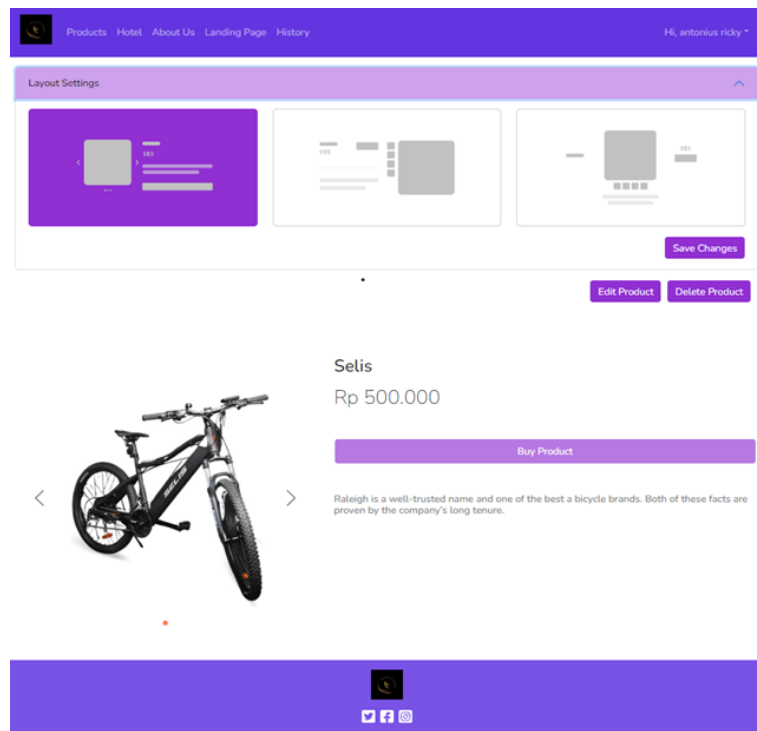
Berikut merupakan hasil evaluasi *Eight Golden Rules*.

##### 1. *Strive for Consistency*

Tampilan pada aplikasi terdiri atas *header* dan *footer* untuk setiap halaman, dan untuk penggunaan warna menyesuaikan dengan tema warna yang sudah dipilih dari pengaturan tema, yang menyesuaikan warna primer dan sekunder. Tombol modifikasi yang peletakannya sama dengan yang lain. *Font* yang digunakan untuk setiap bagian menu, isi konten dan lainnya menggunakan ukuran dan jenis yang sama, adapun seperti tampilan konfigurasi *layout* yang dibuat menjadi pilihan-pilihan yang telah ditentukan.



Gambar 4.42. Evaluasi *Strive for Consistency* 1

Gambar 4.43. Evaluasi *Strive for Consistency* 2

## 2. *Seek Universal Usability*

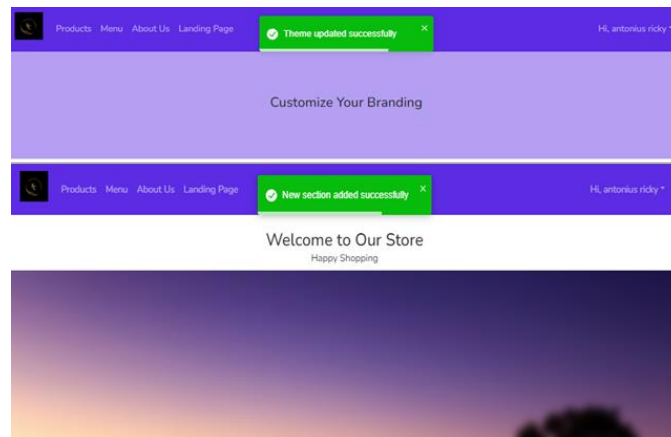
Tampilan bahasa yang dipilih dengan bahasa Inggris untuk mencangkup banyak pengguna agar lebih mudah memahami dalam menggunakan aplikasi. Bahasa yang umum juga akan membuat lebih bagus untuk dilihat karena sudah terbiasa dengan tampilan bahasa yang digunakan setiap harinya.

Gambar 4.44. Evaluasi *Seek Universal Usability*

## 3. *Offer Informative Feedback*

Tampilan untuk memberikan informasi terhadap apa yang *user* lakukan seperti perubahan, penambahan, penghapusan akan

ditampilkan untuk memudahkan pengguna dalam mengetahui kondisi atau respon aplikasi terhadap apa yang dilakukan.



Gambar 4.45. Evaluasi *Offer Informative Feedback*

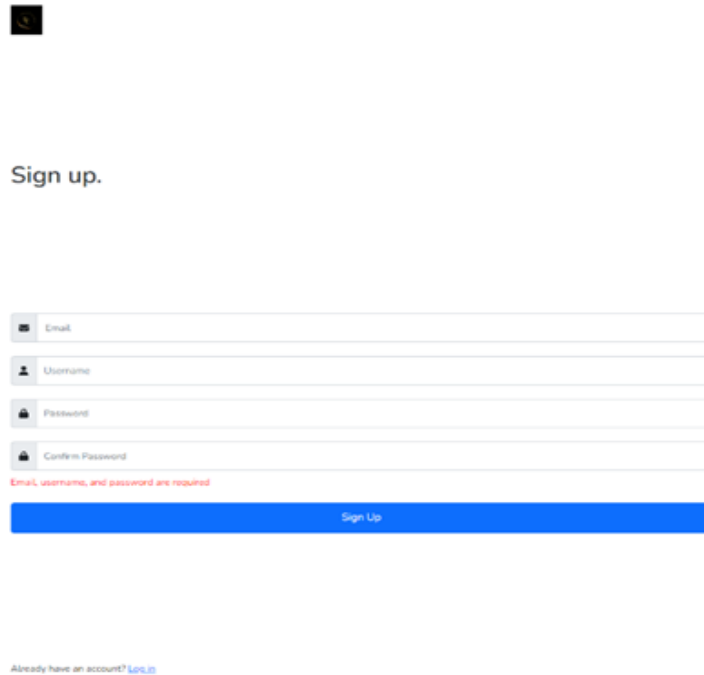
#### 4. *Design Dialogs to Yield Closure*

Tampilan aplikasi akan memunculkan pesan untuk memberitahukan pengguna bahwa aksinya dikelompokkan menjadi bagian awal, tengah, dan akhir (seperti perubahan *password*, dimana dimulai dari permintaan perubahan *password*, konfirmasi perubahan, dan perubahan *password*). Pesan tersebut menyampaikan bahwa proses sudah selesai dan dapat dilanjutkan untuk proses lainnya.

Gambar 4.46. Evaluasi *Design Dialogs to Yield Closure*

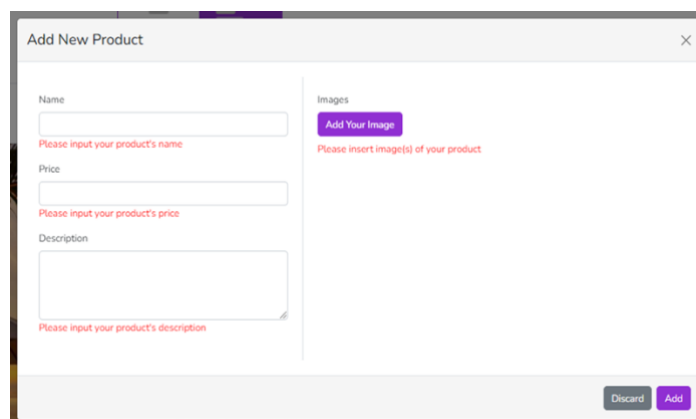
### 5. *Prevent Errors*

Tampilan aplikasi untuk membantu *user* menghindari kesalahan dengan menampilkan pesan yang berhubungan dengan pengisian *form* seperti pada saat mengisi *form* autentikasi, penambahan atau perubahan pada komponen produk.



The image shows a 'Sign up' form with four input fields: Email, Username, Password, and Confirm Password. Below the Password field, a red error message states: 'Email, username, and password are required'. A blue 'Sign Up' button is at the bottom. Below the button, a link says 'Already have an account? [Login](#)'.

Gambar 4.47. Evaluasi *Prevent Errors* 1



The image shows an 'Add New Product' form. It has three input fields on the left: Name, Price, and Description. Each field has a red error message below it: 'Please input your product's name', 'Please input your product's price', and 'Please input your product's description'. On the right, there is an 'Images' section with an 'Add Your Image' button and a red error message: 'Please insert image(s) of your product'. At the bottom right, there are 'Discard' and 'Add' buttons.

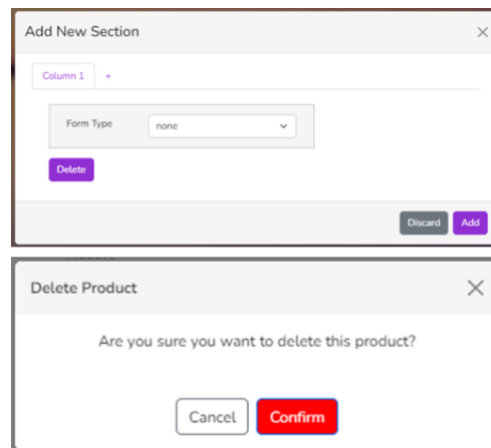
Gambar 4.48. Evaluasi *Prevent Errors* 2

### 6. *Permit Easy Reversal of Actions*

Tampilan pada setiap melakukan aksi perubahan pada aplikasi akan dilengkapi dengan tombol untuk membatalkan aksi, atau



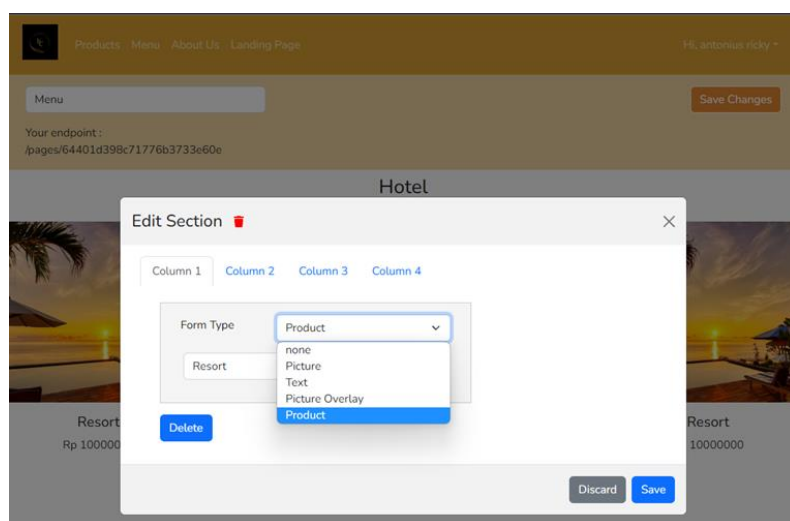
mengembalikan seperti kondisi semula sehingga dapat meminimalisir kesalahan. Tindakan seperti untuk penghapusan juga akan ada konfirmasi lanjutan agar menghindari kesalahan dalam melakukan hal tersebut.



Gambar 4.49. Evaluasi *Permit Easy Reversal of Actions*

## 7. *Keep Users in Control*

Tampilan aplikasi dibuat untuk bisa dikontrol atau dipersonalisasi sesuai dengan keinginan pengguna. Pengaturan tersebut dapat disesuaikan seperti mengubah bagian pada halaman agar dapat menyesuaikan dengan gambaran tampilan halaman sesuai dengan preferensi pengguna yang akan ditampilkan kepada pengunjung halaman nantinya.



Gambar 4.50. Evaluasi *Keep Users in Control*

### 8. *Reduce Short-term Memory Load*

Tampilan aplikasi dibuat dengan sederhana dan dibuat agar tidak membebani pengguna dalam mengakses fitur aplikasi, dengan ditampilkannya isi konten terakhir yang sudah diisi sebelumnya, pada produk ketika saat akan melakukan perubahan pada produk agar tidak perlu untuk mengingat apa isi kontennya.

Gambar 4.51. Evaluasi *Reduce Short-term Memory Load*

### 4.5.3 Lima Faktor Manusia Terukur

Berikut merupakan hasil dari Evaluasi Lima Faktor Manusia Terukur yang dilakukan dengan cara mewawancarai *user*. Berikut merupakan hasil dari Evaluasi Lima Faktor Manusia Terukur yang dilakukan dengan *user*,

#### 1. Evaluasi Waktu Belajar

Berdasarkan dari hasil evaluasi dengan *user*, dari rata-rata waktu yang didapat, menunjukkan bahwa aplikasi dapat dengan mudah dipahami oleh penggunaannya, sehingga aplikasi bisa digunakan dalam jangka waktu yang lama tanpa perlu untuk memahami kembali apa yang sudah pernah dilakukannya. *User* juga dapat dengan mudah mengingat tampilan yang sudah dipahami.

#### 2. Evaluasi Kecepatan Kinerja

Berdasarkan dari hasil evaluasi dengan *user*, bahwa untuk waktu yang digunakan dalam menggunakan aplikasi untuk melaksanakan tugas atau aktivitas yang meliputi perubahan pada produk dan halaman aplikasi cukup baik, mengingat performa yang dirasakan itu

juga cukup baik, dan alur dalam penggunaan aplikasi juga dapat dengan lancar diproses oleh sistem.

### 3. Evaluasi Tingkat Kesalahan Pengguna

Berdasarkan dari hasil evaluasi dengan *user*, tingkat kesalahan yang dapat dilakukan oleh *user* memiliki persentase yang sedikit dan kemungkinan terjadinya error dari pengaruh pengguna terhadap *user* juga cukup kecil. Dari segi aplikasi kesalahan dapat juga diminimalisir dengan adanya pengawasan juga dari sisi sistem.

### 4. Evaluasi Daya Ingat

Berdasarkan dari hasil evaluasi dengan *user*, dengan informasi aplikasi yang telah *user* pahami ketika mencoba untuk mempelajari penggunaan dari tiap komponen, dapat dipastikan bahwa *user* mudah untuk bisa memahami dan mengingat kembali penggunaannya sesuai dengan bagaimana pertama kali mereka mencoba dalam rentan waktu yang cukup lama. Karena memang dari sisi aplikasi didesain untuk memudahkan *user* dalam melakukan tindakan perubahan seperti menambah, merubah, maupun menghapus isi konten.

### 5. Evaluasi Kepuasan Subjektif

Berdasarkan dari hasil evaluasi dengan *user*, banyak yang berpendapat baik terhadap apa yang dirasakan ketika melihat dan mencoba tampilan pada aplikasi, dilihat dari segi tampilannya yang sudah konsisten terhadap pemilihan tema dan fungsi tiap komponen, dari segi kualitas manfaat dari aplikasi juga banyak nilai positif yang dapat diimplementasikan, serta kepuasan dalam keseluruhan aspek dari aplikasi yang memiliki sistem personalisasi yang dapat memuaskan keinginan *user* terhadap tampilan apa yang mereka inginkan.

