BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Dalam penelitian ini melibatkan pengumpulan data pelaporan masalah sebelumnya yang terdiri dari 728 data yang telah diberi pelabelan dari *history* dari data reporting yang ada pada Zettabyte yaitu *developer* (*front-end* atau *back-end)*. Data tersebut kemudian dibersihkan dengan melakukan pemotongan teks menjadi kata-kata, membuang tanda baca, dan menghilangkan kata-kata yang sering muncul menggunakan metode *TF-IDF*. Setelah itu, data tersebut akan diimplementasikan ke dalam algoritma klasifikasi *Naïve Bayes* untuk melakukan pengklasifikasian tipe pengguna. Dengan melakukan langkah-langkah tersebut, diharapkan dapat membangun sebuah model yang dapat memprediksi dengan akurasi tinggi deskripsi dari pelaporan bug tersebut merupakan *front-end* atau *back-end*.

4.2 PEMBAHASAN

4.2.1 Akurasi *Naïve Bayes*

Hasil yang didapat saat pengujian dengan menggunakan *confusion matrix* yang ditunjukkan pada Tabel 4.1 :

Tabel 4.1 Hasil *Confusion Matrix*

Tipe *Developer*	Prediksi *Front-End*	Prediksi *Back-End*
Front-End (0)	26	21
Back-End (1)	1	94

Diperoleh beberapa informasi tentang performa model klasifikasi *Naïve Bayes* dalam memprediksi tipe *developer* berdasarkan deskripsi yang di *report*:

1. Terdapat 26 data tipe *developer front-end* (kelas 0) yang berhasil diprediksi dengan benar sebagai tipe *developer front-end*. Hal ini ditunjukkan oleh nilai *True Positive* (TP) sebesar 26 pada kolom "Prediksi *Front-End"* dan baris "Tipe *Developer Front-End"*

2. Terdapat 94 data tipe *developer back-end* (kelas 1) yang berhasil diprediksi dengan benar sebagai tipe developer *back-end.* Hal ini ditunjukkan oleh nilai *True Negative* (TN) sebesar 94 pada kolom "Prediksi *Back-End"* dan baris "Tipe *Developer Back-End"*
3. Terdapat 1 data tipe *developer back-end* (kelas 1) yang salah diprediksi sebagai tipe *developer front-end*. Hal ini ditunjukkan oleh nilai *False Positive* (FP) sebesar 1 pada kolom "Prediksi *Front-End*) dan baris "Tipe *Developer Back-End*"
4. Terdapat 21 date tipe *developer front-end* (kelas 1) yang salah diprediksi sebagai tipe *developer back-end*. Hal ini ditunjukkan oleh nilai *False Negative* (FN) sebesar 21 pada kolom "Prediksi *Back-End*) dan baris "Tipe *Developer Front-End*).

Dari hasil evaluasi *confusion matrix* ini, dapat dilihat bahwa model memiliki kinerja yang baik dalam memprediksi tipe *developer back-end*, karena TN yang tinggi dan FP yang rendah. Namun, model cenderung memiliki kesulitan dalam memprediksi tipe *developer front-end*, terlihat dari FN yang nol namun FP yang cukup signifikan. Gambar 4.1 menunjukkan akurasi dari *confusion matrix.*


```
Confusion Matrix

[[26 21]
 [ 1 94]]


Classification Report

              precision    recall  f1-score   support

           0       0.96      0.55      0.70        47
           1       0.82      0.99      0.90        95

    accuracy                           0.85       142
   macro avg       0.89      0.77      0.80       142
weighted avg       0.87      0.85      0.83       142
```

Gambar 4.1 *Confusion Matrix*

Tipe *developer front end* mendapatkan *precision* sebesar 96%, *recall* 55% dan F-1 *score* 70%. Dari *confusion matrix* yang dihasilkan dapat disimpulkan bahwa presisi 96% menghasilkan mayoritas dari prediksi yang dilakukan oleh model untuk tipe *developer front-end* benar. *Recall* yang *relative* rendah sebesar 55% yang menandakan bahwa model memiliki kesulitan dalam mengidentifikasi semua sampel data yang sebenarnya termaksud ke dalam tipe *developer front-end*. F-1 Score yang lebih rendah 70% menunjukkan bahwa terdapat *trade-off* antara presisi dan *recall*. F-1 *Score* menggabungkan presisi dan *recall*, sehingga nilai yang lebih rendah menunjukkan bahwa model memiliki keseimbangan yang kurang dalam mengklasifikasikan sampel-sampel data *front-end* dengan benar secara keseluruhan.

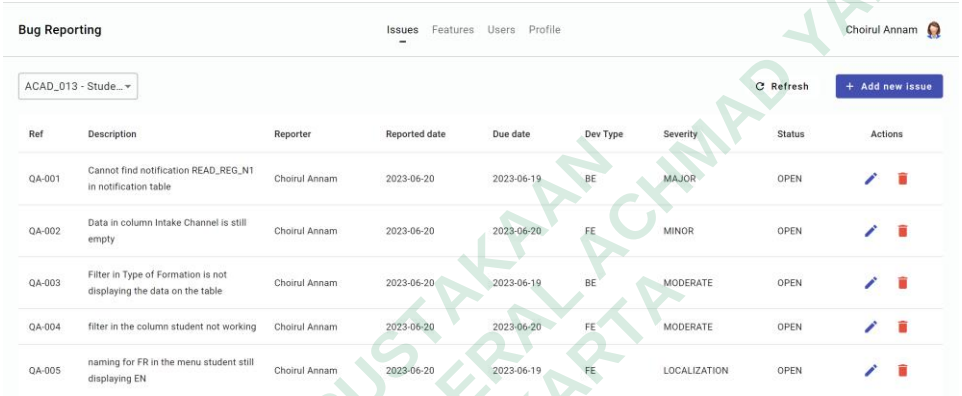
Tipe *developer back-end* mendapatkan *precision* 82%, *recall* 99% dan F-1 *score* 90%. Dapat disimpulkan bahwa hasil presisi yang cukup tinggi sebesar 82% menunjukkan bahwa sebagian besar prediksi yang dilakukan oleh model untuk tipe *developer back-end* adalah benar. Hasil *recall* yang sangat tinggi 99% menandakan bahwa model sangat baik dalam mengidentifikasi dan mengklasifikasikan hampir semua sampel data yang sebenarnya termaksud ke dalam tipe *developer back-end.* F-1 *score* yang baik sebesar 90% menunjukkan bahwa model memiliki keseimbangan yang baik antara presisi dan *recall*, yang menunjukkan kinerja yang baik secara keseluruhan dalam mengklasifikasikan sampel-sampel data *back-end.*

Kesimpulan dari hasil confusion matrix terhadap model, bahwa model klasifikasi untuk tipe *developer back-end* menunjukkan performa lebih baik dibandingkan dengan model untuk tipe *developer front-end.* Model untuk tipe *developer back-end* memiliki recall yang tinggi, menandakan kemampuannya dalam mengklasifikasikan semua sampel data back-end yang benar, sementara model untuk tipe *developer front-end* memiliki presisi yang lebih tinggi, menandakan kemampuannya dalam melakukan prediksi yang benar untuk sampel data front-end. Keseimbangan antara presisi dan recall diukur dengan F1-*score*, bahwa model untuk tipe *developer back-end* memiliki F1-*score* yang lebih tinggi menunjukkan kinerja keseluruhan yang lebih baik dibandingkan dengan model untuk tipe *developer front-end.*

4.2.2 Implementasi Desain Antarmuka

4.2.2.1 Implementasi Halaman Tabel *Bug Reporting*

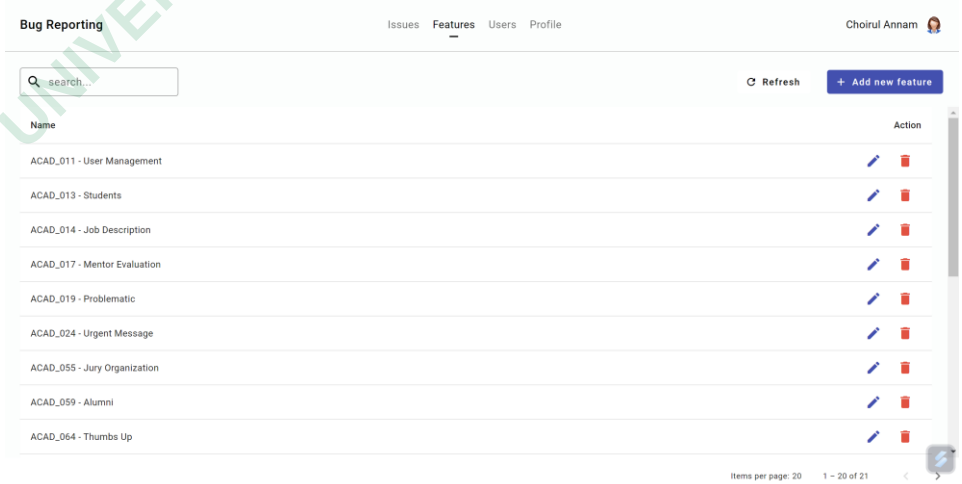
Implementasi halaman tabel *reporting* yang terdapat kolom referensi, *description*, *reporter*, *reporter date*, *due date*, *dev type*, *severity*, status dan *action* yang menampilkan pelaporan *bug* yang telah diinputkan sebelumnya pada Gambar 4.2. dan dapat menyunting dan menghapus pelaporan *bug* dengan menekan tombol yang ada pada kolom *action* yang ditunjukkan pada Gambar 4.2.



Gambar 4.2 Tabel *Bug Reporting*

4.2.2.2 Implementasi Halaman Tabel *Feature* pada Admin

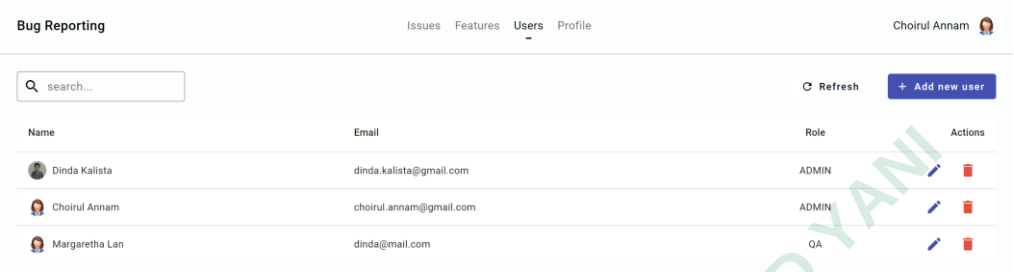
Menampilkan halaman tabel *feature* yang ditunjukkan pada Gambar 4.3 untuk menyimpan nama *feature* baru yang ditambahkan admin dan dapat mengubah atau menghapus *feature*.



Gambar 4.3 Tabel *Feature* Admin

4.2.2.3 Implementasi Halaman Tabel *User* pada Admin

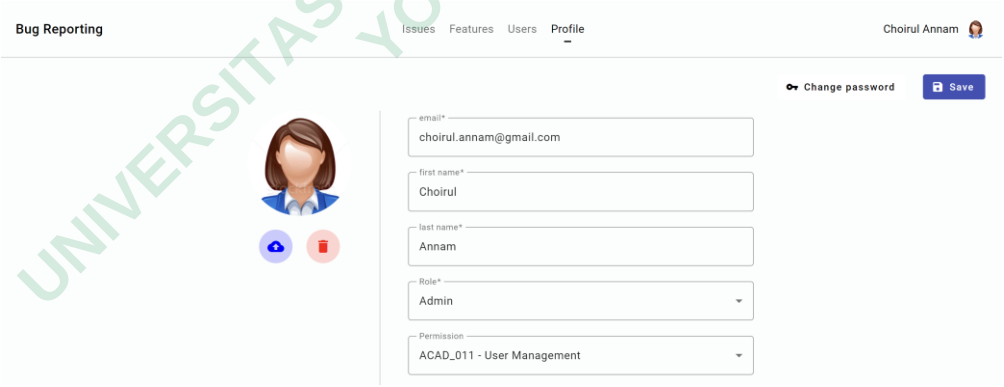
Implementasi dari halaman tabel *user*. Terdapat kolom *name*, *email*, *role* dan *action* yang dapat menyunting atau menghapus pengguna jika suatu saat pengguna tidak lagi berada pada perusahaan yang ditunjukkan pada Gambar 4.4.



Gambar 4.4 Tabel *User*

4.2.2.4 Implementasi Halaman *Profile*

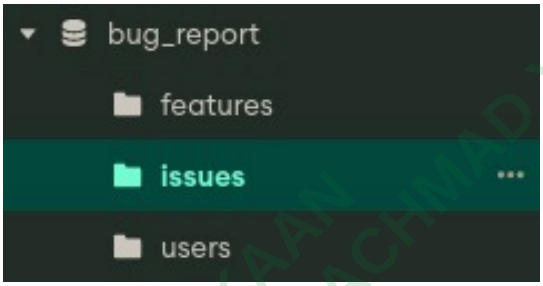
Implementasi dari halaman *profile* yang menampilkan *email*, *first name*, *last name* dari admin yang dapat disunting melalui *field* dan admin dapat menyunting atau menghapus profil dan *password* dari menu profil yang ditunjukkan pada Gambar 4.5.



Gambar 4.5 Halaman *Profile*

4.2.3 Basis Data

Penelitian ini menggunakan basis data MongoDB untuk menyimpan data *feature*, *issue* dan *user* didalam basis data *bug_report* pada Gambar 4.6. Penerapan skema pada *features, issues,* dan *users* memungkinkan akses dan pengolaan data yang efisien, sehingga memudahkan dalam melakukan berbagai operasi seperti pencarian, pengurutan, atau analisis statistik.



Gambar 4.6 Basis data

Skema data *features* yang ditunjukkan pada Tabel 4.2 menggambarkan skema koleksi dari database *features* yang terdapat dua *field* utama yaitu “id” dan “*name*”. “id” digunakan sebagai bidang untuk mengidentifikasi dokumen atau entitas daam database, sementara “nama” merupakan field dengan tipe data string yang menyimpan informasi mengenai nama dari setiap *features*.

Tabel 4.2 Skema Data *Features*

Nama	Tipe
_id	ObjectId
name	String

Skema data *issue* yang menyimpan basis data dari field yang diinputkan pada sistem. Skema ini dirancang untuk mengelola issue atau masalah yang muncul selama pengembangan sistem yang ditunjukkan pada Tabel 4.3. Skema data issue mencakup berbagai field yang menyimpan informasi terkait setiap issue yang didapatkan dalam proses pengembangan sistem yaitu *id, ref, feature_id,*

description, severity, reported_date, due_date, reported_id, status, dev_type, dev_id, dev_eta, dev_actual, qa_id, qa_eta dan *qa_actual* . Skema yang dirancangan untuk mengelola *issue* yang muncul dalam pengembangan sistem.

Tabel 4.3 Skema Data *Issue*

Nama	Tipe
_id	ObjectId
ref	String
feature_id	ObjectId
description	String
severity	String
reported_date	Date
due_date	Date
reporter_id	ObjectId
status	String
dev_type	String
dev_id	ObjectId
dev_eta	Int32
dev_actual	Int32
qa_id	ObjectId
qa_eta	Int32
qa_actual	Int32

Menampilkan skema koleksi dari database untuk pengelolaan data *user* dalam pengembangan sistem. Skema data *user* ini mencakup berbagai *field* yang menyimpan informasi terkait setiap *user* yang terlibat dalam proses pengembangan sistem, *user* yang bertanggung jawab saat pengembang sistem adalah admin, *quality assurance* (qa) dan *developer (front-end* dan *back-end)* yang mencakup informasi terkait *id, picture_id, picture_url, email, first_name, last_name,*

password, active, feature_id dan *role*. Skema yang dirancangan untuk mengelola *user* yang muncul dalam pengembangan sistem yang ditunjukkan pada Tabel 4.4.

Tabel 4.4 Skema data *user*

Nama	Tipe
_id	ObjectId
picture_id	String
picture_url	String
email	String
first_name	String
last_name	String
password	String
active	Boolean
feature	Array
role	String

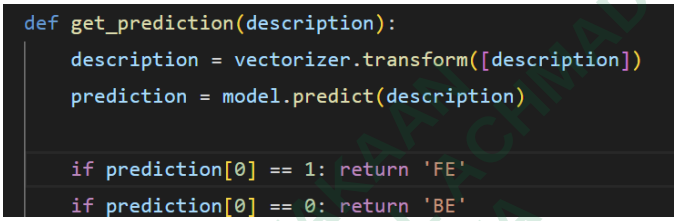
4.2.4 Fitur-Fitur Sistem

Sistem *bug reporting* memiliki beberapa fitur sebagai berikut:

1. Admin, qa dan *developer (front-end* dan *back-end*) memiliki akses untuk *login* dan *logout*
2. Memiliki dialog ketika menambahkan dan memperbarui *issue*, *feature*, dan *user*
3. Menampilkan tabel dan kolom untuk seluruh *reporting*, *feature*, dan *user* yang dapat ditambahkan, diperbarui dan dihapus
4. Memiliki *sorting* untuk setiap kolom di tabel *issue*, *feature*, dan *user*
5. Admin, qa dan *developer (front-end* dan *back-end*) dapat melihat dan mengubah *profile*
6. Admin, qa, dan *developer (front-end* dan *back-end*) dapat mengubah kata sandi

4.2.5 Implementasi ke Sistem *Reporting*

Kode yang ditunjukkan pada Gambar 4.7 merupakan fungsi *get_prediction* yang menerima deskripsi bug sebagai input dan menghasilkan prediksi tipe *developer* berdasarkan deskripsi. Fungsi in imenggunakan metode *vektorisasi* untuk mengubah deskripsi menjadi *vector* fitur dana memprediksi tipe *developer* menggunakan model yang telah dilatih sebelumnya. Jika dari prediksi yang dihasilkan adalah 1, maka fungsi akan mengembalikan nilai "*front-end"* dan jika prediksi adalah 0, maka fungsi akan mengembalikan nilai "*back-end*"



Gambar 4.7 Kode prediksi tipe *developer*

Kode yang ditunjukkan pada Gambar 4.8. merupakan kode untuk *create issue* yang mengimplementasikan pemanggilan fungsi prediksi '*get_predicition*' untuk memperoleh tipe *developer* berdasarkan deskripsi isi yang telah dibersihkan sebelumnya. Setelah membersihkan data deskripsi dan mengonversinya menjadi kamus melalui '*issue.dict()*', kode menambahkan *entri* baru '*dev_type*" ke dalam kamus isu. Nilai *entri* yang diperoleh dengan memanggil fungsi '*get_prediction*' dan menyediakan deskripsi isu sebagai *argument*. Dengan menggunakan fungsi prediksi, sistem secara otomatis menentukan tipe *developer* yang paling sesuai dengan deskripsi bug yang diberikan

```python
# Create issue
@router.post("/", response_description="Create new issue")
def create_issue(request: Request, issue: IssueCreateModel = Body(...)):
    if not request.app.permission["authenticated"]:
        return JSONResponse(status_code=401, content=jsonable_encoder({"detail": "Authentication failed! Token not provided"}))

    # cuma admin dan qa yang bisa create issue
    if request.app.permission["role"] != "ADMIN" and request.app.permission["role"] != "QA":
        return JSONResponse(status_code=401, content=jsonable_encoder({"detail": "You do not have access to add issues"}))

    try:
        issue = issue.dict()
        issue['dev_type'] = get_prediction(issue['description'])

        request.app.database["issues"].insert_one(issue)
        return JSONResponse(status_code=201, content={"detail": "Issue created successfully"})

    except Exception as error:
        raise HTTPException(status_code=500, detail=str(error))
```

Gambar 4.8 Kode untuk menambahkan laporan bug

Kode yang ditunjukkan pada Gambar 4.9 terdapat fungsi "*update issue*" yang digunakan untuk memperbarui isu dalam basis data. Fungsi ini mengintegrasikan metode klasifikasi *Naïve Bayes* untuk memprediksi kelas "*dev_type*" berdasarkan deskripsi isu "*description*". Setelah mendapatkan prediksi, data isu diperbarui dalam basis data.

```python
# Update issue
@router.put("/{id}", response_description="Update issue")
def update_issue(id: str, request: Request, issue: Optional[IssueUpdateModel] = Body(None)):
    if not request.app.permission["authenticated"]:
        return JSONResponse(status_code=401, content=jsonable_encoder({"detail": "Authentication failed! Token not provided"}))
    # cuma admin yang bisa update issue
    # if request.app.permission["role"] != "ADMIN":
    #     return JSONResponse(status_code=401, content=jsonable_encoder({ "detail": "access denied" }))
    try:
        issue = issue.dict(exclude_none=True)

        if "dev_type" in issue and "description" in issue:
            issue["dev_type"] = get_prediction(issue["description"])
        update_result = request.app.database["issues"].update_one(
            {"_id": ObjectId(id)},
            {"$set": issue}
        )
        if update_result.modified_count != 1:
            return JSONResponse(status_code=422, content=jsonable_encoder({"detail": "Failed to update issue"}))
        return JSONResponse(status_code=200, content={"detail": "Issue updated successfully"})
```

Gambar 4.9 Kode untuk memperbarui laporan bug

4.2.6 *Black Box Testing*

Pengujian *black box testing* adalah metode pengujian perangkat lunak yang fokus pada perilaku dari sistem. *Black box* digunakan untuk menguji sistem menerima laporan *bug* dengan benar dan memprosesnya dengan tepat. Keterangan pada saat pengujian *black box* ditunjukkan pada Tabel 4.5.

Tabel 4.5 Keterangan *Black Box Testing*

Skala Jawaban	Keterangan	Skor
B	Berhasil	2
TB	Tidak Berhasil	1

4.2.6.1 *Black box* untuk Admin

Black box testing untuk admin bertujuan untuk melakukan pengujian sistem, admin menguji semua fitur yang tersedia pada sistem seperti menu isu, fitur, pengguna dan profil. Admin memastikan bahwa fungsionalitas dari sistem dapat berjalan dengan baik seperti membuat, memperbarui, melihat dan menghapus isu, fitur maupun pengguna dari tampilan antarmuka dan terdapat 2 admin yang melakukan test pada sistem yang ditunjukkan pada Tabel 4.6.

Tabel 4.6 *Black box* untuk Admin

No	Pertanyaan	B	TB
1	Admin *login* sistem *bug reporting* menggunakan email dan password yang valid	2	0
2	Admin login sistem *bug reporting* menggunakan email dan password yang tidak valid	2	0
3	Admin logout dari sistem dan menuju kehalaman login	2	0
4	Admin mengeklik menu *feature*	2	0
5	Admin mengeklik tombol *"add feature"*	2	0
6	Admin menambahkan fitur menggunakan nama fitur yang valid dan menampilkan di tabel fitur	2	0
7	Admin menambahkan fitur menggunakan nama fitur yang tidak valid dan mendapatkan *popup error*	2	0

No	Pertanyaan	B	TB
8	Admin memperbarui nama fitur yang valid dan menampilan di tabel fitur	2	0
9	Admin memperbarui nama fitur yang tidak valid dan mendapatkan *popup error*	2	0
10	Admin melakukan pencarian berdasarkan nama fitur yang ada pada tabel dan hanya menampilkan fitur yang dicari	2	0
11	Admin melakukan pencarian berdasarkan nama fitur yang tidak ada pada tabel dan tidak menampilkan *feature* yang dicari pada tabel	2	0
12	Admin menghapus *feature* dan tidak ditampilkan pada tabel	2	0
13	Admin mengeklik menu user	2	0
14	Admin mengeklik tombol “add user”	2	0
15	Admin menambahkan *user* baru menggunakan *email* dan *password* yang valid dan berhasil menambahkan *user*	2	0
16	Admin menambahkan *user* baru menggunakan *email* dan *password* yang tidak valid dan tidak berhasil menambahkan *user*	2	0
17	Admin memperbarui *user* menggunakan *email* yang valid dan menampilan di tabel fitur	2	0
18	Admin memperbarui *user* menggunakan *email* yang tidak valid dan tidak dapat menyimpan *user* yang sudah diperbarui	2	0
19	Admin melakukan pencarian berdasarkan nama *user* yang ada pada tabel dan hanya menampilkan *user* yang dicari	2	0
20	Admin melakukan pencarian berdasarkan nama *user* yang tidak ada pada tabel dan tidak menampilkan *user* yang dicari pada tabel	2	0
21	Admin mengeklik menu *profile*	2	0
22	Admin mengubah profil (*first name*, *last name* dan *email*) dan berhasil memperbarui profil	2	0
23	Admin mengubah foto profil dan berhasil memperbarui foto profil	2	0
24	Admin menghapus foto profil dan foto profil terhapus	2	0
25	Admin mengubah kata sandi menggunakan kata sandi yang valid dan menyimpan perubahan	2	0

No	Pertanyaan	B	TB
26	Admin mengubah kata sandi menggunakan kata sandi yang tidak valid dan tidak dapat menyimpan perubahan	2	0
27	Admin mengeklik menu *issue*	2	0
28	Admin mengeklik tombol *"add issue"*	2	0
29	Admin menambahkan *issue menggunakan reporting issue* yang valid dan menampilkan di tabel issue	2	0
30	Admin menambahkan issue menggunakan *reporting issue* yang tidak valid dan mendapatkan *popup error*	2	0
31	Admin memperbarui *reporting issue* yang valid dan menampilan di tabel *issue*	2	0
32	Admin memperbarui *reporting issue* yang tidak valid dan mendapatkan *popup error*	2	0
33	Sorting berfungsi untuk setiap kolom pada tabel *reporting*	2	0
34	Admin memfilter berdasarkan nama *feature* yang ada pada tabel dan menampilkan *reporting issue* untuk *feature* yang dicari	2	0
35	Admin memfilter berdasarkan nama *feature* yang tidak ada pada tabel dan tidak menampilkan *reporting issue* untuk *feature* yang dicari	2	0
36	Admin menghapus *reporting issue* dan tidak ditampilkan pada tabel	2	0

Dari Tabel 4.6 menunjukkan 36 fungsionalitas dari sistem yang dilakukan oleh admin dapat bekerja dengan baik dan sistem dapat berjalan sesuai dengan kinerja yang diinginkan

4.2.6.2 *Black box* untuk *Quality Assurance*

Black box testing untuk *quality assurance* bertujuan untuk melakukan pengujian sistem, qa menguji fitur yang ada pada sistem seperti menu isu dan profil. Terdapat 5 qa yang memastikan bahwa fungsionalitas dari sistem dapat berjalan dengan baik seperti membuat, memperbarui, dan melihat isu dan profil pada tampilan antarmuka yang ditunjukkan pada Tabel 4.7.

Tabel 4.7 *Black box* untuk *quality assurance*

No.	Pertanyaan	B	TB
1	QA *login* sistem *bug reporting* menggunakan *email* dan *password* yang valid	5	0
2	QA *login* sistem *bug reporting* menggunakan *email* dan *password* yang tidak valid	5	0
3	QA logout dari sistem dan menuju kehalaman *login*	5	0
4	QA menambahkan *issue menggunakan reporting issue* yang valid dan menampilkan di tabel issue	5	0
5	QA menambahkan issue menggunakan *reporting issue* yang tidak valid dan mendapatkan *popup error*	5	0
6	QA memperbarui *reporting issue* yang valid dan menampilan di tabel *issue*	5	0
7	QA memperbarui *reporting issue* yang tidak valid dan mendapatkan *popup error*	5	0
8	Sorting berfungsi untuk setiap kolom pada tabel *reporting*	5	0
9	QA mencarai berdasarkan nama *feature* yang ada pada tabel dan menampilkan *reporting issue* untuk *feature* yang dicari	1	4
10	QA mencari berdasarkan nama *feature* yang tidak ada pada tabel dan tidak menampilkan *reporting issue* untuk *feature* yang dicari	5	0
11	QA mengeklik menu *profile*	5	0
12	QA mengubah profil (*first name, last name* dan *email*) dan berhasil memperbarui profil	5	0
13	QA mengubah foto profil dan berhasil memperbarui foto profil	5	0
14	QA menghapus foto profil dan foto profil terhapus	5	0
15	QA mengubah kata sandi menggunakan kata sandi yang valid dan menyimpan perubahan	5	0
16	QA mengubah kata sandi menggunakan kata sandi yang tidak valid dan tidak dapat menyimpan perubahan	5	0

Dari Tabel 4.7 menunjukkan dari 16 fungsionalitas pada sistem yang dilakukan oleh qa yang dapat berjalan dengan baik adalah 15 fungsionalitas dan 1 yang tidak berhasil yaitu fungsi untuk sorting pada setiap kolom.

4.2.6.3 *Black Box Testing* untuk *Developer*

Black box testing untuk *developer* (*front-end* dan *back-end*) memilik akses terbatas pada menu isu dan profil. Sebagai *developer* hanya dapat memperbarui status, estimasi (*eta*), waktu *actual* dan memberikan komentar pada isu yang telah dilaporkan oleh penguji. *Developer* tidak memiliki akses untuk menambahkan isu baru atau mengubah deskripsi isu yang telah dilaporkan oleh penguji. Untuk menu profil *developer* tetap dapat diakses untuk memperbarui infomasi profil pribadi dari *developer* dan terdapat 5 *developer* yang melakukan test terhadap sistem. Hasil *black box testing* ditampilkan Tabel 4.8.

Tabel 4.8 *Black box* untuk *developer* (*front end* atau *back-end*)

No.	Pertanyaan	B5	TB
1	*Developer* (*front-end* dan *back-end*) *login* sistem *bug reporting* menggunakan *email* dan *password* yang valid	5	0
2	*Developer* (*front-end* dan *back-end*) login sistem *bug reporting* menggunakan *email* dan *password* yang tidak valid	5	0
3	*Developer* (*front-end* dan *back-end*) *logout* dari sistem dan menuju kehalaman *login*	5	0
4	*Developer* (*front-end* dan *back-end*) hanya dapat melihat *reporting issue* yang diberikan oleh admin	5	0
5	*Developer* (*front-end* dan *back-end*) mencari berdasarkan nama *feature* yang ada pada tabel dan menampilkan *reporting issue* untuk *feature* yang dicari	5	0
6	*Developer* (*front-end* dan *back-end*) mencari berdasarkan nama *feature* yang tidak ada pada tabel dan tidak menampilkan *reporting issue* untuk *feature* yang dicari	5	0
7	*Sorting* berfungsi untuk setiap kolom pada tabel *reporting*	4	1
8	*Developer* (*front-end* dan *back-end*) hanya dapat mengubah status, eta, *actual* dan komen pada *reporting issue*	5	0
9	*Developer* (*front-end* dan *back-end*) mengubah status, eta, *actual*, dan komen menggunakan data yang valid dan memperbarui pada tabel	5	0
10	*Developer* (*front-end* dan *back-end*) mengubah status, eta, *actual* dan komen menggunakan data	5	0

No.	Pertanyaan	B5	TB
	yang tidak valid dan tidak memperbarui status, eta, actual dan komen		
11	*Developer* (*front-end* dan *back-end*) mengeklik menu *profile*	5	0
12	*Developer* (*front-end* dan *back-end*) mengubah profil (*first name, last name* dan *email*) dan berhasil memperbarui profil	5	0
13	*Developer* (*front-end* dan *back-end*) mengubah foto profil dan berhasil memperbarui foto profil	5	0
14	*Developer* (*front-end* dan *back-end*) menghapus foto profil dan foto profil terhapus	5	0
15	*Developer* (*front-end* dan *back-end*) mengubah kata sandi menggunakan kata sandi yang valid dan menyimpan perubahan	5	0
16	*Developer* (*front-end* dan *back-end*) mengubah kata sandi menggunakan kata sandi yang tidak valid dan tidak dapat menyimpan perubahan	5	0

Dari Tabel 4.8 menunjukkan 16 fungsionalitas terdapat sistem yang dapat bekerja dengan baik untuk *developer* (*front-end* dan *back-end*) sesuai dengan kinerja sistem yang diinginkan

4.2.6.4 Kesimpulan *Black Box Testing*

Berdasarkan dari pengujian *black box* terhadap sistem *bug reporting* dengan memanfaatkan metode klasifikasi naïve bayes di Zettabyte dapat disimpulkan bahwa admin terdiri dari 36 fungsionalitas yang dapat bekerja, sementara itu untuk qa terdiri dari 16 fungsionalitas sistem dapat digunakan, sedangkan *developer* (*front-end* dan *back-end*) terdiri dari 16 fungsionalitas sistem dapat digunakan. Dari pengujian *black box* terdapat 1 fungsionalitas yang tidak dapat bekerja dengan optimal yaitu "*sorting* untuk kolom yang berada pada tabel *reporting*" karena belum menerapkan *alphabetically order* pada sistem yang membuat *sorting* belum berjalan. Tetapi untuk fungsionalitas yang lain dapat berhasil, sehingga secara menyeluruh sistem ini siap diterapkan sesuai dengan kebutuhan pengguna.

4.2.7 Pengujian *User Acceptance Testing*

Responden yang akan mengisi kusioner dibagi menjadi 3 yaitu 5 responden *quality assurance*, 5 responden *developer* dan 2 responden admin yang akan dijelaskan dibawah ini:

4.2.7.1 Pengujian UAT Admin

Hasil presentase pertanyaan dari tingkat setiap jawaban yang didapatkan dari kusioner sebagai berikut.

1. Apakah tampilan aplikasi *bug reporting* menarik?

Tabel 4.9 Hasil Pertanyaan 1 Admin

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	0	5x0 = 0
S	4	1	4x1 = 4
KS	3	1	3x1 = 3
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			7

$$P = \frac{7}{10}\ x\ 100\%$$

= 70%

Dari presentase yang didapatkan pada Tabel 4.9 dengan menghitung skor dari 2 admin yang melakukan pengujian mendapatkan presentase 70% dari total presentase 100%, yang dapat disimpulkan bahwa admin setuju dengan *tampilan bug reporting* yang menarik.

2. Apakah aplikasi *bug reporting* mudah untuk digunakan?

Tabel 4.10 Hasil Pertanyaan 2 Admin

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	0	5x0 = 0
S	4	1	4x1 = 4
KS	3	1	3x1 = 3
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			7

$$P = \frac{7}{10}\ x\ 100\%$$

= 70%

Dari presentase yang didapatkan pada Tabel 4.10 dengan menghitung skor dari 2 admin yang melakukan pengujian mendapatkan presentase 70% dari total presentase 100%, yang dapat disimpulkan bahwa admin setuju dengan tampilan *bug reporting* mudah untuk digunakan.

3. Apakah proses *input*, *update* dan *delete* data tidak mengalami kendala?

Tabel 4.11 Hasil Pertanyaan 3 Admin

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	0	5x0 = 0
S	4	1	4x1 = 4
KS	3	0	3x0 = 0
TS	2	1	2x1 = 2
STS	1	0	1x0 = 0
Jumlah			6

$$P = \frac{6}{10}\ x\ 100\%$$

= 60%

Dari presentase yang didapatkan pada Tabel 4.11 dengan menghitung skor dari 2 admin yang melakukan pengujian mendapatkan presentase 60% dari total presentase 100%, yang dapat disimpulkan bahwa admin setuju dengan proses *input*, *update*, dan *delete* yang tidak memiliki kendala.

4. Apakah aplikasi *bug reporting* dapat menentukan tipe *developer* dengan akurat?

Tabel 4.12 Hasil Pertanyaan 4 Admin

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	0	5x0 = 0
S	4	0	4x0 = 0
KS	3	2	3x2 = 6
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			6

$P = \frac{6}{10}\ x\ 100\%$

= 60%

Dari presentase yang didapatkan pada Tabel 4.12 dengan menghitung skor dari 2 admin yang melakukan pengujian mendapatkan presentase 60% dari total presentase 100%, yang dapat disimpulkan bahwa admin setuju dengan sistem *bug reporting* yang menentukan tipe *developer* (*front-end* dan *back-end*) yang akurat.

5. Apakah aplikasi *bug reporting* ini membantu admin dalam memberikan tugas kepada *developer* untuk memperbaiki *bug reporting*?

Tabel 4.13 Hasil Pertanyaan 5 Admin

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	0	5x0 = 0
S	4	1	4x1 = 4
KS	3	1	3x1 = 3
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			7

$$P = \frac{7}{10}\ x\ 100\%$$

= 70%

Dari presentase yang didapatkan pada Tabel 4.13 dengan menghitung skor dari 2 admin yang melakukan pengujian mendapatkan presentase 70% dari total presentase 100%, yang dapat disimpulkan admin setuju bahwa *bug reporting* membantu admin dalam memberikan tugas kepada *developer* untuk memperbaiki *bug*.

6. Apakah aplikasi dapat memudahkan admin menambahkan *user* berdasarkan *role* dan *permission* fitur yang diberikan?

Tabel 4.14 Hasil Pertanyaan 6 Admin

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	0	5x0 = 0
S	4	1	4x1 = 4
KS	3	1	3x1 = 3
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			7

$$P = \frac{7}{10}\ x\ 100\%$$

= 70%

Dari presentase yang didapatkan pada Tabel 4.14 dengan menghitung skor dari 2 admin yang melakukan pengujian mendapatkan presentase 70% dari total presentase 100%, yang dapat disimpulkan admin setuju bahwa sistem dapat memudahkan admin menambahkan *user* berdasarkan *role* dan *permission* fitur yang diberikan.

Dari 6 pertanyaan yang telah didapatkan setelah admin melakukan pengisian kusioner, maka dapat dihasilkan total rata-rata dari presentase sebagai berikut:

$$= \frac{70\% + 70\% + 60\% + 70\% + 70\%}{6} = 67\%$$

4.2.7.2 Pengujian UAT *Quality Assurance*

Hasil presentase pertanyaan dari tingkat setiap jawaban yang didapatkan dari kusioner sebagai berikut.

1. Apakah tampilan aplikasi *bug reporting* menarik?

Tabel 4.15 Hasil Pertanyaan 1 *Quality Assurance*

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	0	5x0 = 0
S	4	5	4x5 = 20
KS	3	0	3x0 = 0
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			20

$$P = \frac{20}{25}\ x\ 100\%$$

= 80%

Dari presentase yang didapatkan pada Tabel 4.15 dengan menghitung skor dari 5 *quality assurance* yang melakukan pengujian mendapatkan presentase 80% dari total presentase 100%, yang dapat dikategorikan qa sangat setuju bahwa sistem *bug reporting* menarik.

2. Apakah aplikasi *bug reporting* ini mudah untuk digunakan

Tabel 4.16 Hasil Pertanyaan 2 *Quality Assurance*

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	1	5x1 = 5
S	4	3	4x3 = 12
KS	3	1	3x1 = 3
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			20

$P = \frac{20}{25}\ x\ 100\%$

= 80%

Dari presentase yang didapatkan pada Tabel 4.16 dengan menghitung skor dari 5 *quality assurance* yang melakukan pengujian mendapatkan presentase 80% dari total presentase 100%, yang dapat dikategorikan qa sangat setuju bahwa sistem *bug reporting* mudah untuk digunakan.

3. Apakah proses *input, update reporting* tidak mengalami kendala?

Tabel 4.17 Hasil Pertanyaan 3 *Quality Assurance*

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	1	5x1 = 5
S	4	2	4x2 = 8
KS	3	1	3x1 = 3
TS	2	0	2x0 = 0
STS	1	1	1x1 = 1
Jumlah			17

$$P = \frac{17}{25}\ x\ 100\%$$

= 68%

Dari presentase yang didapatkan pada Tabel 4.17 dengan menghitung skor dari 5 *quality assurance* yang melakukan pengujian mendapatkan presentase 68% dari total presentase 100%, yang dapat dikategorikan bahwa qa setuju proses *input, update reporting* tidak mengalami kendala.

4. Apakah aplikasi *bug reporting* dapat menentukan tipe *developer* dengan akurat?

Tabel 4. 18 Hasil Pertanyaan 4 *Quality Assurance*

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	0	5x0 = 0
S	4	3	4x3 = 12
KS	3	2	3x2 = 6
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			18

$$P = \frac{18}{25}\ x\ 100\%$$

= 72%

Dari presentase yang didapatkan pada Tabel 4.18 dengan menghitung skor dari 5 *quality assurance* yang melakukan pengujian mendapatkan presentase 72% dari total presentase 100%, yang dapat dikategorikan bahwa qa setuju *bug reporting* dapat menentukan tipe *developer* dengan akurat.

Dari 4 pertanyaan yang telah didapatkan setelah *quality assurance* (qa) melakukan pengisian kusioner, maka dapat dihasilkan total rata-rata dari presentase sebagai berikut:

$$= \frac{80\% + 80\% + 68\% + 72\%}{4} = 75\%$$

4.2.7.3 Pengujian UAT *Developer*

Hasil presentase pertanyaan dari tingkat setiap jawaban yang didapatkan dari kusioner sebagai berikut.

1. Apakah tampilkan aplikasi *bug reporting* ini menarik?

Tabel 4.19 Hasil Pertanyaan 1 *Developer*

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	1	5x1 = 5
S	4	3	4x3 = 12
KS	3	1	3x1 = 3
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			16

$$P = \frac{20}{25}\ x\ 100\%$$

= 80%

Dari presentase yang didapatkan pada Tabel 4.19 dengan menghitung skor dari 5 *developer (front-end* dan *back-end)* yang melakukan pengujian mendapatkan presentase 80% dari total presentase 100%, yang dapat dikategorikan *developer* sangat setuju bahwa tampilkan aplikasi *bug reporting* ini menarik.

2. Apakah aplikasi *bug reporting* ini mudah untuk digunakan?

Tabel 4.20 Hasil Pertanyaan 2 *Developer*

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	2	5x2 = 10
S	4	1	4x1 = 4
KS	3	2	3x2 = 6
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			15

$$P = \frac{20}{25}\ x\ 100\%$$

= 80%

Dari presentase yang didapatkan pada Tabel 4.20 dengan menghitung skor dari 5 *developer (front-end* dan *back-end)* yang melakukan pengujian mendapatkan presentase 80% dari total presentase 100%, yang dapat dikategorikan *developer* sangat setuju bahwa *bug reporting* ini mudah untuk digunakan.

3. Apakah proses *update* status, estimasi, *actual* dan komen tidak mengalami kendala?

Tabel 4.21 Hasil Pertanyaan 3 *Developer*

Skala Jawaban	Skor	Frekuensi	Presentase
SS	5	2	5x2 = 10
S	4	1	4x1 = 4
KS	3	2	3x2 = 6
TS	2	0	2x0 = 0
STS	1	0	1x0 = 0
Jumlah			15

$$P = \frac{20}{25}\ x\ 100\%$$

= 80%

Dari presentase yang didapatkan pada Tabel 4.21 dengan menghitung skor dari 5 *developer (front-end* dan *back-end)* yang melakukan pengujian mendapatkan presentase 80% dari total presentase 100%, yang dapat dikategorikan *developer* sangat setuju bahwa proses *update* status, eta, *actual*, dan komen tidak mengalami kendala.

Dari 3 pertanyaan yang telah didapatkan setelah developer (*front-end* dan *back-end*) melakukan pengisian kusioner, maka dapat dihasilkan total rata-rata dari presentase sebagai berikut:

$$= \frac{80\% + 80\% + 80\%}{3} = 80\%$$

4.2.7.4 Kesimpulan *User Acceptance Testing*

Berdasarkan hasil pengujian UAT yang telah dilakukan oleh user, diperoleh presentase keberhasilan masing-masing peran dalam menguji sistem. Hasilnya menunjukkan bahwa pengguna admin memiliki tingkat keberhasilan sebesar 67%, sementara itu pengguna qa mencapai tingkat keberhasilan 75%, pengguna developer (*front-end* dan *back-end*) mencatat tingkat keberhasilan tertinggi yaitu 80%. Dari hasil ini dapat disimpulkan bahwa pengujian sistem telah dilakukan dengan cara konsisten, setiap pengguna sistem menunjukkan kemampuan mereka dalam mengidentifikasi dan mengatasi potensi masalah. Pengujian sistem oleh qa mencatat hasil yang cukup baik dalam memastikan kualitas dan fungsionalitas yang dimiliki oleh sistem secara optimal. Meskipun peran admin memiliki tingkat keberhasilan yang lebih rendah dibandingkan dengan peran lainnya, akan tetapi mereka memberikan kontribusi yang cukup penting dalam memberikan masukan dan validasi dalam UAT. Maka dapat disimpulkan dari pengujian UAT menghasilkan kesiapan sistem dan memastikan bahwa sistem telah siap untuk digunakan dengan tingkat kualitas yang cukup baik dengan rata-rata pengujian diatas 60%.