BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil

Pada bagian hasil Rancang Bangun sistem informasi geografis menggunakan metode scrum beberapa analisis kebutuhan sistem dan menghasilkan product *backlog* sebanyak 14 *backlog* yang harus dikerjakan oleh peneliti. Berikut merupakan daftar product *backlog* yang berhasil disusun.

Tabel 4. 1 *Product backlog*

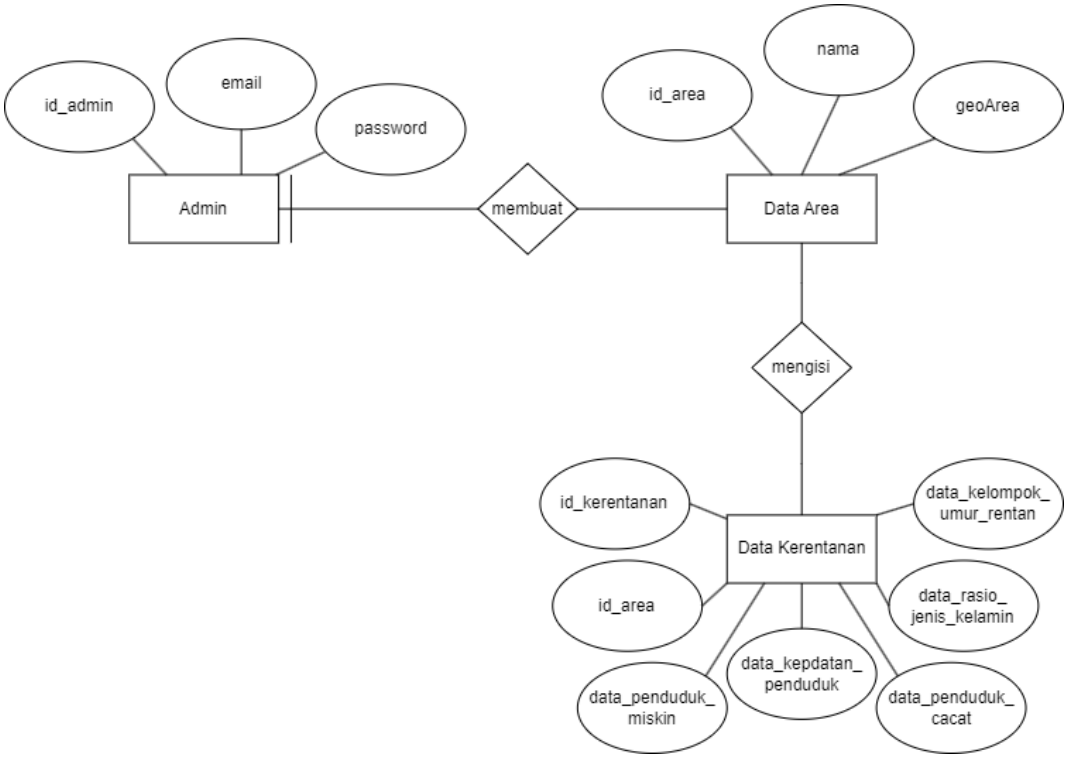
No	Product *Backlog*	Prioritas
1	Merancang *Database*	Sedang
2	Instalasi *Auth* untuk *admin* dengan *laravel UI*	Tinggi
3	Membuat Halaman utama / *Landing Page*	Sedang
4	Membuat Halaman Detail pemetaan	Tinggi
5	Membuat Halaman *Dashboard Admin*	Sedang
6	Instalasi *Datatable* untuk area dan data kerentanan	Sedang
7	Membuat Halaman Kelola data area	Sedang
8	Membuat Halaman Kelola data kerentanan	Tinggi
9	Membuat *model* dan *migration* dari *design database*	Sedang
10	Membuat *Controller* dan *View dashboard – Admin*	Sedang
11	Membuat *Controller* dan *View* Area – *Admin*	Sedang
12	Membuat *Controller* dan *View* Data kerentanan – *Admin*	Tinggi
13	Memasukan data area	Rendah
14	Memasukan data kerentanan	Rendah

Ketika semua analisis kebutuhan sistem sudah ada dan juga kebutuhan fitur sudah terdaftar di *backlog* peneliti mulai melakukan pengembangan aplikasi dengan melakukan perulangan atau *Sprint* sebanyak dua kali.

4.1.1 Analisis Kebutuhan Sistem

1. ERD (*Entity Relationship Diagram*)

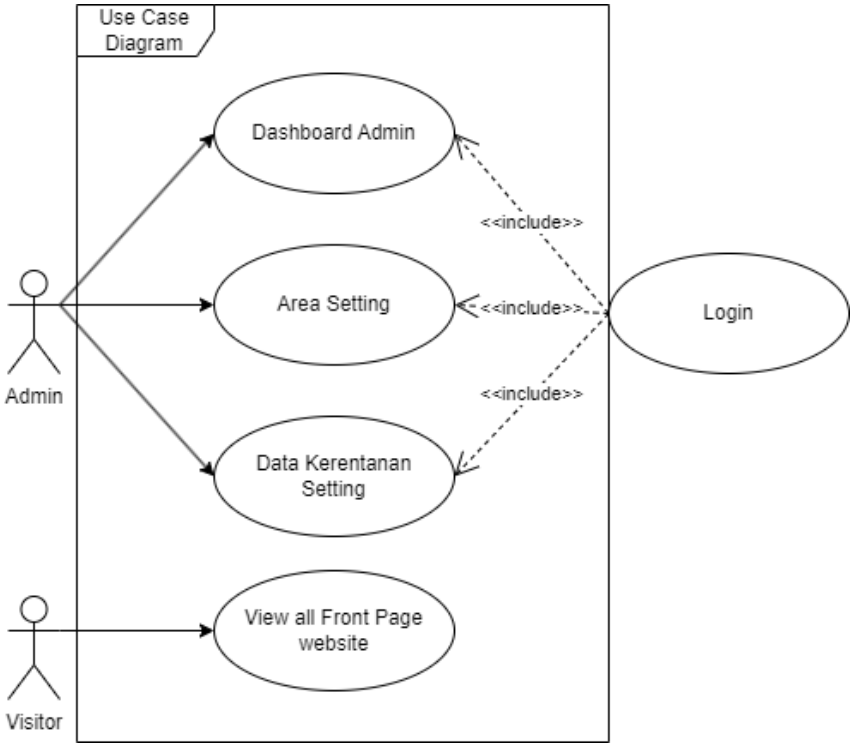
ERD untuk sistem informasi geografis berhasil dibuat dengan melibatkan beberapa entitas seperi admin. Seperti gambar 4.1 entitas admin dapat membuat data area dan memiliki relasi ke data kerentanan untuk mengisi isi data kerentanan dari tabel data area.



Gambar 4. 1. ERD

2. Membuat *Usecase Diagram*

Use Case Diagram merupakan salah satu jenis diagram dalam UML yang digunakan untuk menggambarkan interaksi antara sistem dan aktor-aktor yang terlibat dalam penggunaan sistem. Berikut adalah *use case diagram* pada sistem informasi geografis kerentanan sosial kabupaten Banyumas berbasis *website.*



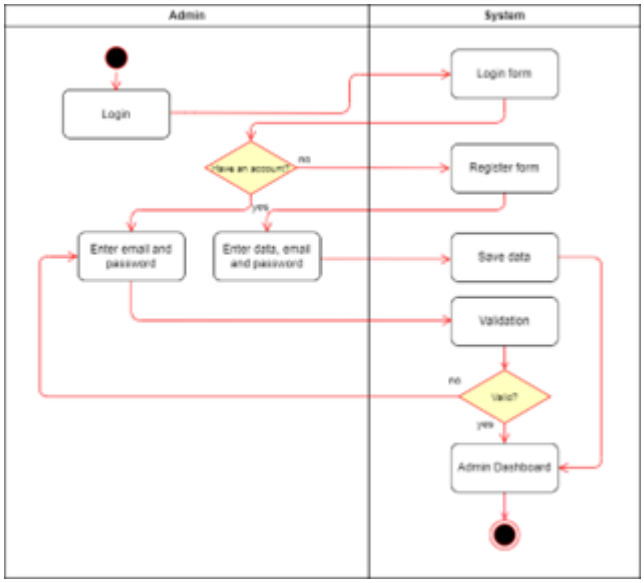
Gambar 4. 2 *Use Case* diagram

Berdasarkan *Use Case* diagram yang telah dibuat seperti pada gambar diatas terdapat 2 actor, actor uama yang dapat berinteraksi dengan system yaitu admin. *Use Case* paling sedikit dimiliki oleh *visitor*, sedangkan admin memiliki *Use Case* sebanyak empat diantaanya *login*, area *setting*, dashboard *admin*, data kerentanan *setting*, dan *download setting*.

3. *Activity diagram*

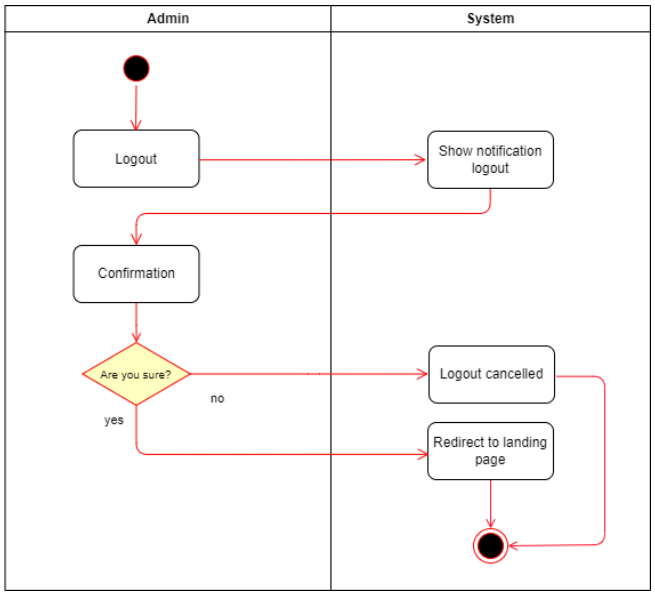
Selanjutnya, untuk merepresentasikan alur kerja atau proses pada sebuah sistem yang akan dibangun, digunakan jenis diagram perilaku yang disebut *activity diagram*. Peneliti menggunakan *activity diagram* untuk memodelkan alur kerja sistem informasi geografis untuk kerentanas sosial BPBD Banyumas berbasis *website* yang akan dibangun. Berikut ini adalah beberapa contoh *activity diagram* yang dirancang untuk mendukung proses pembuatan sistem *website* tersebut, antara lain:

1. *Login/register* admin



Gambar 4. 3 *Activity Diagram login/register* admin

2. *Activity diagram login refine*

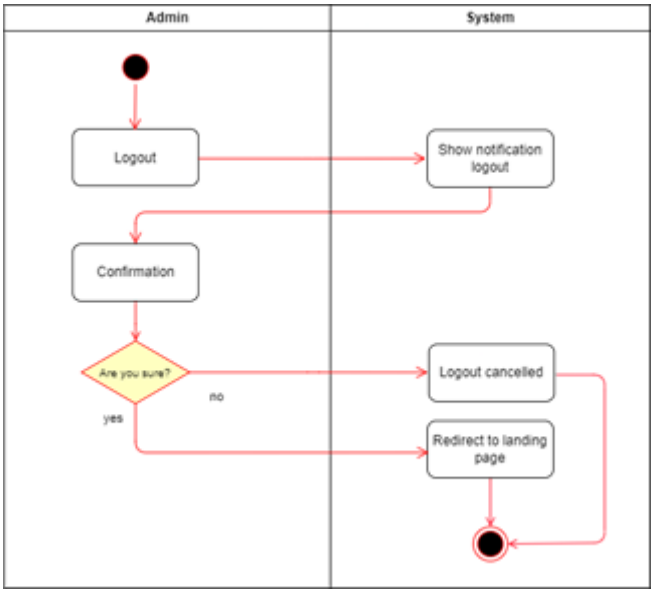


Gambar 4. 4. *Activity Diagram login refine*

Gambar 4.3 menunjukkan alur *login* untuk admin pada *dashboard admin* sebelum adanya penyesuaian dengan kebutuhan *user*. Setelah dilakukan tahapan *test*, kemudian ditemukan ketidaksesuaian dengan kebutuhan *user*, yang selanjutnya dilakukan *refine* atau perbaikan sehingga menghasilkan diagram

yang tampil pada gambar 4.4. Proses *login* akan dimulai dengan pengguna membuka halaman *login* pada aplikasi. Selanjutnya, pengguna akan diminta untuk memasukkan *username* dan *password* yang sudah terdaftar pada *database* sistem. Setelah itu, sistem akan memvalidasi data yang dimasukkan oleh pengguna untuk memastikan bahwa data yang dimasukkan sudah sesuai dengan yang terdaftar pada *database*. Jika data yang dimasukkan sudah sesuai, pengguna akan diarahkan ke halaman *dashboard admin*. Namun, jika data yang dimasukkan tidak sesuai, sistem akan memberikan pesan *error* dan meminta pengguna untuk memasukkan data yang benar.

3. *Logout Admin*



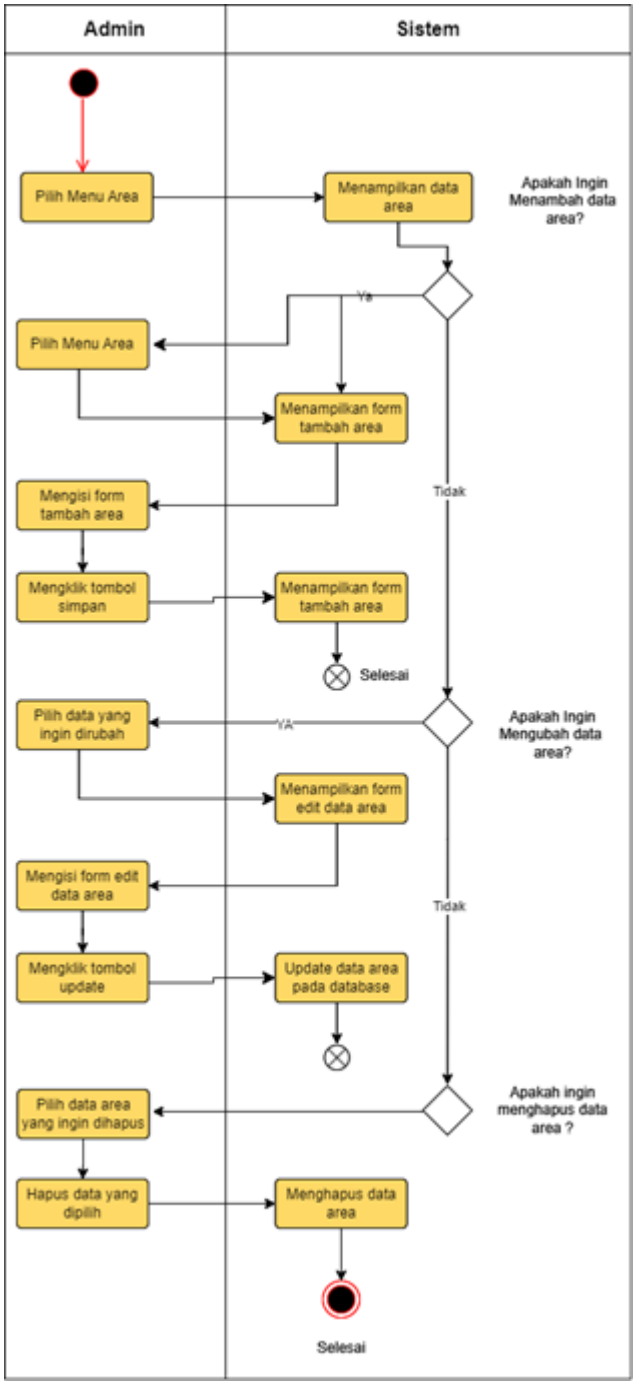
Gambar 4. 5. Logout Admin

Pada gambar 4.5, terdapat *Activity Diagram* yang menggambarkan alur proses *logout* pada aplikasi. Saat *admin* atau operator ingin keluar dari aplikasi, mereka harus melakukan proses *logout* terlebih dahulu. Proses *logout* dimulai dengan pengguna memilih opsi "*Logout*" pada halaman *dashboard* admin. Setelah opsi "*Logout*" dipilih, sistem akan menampilkan notifikasi

confirmation untuk meminta pengguna mengonfirmasi apakah mereka benar-benar ingin keluar atau tidak. Jika pengguna memilih opsi "*Yes*", sistem akan menyelesaikan proses *logout* dan menghapus *session token* yang tersimpan pada aplikasi untuk mencegah akses tanpa izin pada halaman yang sebelumnya telah diakses oleh pengguna. Namun, jika pengguna memilih opsi "*No*", sistem akan membatalkan proses *logout* dan pengguna akan tetap berada pada halaman *dashboard admin*. Setelah itu, pengguna dapat melanjutkan aktivitasnya pada aplikasi.

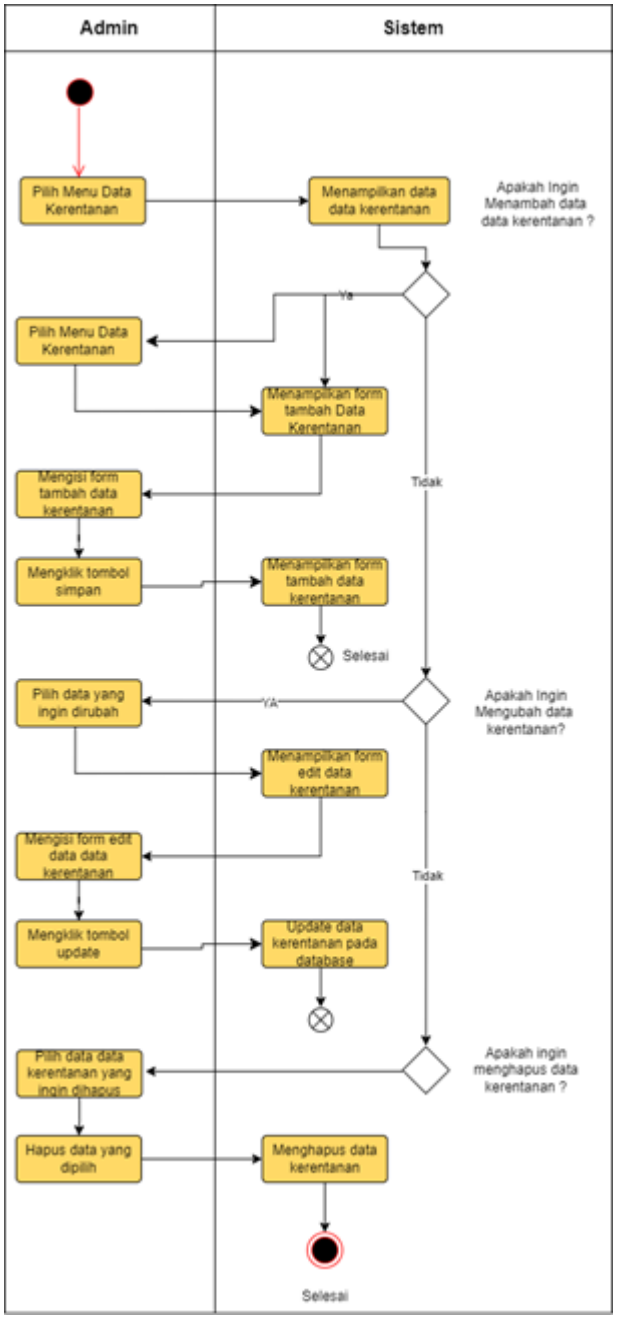
4. *Activiy diagram* Kelola Data area

Pada gambar 4.6 *diagram activity* terdapat proses *admin* mengelola data area. Alur dimulai ketika admin memilih Kelola data area dan kemudian muncul beberapa area kemudian ketika klik tambah area admin akan memasukan nama area, luas area dan dapat menentukan titik titik yang ada pada area tersebut lalu klik *save.* Kemudian ketika admin ingin mengubah data area maka klik tombol *update* kemudian isi data mana saja yang akan di *update* jika sudah maka klik update dan data yang tadi diubah akan disimpan, Terdapat juga fitur *delete* area yang memiliki notifikasi *confirmation* dengan pilihan "*yes*" atau "*no*". Sebelum melakukan penghapusan data, sistem akan melakukan validasi terlebih dahulu untuk memastikan data yang dihapus benar-benar ingin dihapus. Proses validasi data ini penting untuk memastikan integritas dan keakuratan data dalam *database*.



Gambar 4. 6. *Activity Diagram* Keola data area

5. *Activity Diagram* Kelola data kerentanan



Gambar 4. 7. *Activity Diagram* Kelola data kerentanan

Pada gambar 4.7 *diagram activity* terdapat proses admin mengelola data kerentanan. Alur dimulai ketika admin memilih Kelola data kerentanan dan kemudian muncul data area berbentuk tabel kemudian ketika klik tambah data kerentanan admin akan

memasukan nama area, jumlah penduduk, laki-laki, perempuan, kelompok usia rentan, kelompok penduduk miskin, dan kelompok cacat lalu klik *save.* Kemudian ketika admin ingin mengubah data kerentanan maka klik tombol *update* kemudian isi data mana saja yang akan di *update,* jika sudah maka klik update dan data yang tadi diubah akan disimpan, Terdapat juga fitur *delete* data kerentanan yang memiliki notifikasi *confirmation* dengan pilihan "*yes*" atau "*no*". Sebelum melakukan penghapusan data, sistem akan melakukan validasi terlebih dahulu untuk memastikan data yang dihapus benar-benar ingin dihapus. Proses validasi data ini penting untuk memastikan integritas dan keakuratan data dalam *database*.

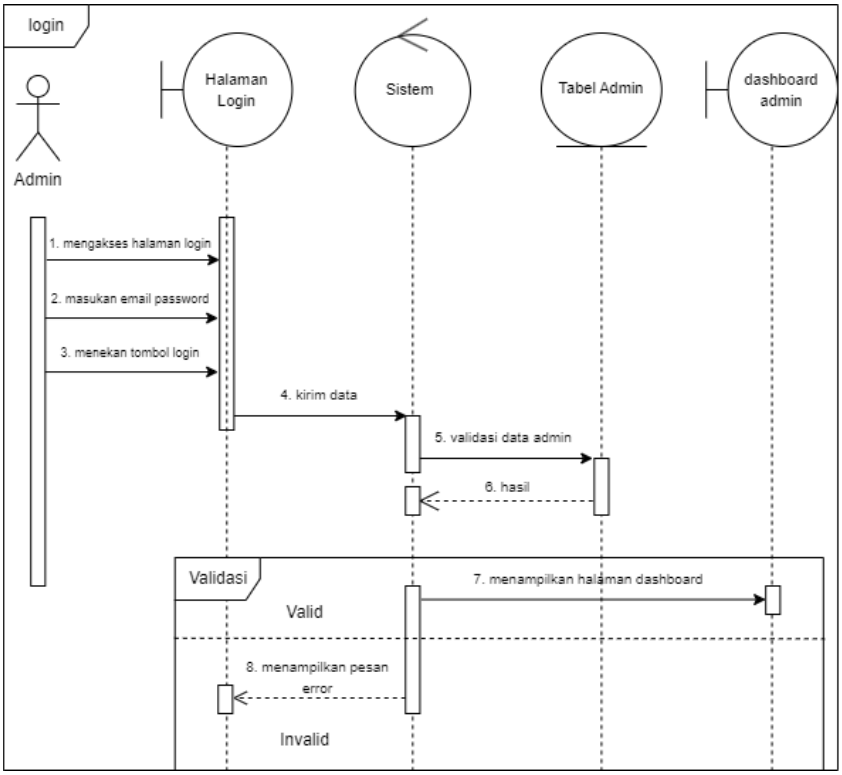
4. *Sequence diagram*

Proses selanjutnya dalam pembuatan diagram UML adalah *sequence diagram*. Diagram ini sangat berguna dalam menggambarkan urutan dan interaksi antara objek atau aktor dalam sistem, sehingga memudahkan dalam mengidentifikasi masalah dan melakukan perbaikan pada sistem. Untuk membangun sistem informasi geografis kerentanan sosial berbasis *website* BPBD Kabupaten Banyumas, beberapa rancangan *sequence diagram* diperlukan untuk menggambarkan alur kerja sistem secara detail. Adapun rancangannya sebagai berikut:

1. *Login admin*

Pada gambar 4.8 disajikan sebuah proses untuk *admin* dapat melakukan *login*. Proses dimulai dengan *admin* memasuki halaman *login* kemudian memasukan data *email* dan *password* kemudian menekan tombol *login*. Ketika tombol ditekan halaman *login* akan mengirim data *email* dan *password* ke sistem. Sistem akan melakukan validasi data dengan mencocokan data di database. Jika validasi berhasil maka sistem

akan mengalihkan admin ke halaman *dashboard admin*. Tetapi jika validasi gagal maka sistem akan mengirimkan pesan kesalahan.

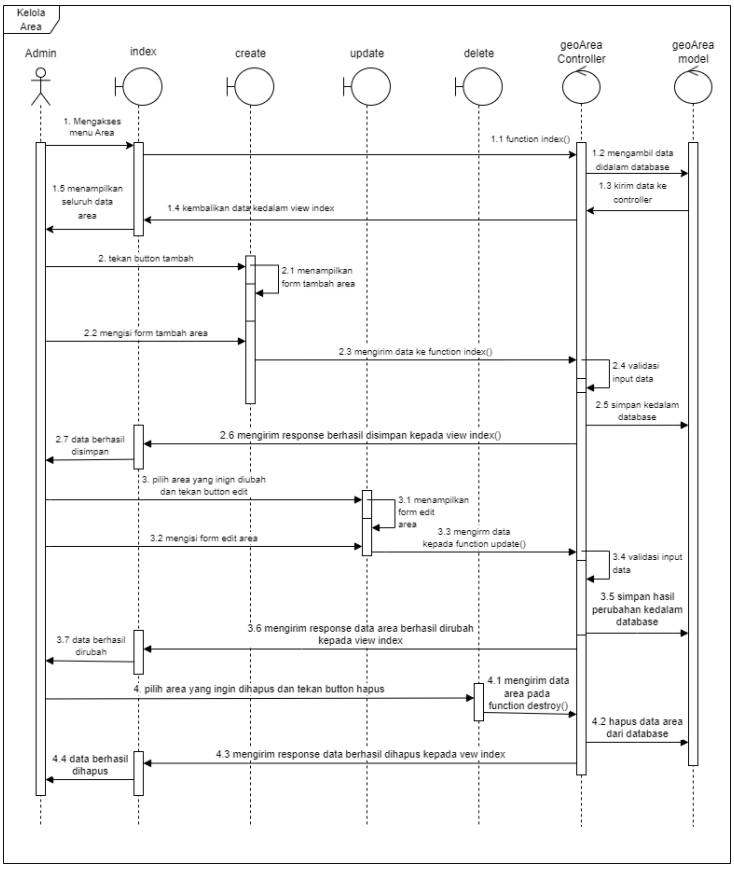


Gambar 4. 8. *Sequence diagram login* admin

2. Kelola Data Area

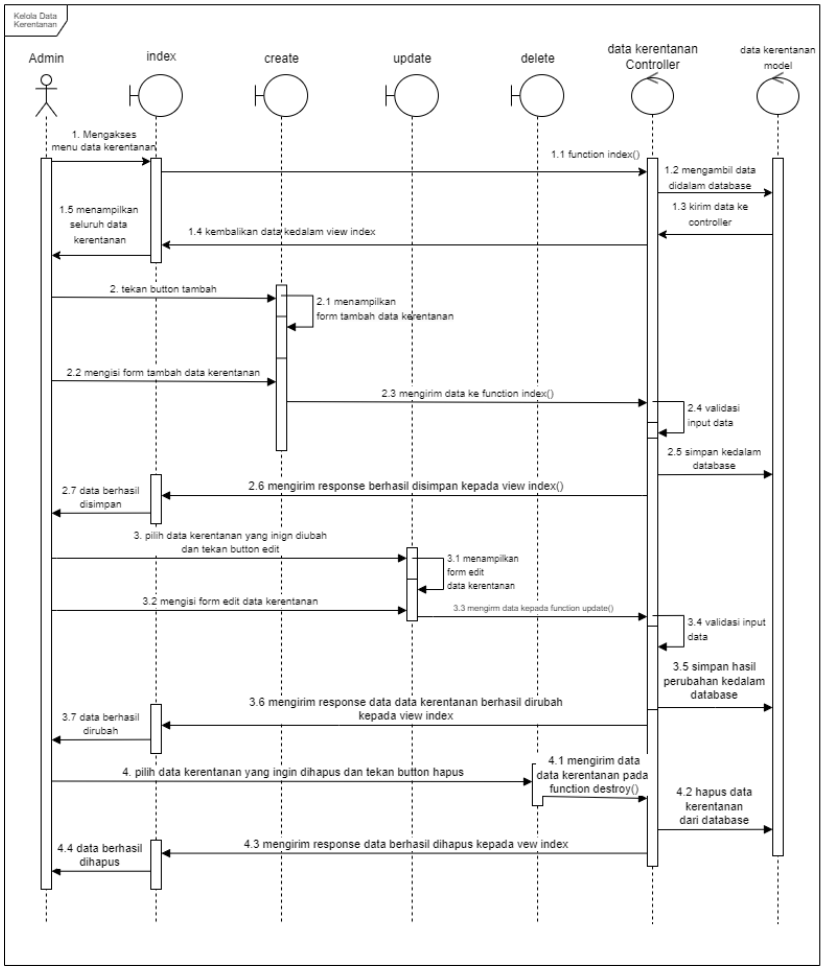
Pada gambar 4.9 menggambarkan *sequence diagram* untuk alur *admin* Kelola data area yaitu meliputi *create, read, update* dan *delete*. Pada diagram tersebut, terdaoat *actor admin* yang dapat mengakses halaman Kelola data area. Selanjutnya *admin* dapat mengakses "*add area*". Selanjutnya, *admin* dapat mengisi form yang akan divalidasi melalui *geoAreaController* sebelum disimpan ke database. Apabila data tidak sesuai dengan ketentuan sistem yang telah ditentukan, maka *process* akan menampilkan error. Ada juga bagian *update*, *admin* dapat memilih data yang akan di *update* dan kemudian diarahkan ke *form* edit. Di sana, *admin* dapat mengubah data yang telah dipilih

dengan data yang baru. Setelah data yang baru telah diisi, data tersebut akan melewati validasi oleh *geoAreaController* sebelum disimpan ke *database.* Jika validasi gagal maka *process* akan menampilkan *error* dan kembali ke *form* edit untuk diperbaiki. Namun, jika validasi berhasil, maka data akan disimpan ke dalam *database*. Selanjutnya dibagian *delete, Admin* dapat memilih data yang ingin dihapus dan memilih opsi "*delete*". Kemudian, data akan melalui proses validasi ke *geoAreaController* sebelum dihapus dari *database*. Jika proses validasi berhasil, maka data akan dihapus dari *database*. Namun, jika proses validasi gagal atau *admin* memilih untuk membatalkan proses, maka data tidak akan dihapus dari *database* dan proses akan dibatalkan. proses selesai, *admin* akan diarahkan kembali ke halaman Kelola area.



Gambar 4. 9. *Sequence diagram* data area

3. Kelola data kerentanan



Gambar 4.10. *Sequence diagram* data kerentanan

Gambar 4.10 menggambarkan diagram urutan untuk alur *admin* dalam mengelola data kerentanan. Alur ini meliputi operasi-*create, read, update,* dan *delete*. Dalam diagram ini, ada aktor admin yang memiliki akses ke halaman "Kelola Data Kerentanan." Setelahnya, *admin* dapat memilih "Tambah Data." *Admin* akan mengisi formulir, yang kemudian akan diverifikasi melalui kontroler dataKerentanan sebelum disimpan dalam basis data. Jika data tidak sesuai dengan aturan yang ditetapkan sistem, proses akan menampilkan pesan error.

Selanjutnya, terdapat tahap *update*. *Admin* memilih data yang

ingin diubah dan diarahkan ke halaman formulir pengeditan. Di sana, *admin* dapat memperbarui data yang telah dipilih dengan data baru. Setelah proses pengeditan selesai, data akan mengalami verifikasi oleh kontroler dataKerentanan sebelum disimpan dalam basis data. Jika verifikasi tidak berhasil, proses akan menampilkan pesan *error* dan kembali ke formulir pengeditan. Namun, jika verifikasi berhasil, data akan disimpan dalam basis data.

Bagian terakhir adalah tahap delete. *Admin* memilih data yang ingin dihapus dan memilih opsi "hapus." Data akan melewati proses verifikasi oleh kontroler dataKerentanan sebelum dihapus dari basis data. Jika verifikasi berhasil, data akan dihapus dari basis data. Jika verifikasi tidak berhasil atau admin memilih untuk membatalkan proses, data tidak akan dihapus dari basis data dan proses akan dibatalkan.

Setelah semua proses selesai, admin akan diarahkan kembali ke halaman "Kelola Data Kerentanan." Dengan demikian, diagram urutan ini menggambarkan langkah-langkah dalam mengelola data kerentanan oleh admin dalam sistem informasi geografis berbasis website.

4.1.2 *Sprint* 1

Pada *Sprint* 1 terdapat tujuh *backlog* atau fitur yang dikembangkan selama dua minggu. Berikut tahapan *scrum* yang dilakukan pada *Sprint* 1.

1. *Sprint planning*

Sprint 1 memiliki *timebox* selama 2 minggu dimulai dari tanggal 1 Mei 2023 sampai 14 Mei 2023 dengan jumlah *Story Poin*t sebanyak 14. Berikut merupakan detail dari *Sprint planning* pada *Sprint* 1.

Tabel 4. 2 *Sprint Planning* 1

No	*Backlog*	DOD	Prioritas	*Story Point*	*Sprint Goal*
1	Merancang *Database*	Dihasilkan rancangan *database*	Sedang	1	Berhasil merancang *database*, menginstal *auth* untuk *admin* dengan *Laravel UI*, membuat halaman utama/ *landing page*, membuat halaman detail pemetaan. Membuat halaman *Dashboard admin*, instalasi *datatable* untuk area dan data kerentanan
2	Instalasi *Auth* untuk *admin* dengan *laravel UI*	Dihasilkan autentikasi dan halaman untuk *register*, *login* dan *logout* untuk *admin*	Tinggi	3	
3	Membuat Halaman utama / *Landing Page*	Dihasilkan halaman *landing page* yang interaktif	Sedang	2	
4	Membuat Halaman Detail pemetaan	Dihasilkan halaman detail pemetaan	Tinggi	3	
5	Membuat Halaman *Dashboard Admin*	DIhasilkan halaman *Dashboard admin*	Sedang	2	
6	Instalasi *Datatable* untuk area dan data kerentanan	Dihasilkan *Datatabel* untuk data area dan data kerentanan	Sedang	2	
7	Membuat Halaman Kelola data area	Dihasilkan halaman Kelola data area untuk *admin*	Sedang	2	

2. *Daily scrum*

Pada tahap ini peneliti melakukan monitoring *backlog* seperti pada tabel 4.2 Peneliti juga melakukan pengujian pada fitur yang sudah berhasil dikerjakan seperti pada tabel 4.3.

Tabel 4. 3. *Sprint* monitoring *Sprint* 1

No	*Backlog*	Progres
1	Merancang *Database*	*Done*
2	Instalasi *Auth* untuk *admin* dengan *laravel UI*	*Done*
3	Membuat Halaman utama / *Landing Page*	*Done*
4	Membuat Halaman Detail pemetaan	*Done*
5	Membuat Halaman *Dashboard Admin*	*Done*
6	Instalasi *Datatable* untuk area dan data kerentanan	*Done*
7	Membuat Halaman Kelola data area	*Done*

3. *Sprint review*

Setelah mendekati akhir *Sprint* peneliti melakukan *Review* dengan *user*. Berikut merupakan hasil dan detail *Sprint Review* dari *Sprint* 1. peneliti mendemokan dan menjelaskan firur-fitur yang telah dikerjakan. Peneliti dan *user* mencocokan apakah fitur yang dikerjakan sudah memenuhi DOD atau belum. Proses *Review* yang dilakukan menghasilkan kesimpulan bahwa semua fitur telah memenuhi DOD dengan detail seperti pada tabel 4.4 di bawah ini.

Tabel 4. 4. *Sprint Review Sprint* 1

No	*Backlog*	Memenuhi DOD	Lolos Pengujian
1	Merancang *Database*	Ya	Tidak memerlukan pengujian

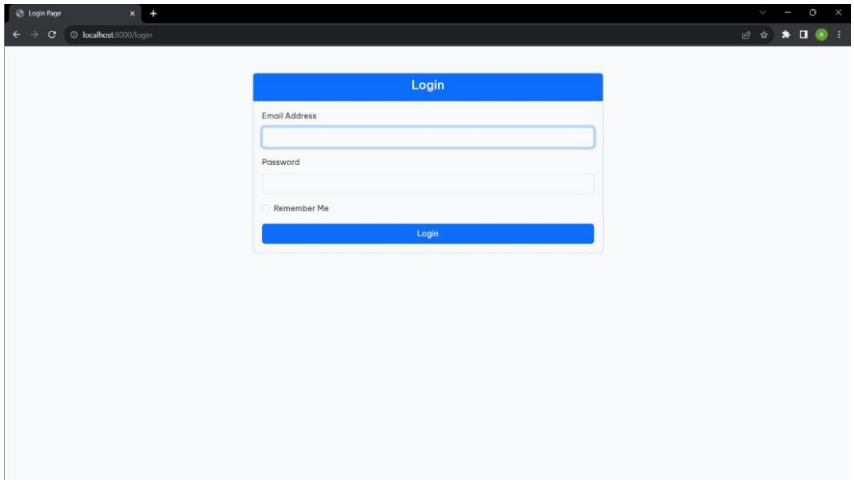
No	*Backlog*	Memenuhi DOD	Lolos Pengujian
2	Instalasi *Auth* untuk *admin* dengan *laravel UI*	Ya	Ya
3	Membuat Halaman utama / *Landing Page*	Ya	Ya
4	Membuat Halaman Detail pemetaan	Ya	Ya
5	Membuat Halaman *Dashboard Admin*	Ya	Ya
6	Instalasi *Datatable* untuk area dan data kerentanan	Ya	Ya
7	Membuat Halaman Kelola data area	Ya	Ya

a. Merancang *Database*

Merancang *database* melibatkan pembuatan struktur yang terorganisir untuk menyimpan dan mengelola data dengan efisien. Proses ini melibatkan identifikasi *entitas* (objek) yang akan disimpan, atribut yang merepresentasikan karakteristik *entitas* tersebut, serta hubungan antara entitas. Desain database juga mempertimbangkan normalisasi untuk mengurangi redundansi data dan memastikan *integritas*. Tujuan akhir dari merancang *database* adalah menciptakan kerangka yang kuat dan efisien untuk menyimpan informasi, memfasilitasi pencarian dan manipulasi data, serta mendukung kebutuhan aplikasi atau sistem yang akan menggunakannya.

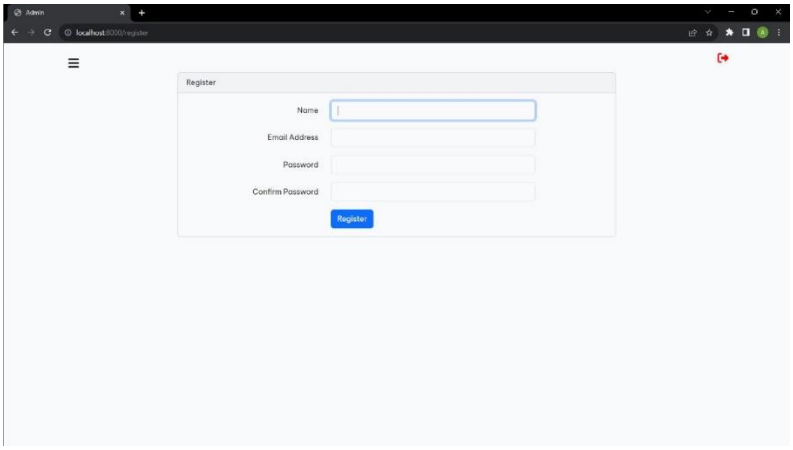
b. Instalasi Auth untuk admin dengan laravel UI

Instalasi *Auth* untuk *admin* dengan *Laravel UI* melibatkan penggunaan alat bantu *Laravel UI* untuk mengimplementasikan sistem otentikasi dalam proyek berbasis *Laravel*. Langkah-langkahnya meliputi pemasangan *Laravel UI package*, konfigurasi otentikasi untuk mengelola *register*,*login* dan *logout*, pembuatan role khusus '*admin'* dengan hak akses yang sesuai, serta integrasi tampilan dan logika pengendalian untuk halaman *login* dan *dashboard admin*. Proses ini memungkinkan penggunaan fitur autentikasi yang aman dan efisien, serta memastikan bahwa *admin* memiliki kontrol penuh terhadap bagian-bagian penting dari aplikasi.



Gambar 4. 11 Halaman *Login Admin*

Halaman login *admin* seperti pada gambar 4.11 terdiri dari kolom *email* dan *password* yang akan digunakan oleh *admin* untuk memasukan *credentials* agar dapat masuk kehalaman dashboard admin setelah mengeklik tombol masuk tombol masuk.



Gambar 4. 12 *Halaman Register*

Desain halaman *register admin* seperti pada gambar 4.12 terdiri dari kolom *name, email, password* dan *confirm password* yang akan digunakan oleh *user* untuk memasukan data untuk mendaftar sebagai *admin* setelah mengeklik tombol *register*.

c. Membuat Halaman utama / Landing Page

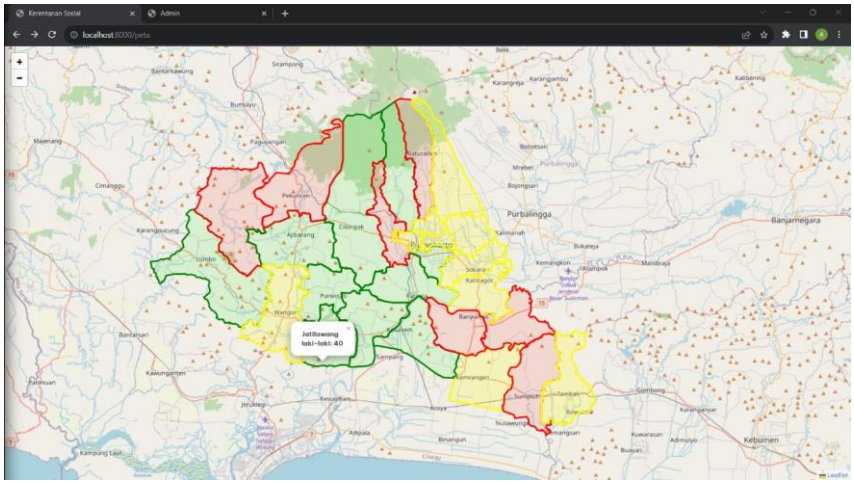
Halaman *landing page* untuk *website* sistem Informasi geografis berhasil dibuat dengan detail seperti gambar 4.11. *Landing page* memiliki tampilan simpel dan ada *button* lihat peta untuk melihat data detail pemetaan persebaran kerentanan sosial.

Beberapa penggalan kode di atas, dapat diakses secara lengkap melalui link github berikut : https://github.com/Muhammadallam12/pemetaan-kerentanan-sosial .

Gambar 4. 13 Halaman *Landing Page*

d. Membuat Halaman Detail pemetaan

Halaman Detail Pemetaan pada *website* sistem Informasi geografis berhasil dibuat dengan detail seperti pada gambar 4.14. Halaman ini menampilkan informasi mengenai data area per kecamatan, persebaran indikasi kerentanan sosial ditandai dengan warna hijau, kuning, dan merah, serta ketika di-klik akan muncul data yang berisi informasi tentang kepadatan penduduk dan kelompok rentan.



Gambar 4. 14 Halaman Detail Pemetaan

e. Membuat halaman *dashboard admin*

Gambar 4. 15 Halaman *Dashboard Admin*

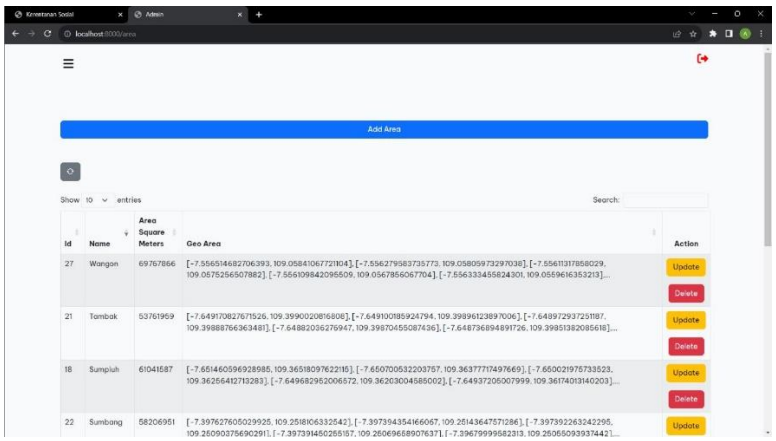
Pada gambar 4.15, halaman *dashboard admin* memiliki tampilan yang sangat simpel namun informatif. Di tengah halaman, terdapat sebuah chart yang secara visual menggambarkan data penduduk dari bulan Januari hingga Desember. Chart ini memberikan pandangan yang jelas tentang tren kenaikan atau

penurunan jumlah kerentanan sosial selama setahun. Kesederhanaan tampilan memudahkan *admin* untuk dengan cepat memahami informasi utama mengenai data penduduk dan membuat keputusan yang berdasarkan pada analisis *visual* yang disajikan oleh chart tersebut.

f. Instalasi *Datatable* untuk area dan data kerentanan

Instalasi *DataTable* untuk *admin* dan data kerentanan melibatkan penerapan alat bantu *DataTable* dalam aplikasi, memungkinkan pengorganisasian dan presentasi data area serta data kerentanan secara efisien. Proses ini mencakup integrasi *DataTable* ke dalam *framework Laravel* atau *platform* yang digunakan, pengambilan data dari sumber yang relevan, dan konfigurasi tampilan tabel yang dapat diurutkan dan dicari. Dengan demikian, *admin* dapat dengan mudah menjelajahi informasi area dan data kerentanan, sambil memberikan fleksibilitas dalam pengaturan tampilan dan interaksi dengan data yang diperlukan.

g. Membuat Halaman Kelola data area – *admin*



Gambar 4. 16 Halaman Kelola data area

Pada gambar 4.16, terdapat tampilan halaman "Kelola Data Area" yang menyajikan sebuah tabel berisi informasi tentang berbagai area. Di dalam tabel ini, *admin* memiliki kemampuan untuk melakukan operasi CRUD (*Create, Read, Update, Delete*) terhadap data area. Fitur ini memungkinkan *admin* untuk menambahkan, melihat, mengubah, dan menghapus area sesuai dengan kebutuhan. Informasi yang terdapat dalam tabel tersebut memberikan gambaran komprehensif tentang berbagai area yang terdaftar dalam sistem, sehingga admin dapat secara efektif mengelola data terkait area dengan mudah dan akurat.

4. *Sprint Retrospective*

Berikut merupakan detail *Sprint Retrospective* pada *Sprint* 1.

Tabel 4. 5. *Sprint Retrospective* 1

No	*What Went Well*	*What Didn't Go Well*	*What Can Be Imporved*
1	Semua fitur berhasil dikerjakan sebelum deadline	-	-

Pada tahap ini, *scrum master* melakukan evaluasi *Sprint 1* yang telah dilaksanakan selama dua minggu, dalam dua minggu tersebut terdapat 7 *task* atau tugas untuk *team developer*, dari tugas tersebut terdapat perencanaan dan *actual* yang sudah ditentukan dengan *story point* setiap tugasnya. Dapat disimpulkan bahwa *Sprint 1* dilaksanakan secara lebih cepat dibandingan dari perencanaan, hal ini disebabkan dari poin *actual* yang telah dikerjakan oleh *team developer*. Hal ini menjadi sebuah pencapaian bahwa *Sprint 1* telah berjalan lancar.

4.1.3 *Sprint* 2

Pada *Sprint* 2 terdapat tujuh *backlog* atau fitur yang dikembangkan selama dua minggu. Berikut tahapan *scrum* yang dilakukan pada *Sprint* 1.

1. *Sprint planning*

Tabel 4. 6. *Sprint* 2

No	*Backlog*	DOD	Prioritas	*Story Point*	*Sprint Goal*
1	Membuat Halaman Kelola data kerentanan	Dihasilkan Halaman Kelola data kerentanan	Tinggi	3	Menghasilkan halaman Kelola data kerentanan, menghasilkan *model* dan *migration database*, menghasilkan fitur pada *dashboard admin*, menghasilkan fitur *CRUD* pada Kelola data area, menghasilkan fitur *CRUD* pada Kelola data kerentanan, berhasil memasukan data area dan data kerentanan
2	Membuat *model* dan *migration* dari *design database*	Dihasilkan *model* dan *migration* sesuai dengan ERD	Sedang	2	
3	Membuat *Controller* dan *View dashboard* – *Admin*	Dihasilkan fungsi untuk menampilkan chart di halaman area	Sedang	2	
4	Membuat *Controller* dan *View* Area – *Admin*	Dihasilkan fitur *create, read, update* dan *delete* untuk Kelola area	Sedang	2	
5	Membuat *Controller* dan *View* Data kerentanan – *Admin*	Dihasilkan fitur *create, read, update* dan *delete* untuk Kelola data kerentanan	Tinggi	3	

No	*Backlog*	DOD	Prioritas	*Story Point*	*Sprint Goal*
6	Memasukan data area	Berhasil memasukan data area berdasarkan data yang sudah ada	Rendah	1	
7	Memasukan data kerentanan	Berhasil memasukan data kerentanan berdasarkan data yang sudah ada per kecamatan	Rendah	1	

2. Daily scrum

Pada tahap ini, peneliti melakukan pemantauan terhadap *backlog* seperti yang terlihat pada Tabel 4.2. Selain itu, peneliti juga melakukan pengecekan terhadap fitur-fitur yang sudah berhasil dikerjakan, sebagaimana yang ditunjukkan pada Tabel 4.3. Tujuan dari tahap ini adalah untuk memastikan bahwa pekerjaan yang tercatat dalam *backlog* telah selesai atau belum. Proses ini penting untuk mengontrol kemajuan proyek dan memastikan bahwa semua komponen yang diperlukan telah diselesaikan dengan baik.

Tabel 4. 7 *Sprint monitoring Sprint* 2

No	*Backlog*	Progres
1	Membuat Halaman Kelola data kerentanan	*Done*
2	Membuat *model* dan *migration* dari *design database*	*Done*

3	Membuat *Controller* dan *View dashboard – Admin*	*Done*
4	Membuat *Controller* dan *View* Area – *Admin*	*Done*
5	Membuat *Controller* dan *View* Data kerentanan – *Admin*	*Done*
6	Memasukan data area	*Done*
7	Memasukan data kerentanan	*Done*

3. Sprint Review

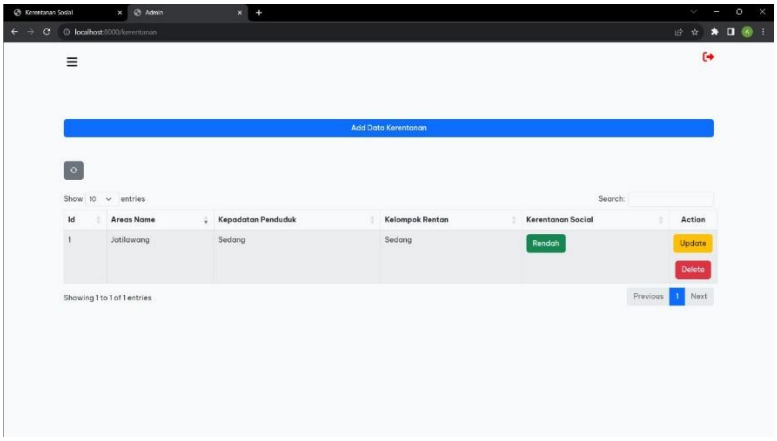
Setelah mendekati akhir *Sprint*, peneliti melaksanakan *Review* dengan user. Berikut merupakan hasil dan detail *Sprint Review* dari *Sprint* 2: peneliti menjelaskan dan menjelaskan fitur-fitur yang telah dikerjakan. Peneliti dan *user* membandingkan apakah fitur yang dikerjakan sudah memenuhi *DOD* atau belum dan apakah sudah lolos pengujian atau tidak. Proses *Review* yang dilakukan menghasilkan kesimpulan bahwa sebagian besar fitur telah memenuhi DOD dengan rincian seperti yang terlihat pada Tabel 4.4 di bawah ini.

Tabel 4. 8 *Sprint review Sprint* 2

No	*Backlog*	Memenuhi DOD	Lolos Pengujian
1	Membuat Halaman Kelola data kerentanan	Ya	Ya
2	Membuat *model* dan *migration* dari *design database*	Ya	Ya
3	Membuat *Controller* dan *View dashboard – Admin*	Ya	Ya
4	Membuat *Controller* dan *View* Area – *Admin*	Ya	Ya

5	Membuat *Controller* dan *View* Data kerentanan – *Admin*	Ya	Ya
6	Memasukan data area	Ya	Tidak memerlukan pengujian
7	Memasukan data kerentanan	Ya	Tidak memerlukan pengujian

a. Membuat Halaman Kelola data kerentanan



Gambar 4. 17 Halaman Kelola data kerentanan

Pada gambar 4.17, terdapat tampilan halaman "Kelola Data Kerentanan" yang menampilkan sebuah tabel berisi informasi mengenai kerentanan yang ada. Di dalam tabel ini, admin memiliki kemampuan untuk melakukan operasi CRUD (*Create*, *Read*, *Update*, *Delete*) terhadap data kerentanan. Fitur ini memungkinkan *admin* untuk menambahkan, melihat, mengubah, dan menghapus entri-entri kerentanan sesuai dengan kebutuhan. Informasi yang tertera dalam tabel ini memberikan gambaran menyeluruh tentang kerentanan yang terdaftar dalam sistem, memberikan admin kemampuan untuk efektif mengelola data kerentanan dengan mudah dan akurat.

b. Membuat *model* dan *migration* dari *design database*

Langkah ini melibatkan pembuatan *model* dan *migrasi* berdasarkan *desain database* yang telah dirancang sebelumnya. Dalam proses ini, *model* akan mewakili entitas dan hubungan antar entitas dalam *database*, sedangkan *migrasi* digunakan untuk menciptakan skema tabel di basis data. Dengan menciptakan *model* dan *migrasi* yang sesuai dengan *desain database*, proyek dapat memiliki struktur yang terorganisir, yang memungkinkan aplikasi untuk berinteraksi dengan basis data dengan cara yang efisien dan konsisten sesuai dengan kebutuhan.

Beikut kode *model* dan *migration* merujuk pada implementasi dari *model* dan *migrasi* dalam *framework* pengembangan seperti *Laravel* :

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];
```

```
    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];
}
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
       Schema::create('users', function (Blueprint $table)
```

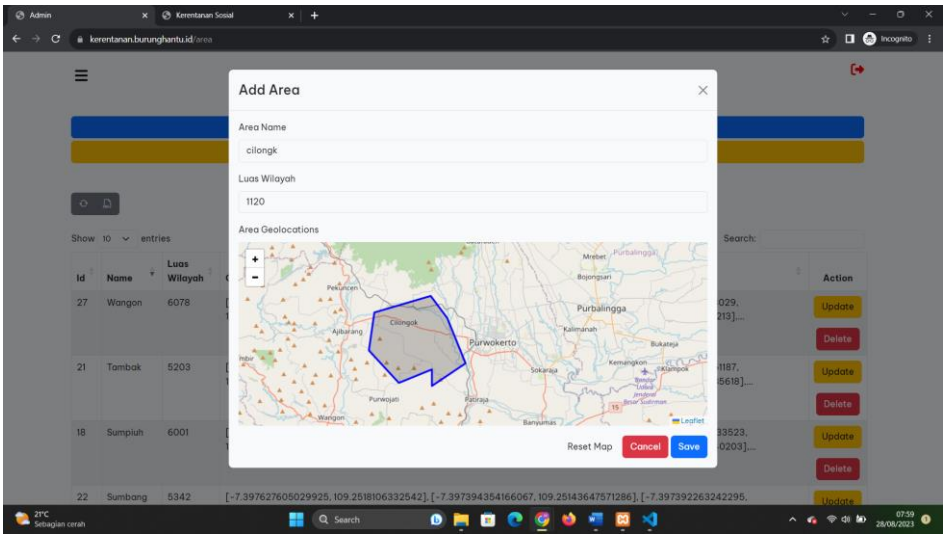
```
{
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
                  $table->timestamp('email_verified_at')-
>nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
};
```

c. Membuat Controller dan View dashboard – Admin

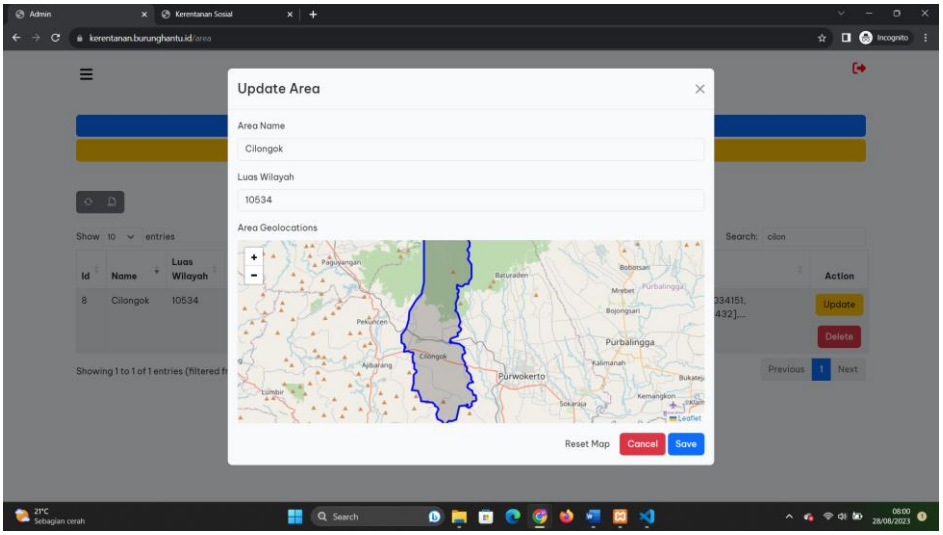
Pada tahap ini, dilakukan pembuatan *Controller* dan *View* khusus untuk halaman *dashboard admin*. *Controller* berfungsi sebagai penghubung antara tampilan (*View*) dan logika aplikasi. Dalam hal ini, *Controller* akan mengatur bagaimana data ditampilkan di *View dashboard admin*. Sementara itu, *View* akan menampilkan informasi secara visual kepada *admin*, seperti grafik data penduduk rentan. Proses pembuatan *Controller* dan *View* ini memungkinkan *admin* untuk melihat dan mengelola informasi penting dengan cara yang lebih terstruktur dan interaktif.

d. Membuat Controller dan View Area – Admin

Pada tahap ini, dilakukan pembuatan *Controller* dan *View* khusus untuk halaman "Kelola Data Area" oleh *admin. Controller* bertindak sebagai perantara antara tampilan (*View*) dan logika aplikasi. Dalam hal ini, *Controller* mengelola bagaimana data area ditampilkan dan diolah di *View*. Sementara itu, *View* akan menampilkan data area dalam format yang mudah dimengerti, sehingga *admin* dapat melakukan operasi *Create, Read, Update,* dan *Delete* (CRUD) terhadap *entri-entri* area. Proses ini memungkinkan *admin* untuk mengelola informasi area dengan efisien dan akurat melalui antarmuka yang disediakan, berikut gambar 4.18tambah area dan gambar 4.19 gambar update area



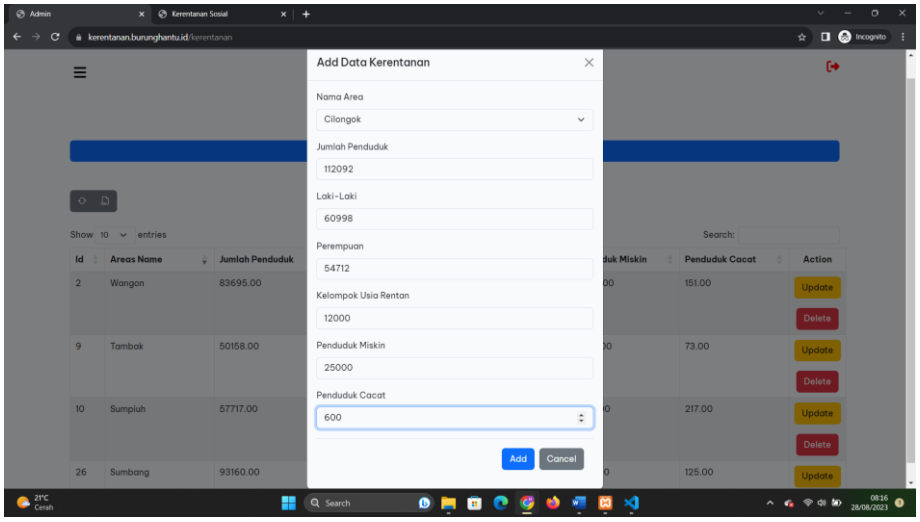
Gambar 4. 18 Halaman tambah area



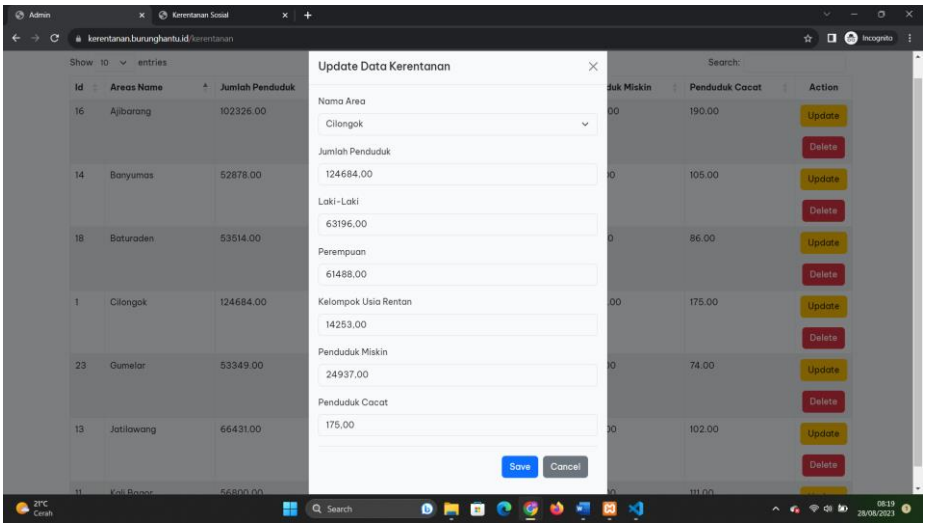
Gambar 4. 19 Ubah area

e. Membuat *Controller* dan *View* Data kerentanan – *Admin*

Pada tahap ini, dilakukan pembuatan *Controller* dan *View* khusus untuk halaman "Kelola Data Kerentanan" oleh *admin*. *Controller* berperan sebagai penghubung antara tampilan (*View*) dan logika aplikasi. Dalam hal ini, *Controller* mengatur bagaimana data kerentanan ditampilkan dan diolah di *View*. Sementara itu, *View* akan menampilkan informasi kerentanan dalam format yang mudah dimengerti, memungkinkan admin untuk melakukan operasi *Create*, *Read, Update*, dan *Delete* (CRUD) pada *entri-entri* kerentanan. Proses ini memungkinkan *admin* untuk mengelola data kerentanan dengan efisien dan akurat melalui antarmuka yang disediakan. Gambar 4.20 menunjukkan contoh halaman tambah data kerentanan, sementara Gambar 4.21 menggambarkan tampilan untuk mengubah data kerentanan.



Gambar 4. 20 Tambah data kerentanan



Gambar 4. 21 Ubah data kerentanan

f. Memasukan data area

Pada tahap ini, *admin* dapat memasukkan data area sesuai dengan bahan data yang diambil dari BPS Kabupaten Banyumas. *Admin* akan menggunakan antarmuka yang disediakan untuk menginput informasi area ke dalam sistem. Proses ini memungkinkan *admin* untuk mengelola data area dengan akurat, memastikan bahwa informasi yang dimasukkan sesuai dengan sumber data resmi dari BPS Kabupaten Banyumas. Setelah memasukkan data, sistem akan memproses dan menyimpannya dalam *database*.

g. Memasukkan data kerentanan

Pada tahap ini, *admin* dapat memasukkan data kerentanan sosial sesuai dengan bahan data yang diambil dari BPS Kabupaten Banyumas. *Admin* akan menggunakan antarmuka yang disediakan untuk menginput informasi kerentanan sosial ke dalam sistem. Proses ini memungkinkan admin untuk mengelola data kerentanan sosial dengan akurat, memastikan bahwa informasi yang dimasukkan sesuai dengan sumber data resmi dari BPS Kabupaten Banyumas. Setelah memasukkan data, sistem akan memproses dan menyimpannya dalam *database*.

4. Sprint retrospective

Berikut tabel 4.9 detail *Sprint Retrospective* pada *Sprint* 2.

Tabel 4. 9. *Sprint Retrospective* 2

No	*What Went Well*	*What Didn't Go Well*	*What Can Be Imporved*
1	Semua fitur berhasil dikerjakan sebelum deadline	-	-

Pada tahap ini, *scrum master* melakukan evaluasi *Sprint 2* yang telah dilaksanakan selama dua minggu, dalam dua minggu tersebut terdapat 7 *task* atau tugas untuk *team developer*, dari tugas tersebut terdapat perencanaan dan *actual* yang sudah ditentukan dengan *story point* setiap tugasnya. Dapat disimpulkan bahwa *Sprint 2* dilaksanakan secara lebih cepat dibandingan dari perencanaan, hal ini disebabkan dari poin *actual* yang telah dikerjakan oleh *team developer*. Hal ini menjadi sebuah pencapaian bahwa *Sprint 2* telah berjalan lancar.

4.1.4 Pengujian *Blackbox testing*

Dari hasil pembuatan *website* sistem informasi geografis ini menggunakan

pengujian *Blackbox Testing* yang dimana disajikan dalam Tabel 4.10, 4.11, 4.12 berikut ini:

Tabel 4. 10. testing login

No	Skenario Pengujian	*Test Case*	Hasil yang diharapkan	Status
ID 01	Mengosongkan *Email* dan *Password* lalu menekan tombol *submit*	*Email*: (kosong) *Password*: (kosong)	Menampilkan *error response email* dan *password* wajib disi	*Pass*
ID 02	Mengisi *email* dan *password* lalu menekan tombol *submit*	*Email*: (admin@admin.com) *Password*: (123)	Menampilkan *error response auth failed*	*Pass*
ID 03	Mengisi *email* dan *password* lalu menekan tombol *submit*	*Email*: (admin@admin.com) *Password*: (admin123)	Menampilkan halaman *dashboard* dikarenakan *email* dan *password* benar	*Pass*

Tabel 4. 11. testing menu area

No	Skenario Pengujian	*Test Case*	Hasil yang diharapkan	Status
ID 04	Memilih menu area	-	Menampilkan data seluruh area	*Pass*
ID 05	Menekan tombol *create* pada halaman area	-	Menampilkan halaman create yang berisi form pengisian data baru area	*Pass*
ID 06	Mengosongkan *input* nama area, luas wilayah dan geoarea pada halaman *create*	Nama area: dikosongkan, Geoarea : dikosongkan	Menampilkan *error response* bahwa nama area, luas wilayah dan geoarea belum diisi	*Pass*
ID 07	Mengisi *input* nama area dan geoarea pada halaman *create*	Nama area: Cilongok	Menampilkan *response success* bahwa berhasil disimpan dalam *database*	*Pass*
ID 08	Menekan tombol hapus pada halaman area	-	Menampilkan *response success* bahwa data area yang dipilih berhasil dihapus	*Pass*

Tabel 4. 12. Testing menu data kerentanan

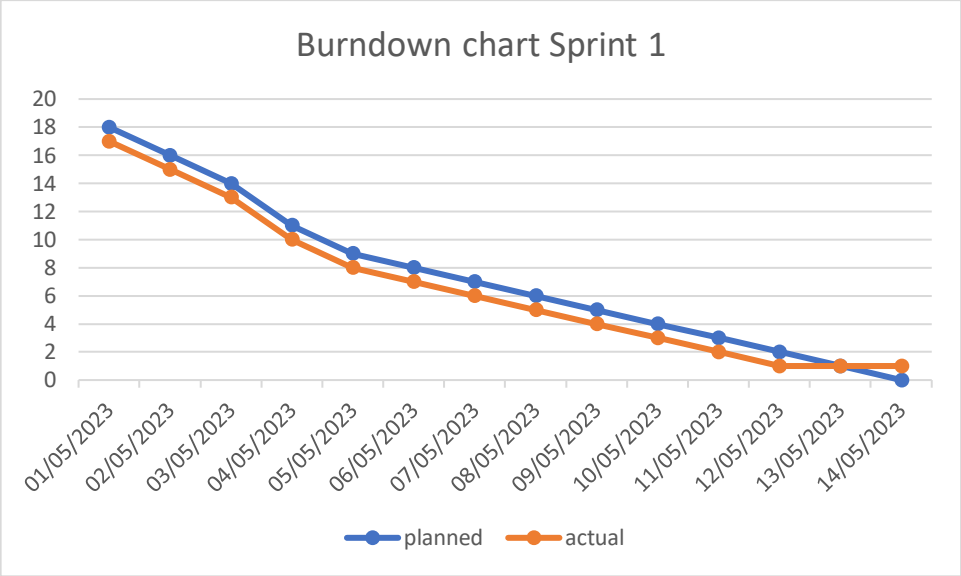
No	Skenario Pengujian	*Test Case*	Hasil yang diharapkan	Status
ID 09	Memilih menu data kerentanan	-	Menampilkan data seluruh data kerentanan	*Pass*
ID 10	Menekan tombol *create* pada halaman data kerentanan	-	Menampilkan halaman create yang berisi form pengisian data baru data kerentanan	*Pass*
ID 11	Mengosongkan seluruh *input* pada halaman *create* data kerentanan lalu menekan tombol *submit*	Nama area: dikosongkan, Jumlah penduduk: Dikosongkan Laki laki: Dikosongkan perempuan: Dikosongkan Umur rentan: dikosongkan Penduduk miskin : dikosongkan Penduduk cacat : dikosongkan	Menampilkan *error response* bahwa nama area, jumlah penduduk, laki laki, Perempuan, umur rentan, penduduk miskin, penduduk cacat belum diisi	*Pass*
ID 12	Mengisi seluruh *input* pada halaman *create* data kerentanan lalu menekan tombol *submit*	Nama area: cilongok, Jumlah penduduk: 124684, Laki laki: 63196, perempuan: 61488, Umur rentan: 14253, Penduduk miskin : 24937, Penduduk cacat : 175	Menampilkan *response success* bahwa berhasil disimpan dalam *database*	*Pass*
ID 13	Menekan tombol hapus pada halaman data kerentanan	-	Menampilkan *response success* bahwa data kerentanan yang dipilih berhasil dihapus	*Pass*

4.2 Pembahasan

Proses *Sprint* yang dilakukan sebanyak dua kali telah berhasil diselesaikan dan menghasilkan produk sebagai *output* berupa *desain sistem*, dan *website* informasi geografis. Selain itu, dilakukan pembahasan terkait setiap *Sprint* yang telah berjalan dan juga pelaksanaan pengujian yang telah dilakukan. Proses ini mencakup evaluasi mendalam mengenai kemajuan dan hasil dari masing-masing *Sprint*, serta pengujian fungsionalitas dan kualitas untuk

memastikan bahwa produk yang dihasilkan sesuai dengan harapan dan memenuhi standar yang ditetapkan.

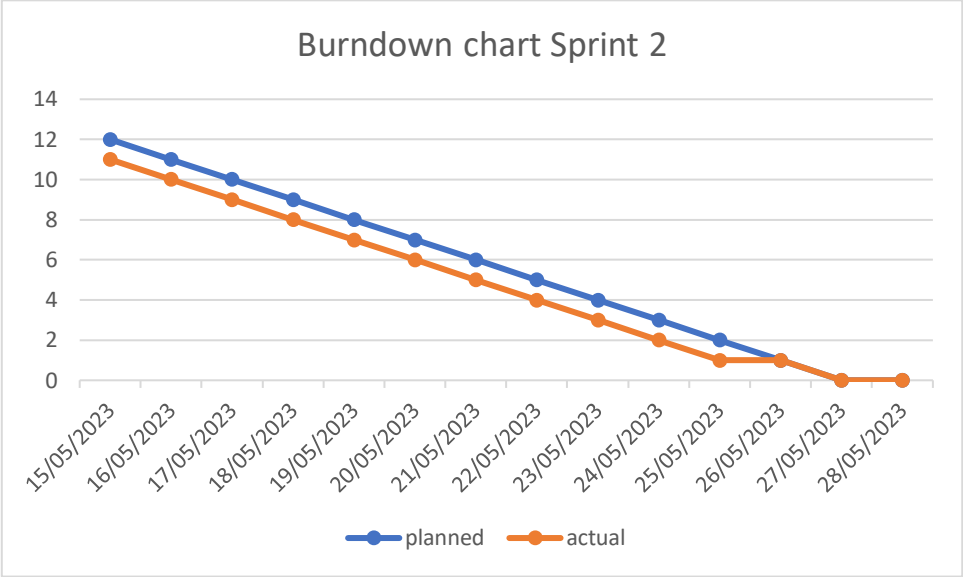
4.2.1 *Sprint* 1



Gambar 4. 22 *Burndown chart Sprint* 1

Sprint 1 terdiri dari 9 *Story Poin*t yang dimulai dari tanggal 1 Mei 2023 sampai tanggal 09 Mei 2023. Gambar 4.22 merupakan burndown chart *Sprint* 1 yang memberikan informasi bahwa pengerjaan fitur selesai sebelum waktu perkiraan yaitu selesai pada tanggal 14 Maret 2023 dengan waktu perkiraan selesai pada 09 Maret 2023.

4.2.2 *Sprint 2*



Gambar 4. 23 *Burndown chart Sprint* 2

Sprint 2 terdiri dari 12 *Story Poin*t yang dimulai dari tanggal 15 Mei 2023 sampai dengan tanggal 28 Mei 2023. Gambar 4.23 merupakan burndown chart *Sprint* 2 yang memberikan informasi bahwa pengerjaan fitur selesai sebelum waktu perkiraan yaitu selesai pada tanggal 28 Mei 2023 dengan waktu perkiraan selesai pada 26 Mei 2023.

4.2.3 *Pengujian Blackbox*

Setelah peneliti memperoleh hasil dari proses pengembangan sistem menggunakan metode *Scrum*, langkah berikutnya adalah mengurai hasil tersebut secara rinci guna mendapatkan pemahaman yang lebih mendalam. Ini mencakup pembahasan detail mengenai temuan, perbandingan dengan tujuan awal, serta evaluasi kesesuaian dengan kebutuhan pengguna. Selanjutnya, kesimpulan dari hasil pengujian yang telah dilakukan disajikan. Tabel pengujian *blackbox testing* juga diperlihatkan sebagai bagian penting dari analisis, yang merinci berbagai skenario uji yang telah dijalankan dan hasil yang terkait. Semua tahapan ini penting untuk memastikan kualitas dan kinerja sistem yang dikembangkan.

Tabel 4. 13 Hasil Pengujian Blackbox

Fitur *Website*	***Chrome***	***Mozilla***	***Safari***
Login	✓	✓	✓
Register	✓	✓	✓
Logout	✓	✓	✓
Create Area	✓	✓	✓
Read Area	✓	✓	✓
Update Area	✓	✓	✓
Delete Area	✓	✓	✓
Create Data Kerentanan	✓	✓	✓
Read Data Kerentanan	✓	✓	✓
Update Data Kerentanan	✓	✓	✓
Delete Data Kerentanan	✓	✓	✓

Tabel 4.13 menampilkan hasil uji *Blackbox Testing* yang dilakukan oleh peneliti pada berbagai fitur yang ada di *website* yang sebelumnya telah selesai dikembangkan. *Blackbox Testing* merupakan metode pengujian perangkat lunak yang dilakukan tanpa memerhatikan struktur *internal* dari sistem yang diuji. Dalam konteks ini, pengujian *blackbox* dilakukan pada tiga browser yang berbeda yaitu *Chrome, Mozilla, dan Safari*. Hasil pengujian *blackbox* ini membantu dalam mengevaluasi fungsionalitas dan kompatibilitas sistem di berbagai lingkungan *browser* yang umum digunakan oleh pengguna. Hasil dari pengujian *blackbox* menunjukkan bahwa sebagian besar fitur *website* telah berhasil diuji dan berfungsi dengan baik pada ketiga *browser* tersebut. Berikut adalah penjelasan mengenai setiap fitur yang telah diuji:

1) *Login*

Fitur ini berhasil lolos pengujian terhadap ketiga *browser* yang digunakan. Fitur *login* akan digunakan oleh *admin* untuk masuk ke halaman *dashboard*. Proses pengujian yang telah dilakukan telah memastikan bahwa fitur ini berfungsi dengan baik di berbagai lingkungan *browser* yang umum digunakan, sehingga memungkinkan *admin* untuk dengan lancar mengakses halaman *dashboard* menggunakan fitur *login* yang telah diuji.

2) *Register*

Fitur ini berhasil lolos pengujian terhadap tiga *browser* yang digunakan. Fitur *registrasi* akan digunakan oleh calon *admin* untuk membuat akun *admin*, dan setelah itu akan dilanjutkan untuk masuk ke halaman *dashboard*. Pengujian yang telah dilakukan memastikan bahwa fitur *registrasi* berfungsi dengan baik di berbagai lingkungan *browser* yang umum digunakan. Hal ini memungkinkan calon *admin* untuk dengan sukses mendaftarkan akun dan melanjutkan ke halaman *dashboard* setelah proses *registrasi* selesai.

3) *Logout*

Fitur ini telah berhasil melewati uji coba pada tiga jenis *browser* yang digunakan. Fitur *logout* akan digunakan oleh *admin* untuk keluar dari sesi akses, kemudian admin dapat mengakses kembali halaman *dashboard*. Pengujian yang telah dilakukan memastikan bahwa fitur *logout* berfungsi dengan baik dalam berbagai lingkungan *browser* yang umum digunakan. Hal ini memungkinkan *admin* untuk keluar dari akun dengan sukses dan kemudian dapat kembali masuk ke halaman *dashboard* setelah proses *logout* selesai.

4) *Create* Area

Fitur ini berhasil lolos pengujian terhadap tiga jenis *browser* yang digunakan. Fitur "*Create* Area" akan digunakan oleh *admin* untuk menambahkan data area secara satu per satu melalui pengisian formulir. Proses pengujian yang telah dilakukan memastikan bahwa fitur ini berfungsi dengan baik dalam berbagai lingkungan *browser* yang umum digunakan. Hal ini memungkinkan admin untuk dengan sukses menambahkan data area secara individual menggunakan formulir yang telah diuji.

5) *Read* Area

Fitur ini berhasil melewati pengujian pada tiga jenis *browser* yang digunakan. Fitur "*Read* Area" ini akan digunakan oleh *admin* untuk membaca informasi mengenai area-area yang ada. Proses pengujian yang telah dilakukan memastikan bahwa fitur ini beroperasi dengan baik dalam berbagai lingkungan *browser* yang umum digunakan. Dengan fitur ini, *admin* dapat dengan sukses membaca informasi mengenai berbagai area yang tersedia di sistem, sesuai dengan hasil pengujian yang telah dilakukan.

6) *Update* Area

Fitur ini berhasil melewati pengujian pada tiga jenis *browser* yang digunakan. Fitur "*Update* Area" akan digunakan oleh *admin* untuk mengubah data area satu per satu melalui pengisian formulir. Proses pengujian yang telah dilakukan memastikan bahwa fitur ini beroperasi dengan baik dalam berbagai lingkungan *browser* yang umum digunakan. Dengan fitur ini, *admin* dapat dengan sukses mengubah data area secara individual menggunakan formulir yang telah diuji.

7) *Delete* Area

Fitur ini berhasil melewati pengujian pada tiga jenis *browser* yang digunakan. Fitur "*Delete* Area" akan digunakan oleh *admin* untuk menghapus data area satu per satu melalui pengisian formulir. Proses pengujian yang telah dilakukan memastikan bahwa fitur ini beroperasi dengan baik dalam berbagai lingkungan *browser* yang umum digunakan. Dengan fitur ini, *admin* dapat dengan sukses menghapus data area secara individual menggunakan formulir yang telah diuji.

8) *Create* Data Kerentanan

Fitur ini berhasil melewati pengujian pada tiga jenis *browser* yang digunakan. Fitur "*Create* Data Kerentanan" akan digunakan oleh *admin* untuk menambahkan data kerentanan satu per satu melalui pengisian formulir. Proses pengujian yang telah dilakukan memastikan bahwa fitur ini beroperasi dengan baik dalam berbagai lingkungan *browser* yang umum digunakan. Dengan fitur ini, *admin* dapat dengan sukses menambahkan data kerentanan secara individual menggunakan formulir yang telah diuji.

9) *Read* Data Kerentanan

Fitur ini berhasil melewati pengujian pada tiga jenis *browser* yang digunakan. Fitur "*Read* Data Kerentanan" ini akan digunakan oleh *admin* untuk membaca informasi mengenai data kerentanan yang ada. Proses pengujian yang telah dilakukan memastikan bahwa fitur ini beroperasi dengan baik dalam berbagai lingkungan *browser* yang umum digunakan. Dengan fitur ini, *admin* dapat dengan sukses membaca informasi mengenai data kerentanan sesuai dengan hasil pengujian yang telah dilakukan.

10) *Update* Data Kerentanan

Fitur ini berhasil melewati pengujian pada tiga jenis *browser* yang digunakan. Fitur "*Update* Data Kerentanan" akan digunakan oleh *admin* untuk mengubah data kerentanan satu per satu melalui pengisian formulir. Proses pengujian yang telah dilakukan memastikan bahwa fitur ini beroperasi dengan baik dalam berbagai lingkungan browser yang umum digunakan. Dengan fitur ini, admin dapat dengan sukses mengubah data kerentanan secara individual menggunakan formulir yang telah diuji.

11) *Delete* Data Kerentanan

Fitur ini berhasil lolos pengujian terhadap tiga jenis *browser* yang digunakan. Fitur "*Delete* Data Kerentanan" akan digunakan oleh *admin* untuk menghapus data kerentanan satu per satu melalui pengisian formulir. Proses pengujian yang telah dilakukan memastikan bahwa fitur ini beroperasi dengan baik dalam berbagai lingkungan *browser* yang umum digunakan. Dengan fitur ini, admin dapat dengan sukses menghapus data kerentanan secara individual menggunakan formulir yang telah diuji.