PENGEMBANGAN APLIKASI PEMASARAN PRODUK USAHA MAHASISWA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS LAMPUNG BERBASIS WEB

(Skripsi)

Oleh

NAFASYA RAHMA SAFITRA
NPM 2017051023

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2024

PENGEMBANGAN APLIKASI PEMASARAN PRODUK USAHA MAHASISWA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS LAMPUNG BERBASIS WEB

Oleh

NAFASYA RAHMA SAFITRA

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA KOMPUTER

Pada

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2024

ABSTRAK

PENGEMBANGAN APLIKASI PEMASARAN PRODUK USAHA MAHASISWA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS LAMPUNG BERBASIS WEB

Oleh:

NAFASYA RAHMA SAFITRA

Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung secara aktif mendukung kegiatan kewirausahaan mahaiswa. Namun, pada saat ini FMIPA belum mempunyai suatu fasilitas yang dapat mengakomodasi dan memasarkan produk para mahasiswa secara terpusat dan terintegrasi. Proses kelola data dan pemasaran produk masih dilakukan secara manual dan individual melalui media sosial masing-masing yang cenderung kurang efektif. Adapun tujuan penelitian ini adalah untuk mengembangkan aplikasi pemasaran produk usaha mahasiswa FMIPA Universitas Lampung untuk memfasilitasi dan memudahkan proses kelola data dan pemasaran produk. Penelitian ini menggunakan metode *Personal Extreme Programming* yang dijalankan dalam tiga iterasi. Teknologi yang digunakan adalah HAPI untuk *back-end* dan React untuk *front-end*. Hasil penelitian ini mencakup REST API dan aplikasi web yang telah melalui serangkaian pengujian, termasuk pengujian fungsional, keamanan, kinerja, serta *User Acceptance Testing*. Kesimpulannya, penelitian yang dilakukan berhasil mengembangkan aplikasi pemasaran produk usaha mahasiswa FMIPA Universitas Lampung berbasis web yang dapat memfasilitasi dan memudahkan proses kelola data dan pemasaran produk di lingkungan Universitas Lampung dan sekitarnya.

Kata Kunci: pemasaran produk, kewirausahaan mahasiswa, aplikasi berbasis web, personal extreme programming.

ABSTRACT

DEVELOPMENT OF A WEB-BASED MARKETING APPLICATION FOR STUDENT BUSINESS PRODUCTS OF THE FACULTY OF MATHEMATICS AND NATURAL SCIENCES, UNIVERSITY OF LAMPUNG

By:

NAFASYA RAHMA SAFITRA

The Faculty of Mathematics and Natural Sciences of Lampung University actively supports student entrepreneurship activities. However, currently FMIPA does not have a facility that can accommodate and market student products in a centralized and integrated manner. The process of managing data and marketing products is still done manually and individually through each social media which tends to be less effective. The purpose of this research is to develop an application for marketing student business products in FMIPA University of Lampung to facilitate the process of managing data and marketing products. This research uses the Personal Extreme Programming method which is run in three iterations. The technology used is HAPI for the back-end and React for the front-end. The results of this research include a REST API and a web application that has gone through several stages of testing, including functional, security, performance, and User Acceptance Testing. In conclusion, the research conducted succeeded in developing a web-based for FMIPA University of Lampung student business product marketing application that can facilitate the process of managing data and marketing products within the University of Lampung and its surroundings.

Keywords: product marketing, student entrepreneurship, web-based application, personal extreme programming.

Judul Skripsi	:	**PENGEMBANGAN APLIKASI PEMASARAN PRODUK USAHA MAHASISWA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS LAMPUNG BERBASIS WEB**
Nama Mahasiswa	:	Nafasya Rahma Safitra
Nomor Pokok Mahasiswa	:	2017051023
Program Studi	:	S1-Ilmu Komputer
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

Bambang Hermanto, S.Kom., M.Cs.
NIP. 197909122008121 002

M. Iqbal Parabi, S.SI., M.T.
NIP. 19901130 201504 1 002

2. Ketua Jurusan Ilmu Komputer

Dwi Sakethi, S.Si., M.Kom.
NIP. 196806111998021001

MENGESAHKAN

1. Tim Penguji

Ketua Penguji : **Bambang Hermanto, S.Kom., M.Cs.** .....

Sekreratis Penguji : **M. Iqbal Parabi, S.SI., M.T.** .....

Penguji Utama : **Didik Kurniawan, S.Si., M.T.** .....

2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Dr. Eng. Heri Satria, S.Si., M.Si.
NIP. 197110012005011002

Tanggal Lulus Ujian Skripsi: **11 Oktober 2024**

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Nafasya Rahma Safitra

NPM : 2017051023

Menyatakan bahwa skripsi saya yang berjudul "**Pengembangan Aplikasi Pemasaran Produk Usaha Mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung Berbasis Web**" merupakan karya saya sendiri dan bukan karya orang lain. Seluruh tulisan yang tertulis dalam skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti skripsi saya merupakan hasil penjiplakan atau dibuat orang lain, maka saya bersedia menerima sanksi berupa pencabutan gelar yang telah saya terima.

Bandar Lampung, 11 Oktober 2024

Nafasya Rahma Safitra

NPM. 2017051023

RIWAYAT HIDUP

Penulis lahir di Pujodadi pada tanggal 5 Desember 2002 sebagai anak pertama dari pasangan Bapak Sobri dan Ibu Agustina Tri Wulandari. Penulis menyelesaikan pendidikan formal di SD Negeri 4 Pujodadi dan selesai pada tahun 2015. Kemudian melanjutkan pendidikan menengah pertama di SMP Negeri 1 Ambarawa yang diselesaikan pada 2018. Penulis selanjutnya menempuh pendidikan menengah atas di SMA Negeri 1 Gadingrejo dan lulus pada tahun 2020.

Pada tahun yang sama penulis terdaftar sebagai mahasiswa Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung melalui jalur SNMPTN. Selama menjadi mahasiswa, penulis aktif mengikuti berbagai kegiatan antara lain.

1. Menjadi asisten dosen mata kuliah Logika pada semester ganjil tahun ajaran 2021/2022 dan mata kuliah Struktur Data dan Algoritma pada semester genap tahun ajaran 2021/2022 di Program Studi S1 Ilmu Komputer.
2. Mengikuti Kursus Desain Interaksi Untuk UI/UX Designer Pemula Program Kredensial Mikro Mahasiswa Indonesia (KMMI) tahun 2021.
3. Menjadi penerima Beasiswa Bank Indonesia tahun 2022.
4. Mengikuti program Studi Independen Bersertifikat Batch 3 di Dicoding Academy jalur pembelajaran Front-End Web dan Back-End pada tahun 2022.
5. Melaksanakan Kuliah Kerja Nyata (KKN) Mandiri di Desa Fajar Baru, Kecamatan Jati Agung, Kabupaten Lampung Selatan pada tahun 2023.

6. Menjadi Sekretaris Divisi Kemitraan dan Kerjasama Generasi Baru Indonesia (GenBI) Komisariat Universitas Lampung pada tahun 2023.
7. Menjadi penerima pendanaan Program Mahasiswa Wirausaha (PMW) Universitas Lampung tahun 2023.
8. Mengikuti program Studi Independen Bersertifikat Batch 4 di Bangkit Academy jalur pembelajaran Cloud Computing pada tahun 2023.

MOTTO

"*Boleh jadi kamu membenci sesuatu, padahal ia amat baik bagimu, dan boleh jadi (pula) kamu menyukai sesuatu, padahal ia amat buruk bagimu; Allah mengetahui, sedang kamu tidak mengetahui."*
(Q.S. Al-Baqarah [2]: 216).

"*Maka sesungguhnya bersama kesulitan ada kemudahan, sesungguhnya bersama kesulitan ada kemudahan.*"
(Q.S. Al-Insyirah [94]: 5-6)

"*Make progress one step at a time. Remain determined to finish what you've just started.*"
(Bangkit Academy)

PERSEMBAHAN

Alhamdulillahirobbilalamin

Puji dan syukur tercurahkan kepada Allah Subhanahu Wa Ta'ala atas segala Rahmat dan Karunia-Nya sehingga saya dapat menyelesaikan skripsi ini. Shalawat serta salam selalu tercurahkan kepada Nabi Muhammad Shallallahu Alaihi Wasallam.

Kupersembahkan karya ini kepada:

Kedua Orang Tuaku Tercinta

Atas segala pengorbanan, perjuangan, kasih sayang, perhatian, dukungan dan do'a yang selalu menyertaiku. Kuucapkan terima kasih sebesar-besarnya karena telah mendidik dan membesarkanku dengan penuh kasih sayang yang tak akan terbalaskan.

Seluruh Keluarga Besar Ilmu Komputer 2020

Yang senantiasa memberikan semangat dan dukungan.

Almamater Tercinta, Universitas Lampung dan Jurusan Ilmu Komputer

Tempat bernaung mengemban semua ilmu untuk menjadi bekal kehidupan

SANWACANA

Puji syukur kehadirat Allah Subhanahu Wa Ta'ala, karena telah memberikan limpahan nikmat, rahmat dan karunia-Nya. Shalawat serta salam semoga senantiasa tercurahkan kepada junjungan Nabi Muhammad Shallallahu Alaihi Wasallam, sehingga penulis dapat menyelesaikan skripsi yang berjudul "**Pengembangan Aplikasi Pemasaran Produk Usaha Mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung Berbasis Web**" dengan baik dan lancar.

Selesainya skripsi ini tidak terlepas dari bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, ucapan terima kasih ditujukan kepada:

1. Kedua orang tua yang selalu memberikan doa terbaik, dukungan, motivasi, dan kasih sayang yang tak terhingga nilainya.
2. Bapak Bambang Hermanto S.Kom., M.Cs. selaku dosen pembimbing utama yang selalu membimbing, memberikan arahan, masukan dan saran dalam penyelesaian skripsi maupun hal-hal di luar skripsi.
3. Bapak M. Iqbal Parabi S.SI., M.T. selaku dosen pembimbing kedua yang selalu membimbing, memberikan arahan, masukan dan saran dalam penyelesaian skripsi.
4. Bapak Didik Kurniawan S.Si., M.T. sebagai dosen pembahas yang telah memberikan masukan serta saran yang bermanfaat dalam perbaikan skripsi.
5. Bapak Dr. Eng. Heri Satria, S.Si., M.Si. selaku dekan FMIPA Universitas Lampung.
6. Bapak Dwi Sakethi, S.Si., M.Kom. selaku Ketua Jurusan Ilmu Komputer FMIPA Universitas Lampung.

7. Ibu Anie Rose Irawati, S.T., M.Cs. selaku Sekretaris Jurusan Ilmu Komputer FMIPA Universitas Lampung.
8. Bapak Favorisen Rosyking Lumbanraja, S.Kom., M.Si., Ph.D. selaku Dosen Pembimbing Akademik.
9. Seluruh Dosen, Staf, dan Karyawan Jurusan Ilmu Komputer yang telah memberikan ilmu, pelajaran, dan bantuan terbaik selama penulis menempuh pendidikan di Jurusan Ilmu Komputer Universitas Lampung.
10. Adik-adiku Fakhri dan Fandi, serta Bibi Desi yang selalu menjadi penyemangat dan teman yang menghibur saat berada di rumah.
11. Teman seperjuangan semasa kuliah Nur Setiowati, Safiira Rahmah Linisa, dan Hidayatul Khotimah yang selalu ada untuk mendukung, mendoakan, membantu, dan berbagi cerita baik suka maupun duka selama masa perkuliahan.
12. Umi Sri Karnila, Yulia Dwi Putri, Putri Santika Mayangsari, Silvia Rukmana, Irma Azizah, serta teman-teman lainnya yang tidak dapat penulis sebutkan satu persatu yang telah membantu dan menemani penulis pada masa pengerjaan skipsi ini.
13. Teman-teman Jurusan Ilmu Komputer Angkatan 2020 yang menjadi keluarga satu angkatan selama menjalankan masa studi di Jurusan Ilmu Komputer Universitas Lampung.
14. Diri sendiri, terima kasih telah bertanggung jawab dan tidak menyerah untuk menyelesaikan apa yang telah dimulai dengan segala tantangan yang harus dihadapi.

Penulis menyadari bahwa penyusunan skripsi ini masih jauh dari kata sempurna. Namun penulis sangat mengharapkan skripsi ini dapat bermanfaat bagi para civitas akademika Universitas Lampung pada umumnya dan mahasiswa Ilmu Komputer pada khususnya.

Bandar Lampung, 11 Oktober 2024

Nafasya Rahma Safitra
NPM. 2017051023

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR KODE

Kode Halaman

I. PENDAHULUAN

1.1. Latar Belakang

Universitas Lampung merupakan salah satu perguruan tinggi yang senantiasa mendorong serta mendukung minat kewirausahaan para mahasiswa. Berbagai program dan kegiatan diselenggarakan guna menumbuhkan minat dan motivasi mahasiswa dalam wirausaha untuk mempersiapkan lulusan yang kompeten dan berdaya saing tinggi. Tidak hanya di tingkat universitas, kegiatan untuk mendukung kegiatan kewirausahaan di tingkat fakultas juga aktif dilakukan. Misalnya di Fakultas Keguruan dan Ilmu Pendidikan (FKIP) yang mempunyai Edu Fun Kafe sebagai wadah untuk melatih mahasiswa dalam menjalankan bisnis. Selain itu, terdapat FEB Mart di Fakultas Ekonomi dan Bisnis sebagai tempat bagi mahasiswa untuk belajar mengelola usaha dan keuangan. Program tersebut mendapat dukungan penuh dan diharapkan dalam menjadi fasilitas penyaluran minat dan pengembangan keterampilan mahasiswa di bidang bisnis.

Dukungan terhadap kegiatan wirausaha juga secara aktif dilakukan oleh Fakultas Matematika dan Ilmu Pengetahuan Alam misalnya dengan menyelenggaraan lomba dan *event* bazar untuk para mahasiswanya. Namun, pada saat ini FMIPA belum mempunyai suatu fasilitas yang dapat mengakomodasi dan memasarkan produk para mahasiswa secara *online* agar dapat diakses kapan saja dan dimana saja. Selama ini, proses pemasaran *online* produk mahasiswa masih dilakukan secara individual melalui media sosial masing-masing terutama WhatsApp, Instagram, dan TikTok. Jangkauan konsumen juga belum dapat menyeluruh dan terbatas di lingkup teman atau relasi yang dimiliki saja. Pemanfaatan media sosial sebagai platform pemasaran dan promosi belum sepenuhnya optimal karena kurangnya pengetahuan dan waktu dalam mengelola masing-masing platform agar efektif

untuk mendatangkan konsumen. Selain itu, pemesanan masih dilakukan secara konvensional melalui pengiriman pesan (*chat*) kepada penjual. Setelah itu, penjual harus mencatat ulang data atau riwayat pesanan di aplikasi lain seperti aplikasi catatan atau Excel. Hal ini mengakibatkan proses mengelolaan data menjadi kurang efektif. Banyak usaha mahasiswa, baik hasil dari program pembinaan, praktik mata kuliah kewirausahaan, maupun usaha mandiri, terkendala atau bahkan terhenti karena minimnya pembeli. Hal ini juga dapat disebabkan oleh ketidaktahuan target konsumen terhadap informasi produk yang dipasarkan.

Sebagai fakultas yang memiliki Jurusan Ilmu Komputer di dalamnya, sudah sepatutnya FMIPA memiliki solusi berbasis teknologi untuk pemasaran produk usaha mahasiswa yang lebih efisien dan terpusat sehingga akan ada lebih banyak calon konsumen yang mengetahui informasi produk yang ditawarkan. Maka dari itu, adanya sebuah platform digital untuk pemasaran diperlukan sebagai sarana pendukung untuk menunjang proses pengelolaan dan pemasaran produk yang mudah diakses melalui internet. Penggunaan platform digital memberikan dampak positif yang signifikan terhadap pengembangan UMKM. Platform digital memiliki askes pasar yang lebih luas dan mememungkinkan otomatisasi proses bisnis sehingga menghasilkan efisiensi operasional (Huda dan Tukino, 2023).

Penggunaan platform digital untuk membantu proses bisnis terus meningkat secara pesat di masa sekarang. Hal ini dapat ditandai dengan perkembangan bisnis *online* yang dilakukan di kalangan masyarakat tidak terkecuali mahasiswa. Fenomena ini tidak terlepas dari adanya perubahan perilaku konsumen yang cenderung menginginkan hal yang serba mudah dan cepat (Zulfa dan Hidayati, 2018). Oleh karena itu, para pelaku usaha juga harus dapat beradaptasi dengan kondisi yang ada dengan memanfaatkan berbagai media digital sebagai penunjang dalam proses bisnis yang dijalankan. Pemerintah juga telah memberikan dasar hukum terkait hal ini dalam rangka mendukung perkembangan *e-commerce* (niaga elektronik atau niaga-el) di Indonesia. Misalnya dengan dikeluarkannya Peraturan Pemerintah (PP) Nomor 80 Tahun 2019 tentang Perdagangan melalui Sistem Elektronik. Selain itu, terdapat juga Peraturan Menteri Perdagangan Nomor 50 Tahun 2020 Ketentuan Perizinan Usaha, Periklanan, Pembinaan, dan Pengawasan Pelaku Usaha dalam

Perdagangan Melalui Sistem Elektronik. Adanya peraturan-peraturan tersebut diharapkan dapat menjadi kepastian hukum bagi pelaku niaga elektronik di Indonesia serta dapat mendorong pertumbuhan perdagangan digital yang tengah berkembang pesat saat ini.

Berdasarkan latar belakang dan permasalahan yang telah dipaparkan, maka penulis bermaksud untuk menyediakan sebuah platform digital untuk melakukan niaga elektronik di kalangan mahasiswa pada lingkup fakultas dengan membangun Aplikasi Pemasaran Produk Usaha Mahasiswa FMIPA Universitas Lampung Berbasis Web. Aplikasi ini berperan sebagai sarana penyaluran informasi dengan tujuan untuk mempertemukan penjual dan pembeli secara virtual. Aplikasi ini dikembangkan untuk memungkinkan pengelolaan data, pemasaran, pembelian, dan pencatatan dengan mudah dalam satu sistem.

Adanya platform ini diharapkan dapat membantu para mahasiswa untuk memperluas jangkauan pemasaran terutama di lingkungan Universitas Lampung dan sekitarnya serta mempermudah pengelolaan data terkait dengan kegiatan kewirausahaan mereka. Di samping itu, konsumen lebih cepat dalam mendapatkan informasi dan melakukan pemesanan terhadap produk yang diinginkan. Kolaborasi inovatif ini tidak hanya menggabungkan kontribusi mahasiswa di bidang teknologi dan kewirausahaan, tetapi juga memberikan dukungan terhadap kreativitas anak bangsa dalam upaya memajukan *digital entrepreneurship* di kalangan kaum muda.

Aplikasi ini akan dikembangkan berbasis web sehingga mudah diakses dari perangkat apapun melalui internet. Bahasa pemrograman yang digunakan adalah JavaScript dengan *library* React untuk *front-end* dan *framework* HAPI untuk *back-end*. Selain itu, terdapat penerapan mekanisme autentikasi JSON *Web Token* (JWT) untuk memastikan sumber hanya dapat diakses oleh pengguna yang sah dan berwenang sehingga meningkatkan keamanan aplikasi. Penggunaan arsitektur REST API untuk menjembatani aplikasi *front-end* dan *back-end* dapat memberikan fleksibilitas pengembangan lanjutan karena tidak bergantung pada teknologi dan platform yang digunakan. Selain itu, perubahan pada aplikasi juga dapat dilakukan tanpa mempengaruhi keseluruhan fungsionalitas sistem.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka permasalahan yang dapat dirumuskan pada penelitian ini adalah bagaimana mengembangkan sebuah aplikasi berbasis web yang dapat memfasilitasi dan memudahkan proses pengelolaan data dan pemasaran produk usaha mahasiswa FMIPA Universitas Lampung.

1.3. Batasan Masalah

Agar permasalahan yang diteliti tidak meluas, maka peneliti membuat batasan-batasan berikut ini:

1. Cakupan dari penelitian ini meliputi pengembangan aplikasi *back-end*, REST API, dan *front-end* berbasis web.
2. Pengembangan menggunakan bahasa pemrograman JavaScript dengan *library* React untuk *front-end* dan *framework* HAPI untuk *back-end.*
3. Pengembangan sistem tidak termasuk integrasi dengan *payment gateway* dan penyedia jasa pengiriman.
4. Mekanisme pengembalian dana atau barang belum dapat dilakukan melalui sistem.
5. Proses komunikasi antara penjual dan pembeli dilakukan di luar sistem.

1.4. Tujuan

Tujuan yang ingin dicapai penulis dalam penelitian ini adalah mengembangkan aplikasi pemasaran produk usaha mahasiswa FMIPA Universitas Lampung untuk memfasilitasi dan memudahkan proses kelola data dan pemasaran produk sehingga data usaha akan tersimpan dengan baik serta informasi produk dapat tersebar di lingkungan Universitas Lampung dan sekitarnya.

1.5. Manfaat

Penelitian ini memberikan manfaat sebagai berikut:

1. Tersedianya platform yang dapat membantu proses pemasaran produk usaha mahasiswa FMIPA Universitas Lampung.
2. Sebagai sarana belajar dan latihan mahasiswa dalam mengelola bisnis *online* secara mandiri.
3. Sebagai bentuk dukungan untuk kegiatan kewirausahaan di FMIPA Universitas Lampung.
4. Memperdalam pemahaman dan kemampuan penulis dalam mengimplementasikan teknologi *fullstack* JavaScript untuk pengembangan aplikasi.
5. Dapat dijadikan sebagai rujukan literatur dalam penelitian lain yang memiliki topik serupa.

II. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Penelitian terdahulu digunakan sebagai landasan dan sumber referensi bagi penelitian yang akan dilakukan. Dengan mengidentifikasi muatan penelitian-penelitian sebelumnya, penulis dapat menganalisis dan melakukan perbandingan untuk menentukan kesamaan dan perbedaan antara penelitian terdahulu dengan penelitian ini. Berikut adalah beberapa penelitian terdahulu yang dijadikan rujukan dalam penelitian ini.

1. **Pengembangan Sistem Informasi Sentra Inovasi dan Inkubator Bisnis (Sikubis) Berbasis Android**

   Penelitian oleh Heningtyas dkk. (2022) dilakukan untuk mengembangkan kewirausahaan di lingkungan Universitas Lampung dengan menyediakan media untuk jual beli produk riset mahasiswa, dosen, dan tenaga kependidikan yang ada di Inkubator Bisnis Universitas Lampung. Solusi yang diambil adalah dengan mengembangkan aplikasi *marketplace* dengan nama Sikubis berbasis Android untuk pelanggan, dan versi web untuk admin. Metodologi yang digunakan dalam pengembangan sistem adalah *prototyping*. Kesimpulan yang dapat diambil dari penelitian ini yaitu Sikubis dapat membantu tenant di Sentra Inovasi dan Inkubator Bisnis Universitas Lampung yang terdiri dari civitas akademika Universitas Lampung, dalam mengelola distribusi penjualan produk-produk yang dibina. Selain itu, sistem ini juga membantu pihak Universitas Lampung dalam memonitor kegiatan kewirausahaan yang berlangsung di Inkubator Bisnis Universitas Lampung.

2. **Rancang Bangun Sistem Informasi Kewirausahaan Mahasiswa**

   Penelitian ini dilakukan oleh Mukaromah (2019) dengan tujuan sebagai wadah kewirausahaan yang dapat mengakomodasi hasil produk atau layanan yang dihasilkan mahasiswa UPN Veteran Jatim sehingga tetap dapat berlanjut setelah mahasiswa lulus. Proses analisis dan perancangan sistem dilakukan dengan *Iconix Process* menggunakan pendekatan UML yang meliputi *Use case Diagram, Conceptual Data Model* (CDM), dan *Physical Data Model* (PDM). Penelitian ini menggunakan metode *Waterfall* SDLC dan menghasilkan Sistem Informasi Kewirausahaan Mahasiswa sebagai wadah atau forum untuk menjaga jiwa kewirausahaan mahasiswa dan juga alumni UPN Veteran Jatim.

3. **Pengembangan *Student Marketplace* Bagi Mahasiswa Wirausaha UNP**

   Herayono dan Adri (2021) melakukan penelitian dengan mengembangkan aplikasi UNP *Student Marketplace* yang bertujuan untuk memudahkan mahasiswa wirausaha di Universitas Negeri Padang dalam melakukan pemasaran dan promosi tanpa mengeluarkan modal awal yang terlalu banyak. Dengan adanya sistem ini, mahasiswa yang berwirausaha dapat saling berinteraksi dengan mahasiswa yang sedang memerlukan produk yang sesuai. *Marketplace* ini dibuat menggunakan bahasa pemrograman PHP dengan *framework* Codeigniter untuk aplikasi web, dan bahasa pemrograman Flutter dengan IDE Android Studio untuk aplikasi berbasis *mobile*. Penelitian ini menghasilkan aplikasi UNP *Student Marketplace* yang dapat digunakan untuk pemasaran produk, pemesanan produk serta transaksi antar mahasiswa penjual dan mahasiswa pembeli.

4. **Sistem Informasi *Marketplace* Produk Usaha Mikro Kecil Menengah (UMKM)**

   Penelitian ini dilakukan pada Universitas Bina Darma oleh Ardilla dan Hadinata (2022) karena melihat permasalahan pada proses pemasaran produk mahasiswa yang masih dilakukan secara konvensional. Tujuan dibuatnya sistem informasi *marketplace* adalah untuk membantu mahasiswa

dalam pemasaran produk UMKM secara *online* dan cepat. Selain itu, dengan adanya sistem ini diharapkan dapat meningkatkan jangkauan pasar agar lebih luas. Metode penelitian yang digunakan adalah *Rapid Application Development* (RAD) dengan memanfaatkan platform CMS Wordpress. Penelitian ini menghasilkan sebuah Sistem Informasi *Marketplace* Produk Usaha Mikro Kecil Menengah (UMKM) Mahasiswa Universitas Bina Darma berbasis *website*.

2.2. Pemasaran Produk

Pemasaran adalah kegiatan atau aktivitas penyaluran produk (barang dan jasa) kepada para konsumen untuk memenuhi kebutuhan dan keinginan dengan harapan kegiatan tersebut dapat memberi keuntungan bagi kedua belah pihak (Ariyanto dkk., 2023). Produk itu sendiri merupakan sesuatu yang dapat ditawarkan ke pasar untuk diperhatikan dipakai, dimiliki, atau dikonsumsi sehingga dapat memenuhi keinginan atau kebutuhan konsumen (Firmansyah, 2023). Pemasaran produk merupakan bagian dari kegiatan wirausaha atau bisnis yang mempunyai peranan sangat penting. Kegiatan ini bertujuan untuk memperkenalkan atau mempromosikan barang atau jasa yang dimiliki dengan harapan konsumen menjadi tertarik untuk membelinya. Tanpa adanya proses pemasaran produk, kegiatan wirausaha tidak akan berkembang dan dapat berakibat pada kurangnya angka penjualan. Hal ini tentu saja perlu diiringi dengan strategi pemasaran dan kualitas produk yang baik.

Strategi pemasaran suatu produk dapat dilakukan secara *offline* maupun *online*. Pada pemasaran *offline* terjadi transaksi langsung antara produsen dan konsumen. Jangkauan pemasaran *offline* kurang luas dan terbatas pada suatu tempat atau lokasi tertentu saja. Pelaku usaha harus menyiapkan modal lebih untuk membangun atau menyewa tempat usaha di berbagai lokasi untuk memperluas pemasaran. Namun begitu, strategi ini menawarkan keamanan dan kesesuaian produk sehingga konsumen tidak perlu khawatir dengan penipuan atau kecewa dengan produk yang tidak sesuai harapan (Susilawati dkk., 2022).

Adapun strategi pemasaran banyak dilakukan oleh masyarakat pada saat ini yaitu pemasaran *online*. Pemasaran produk secara *online* dapat memanfaatkan berbagai teknologi untuk memperluas jangkauan pemasaran seperti *e-commerce*, *marketplace*, sosial media, dan lainnya. Pemasaran produk secara *online* menjadi pilihan karena alasan kemudahan, kecepatan dan kepraktisan (Susilawati dkk., 2022). Dengan cara pemasaran ini, transaksi tidak terbatas pada tempat tertentu sehingga dapat membantu para pengusaha yang tidak mempunyai cukup modal untuk membuka cabang di berbagai lokasi. Proses promosi untuk memperkenalkan produk ke pasar juga menjadi lebih efisien.

2.3. Aplikasi Berbasis Web

Aplikasi berbasis web adalah perangkat lunak yang dapat diakses menggunakan *web browser* yang terhubung melalui jaringan internet (Saputra, 2020). Aplikasi web tidak membutuhkan proses instalasi sehingga dapat diakses melalui berbagai perangkat. Dengan begitu data dan informasi dapat diakses secara mudah kapanpun dan di manapun.

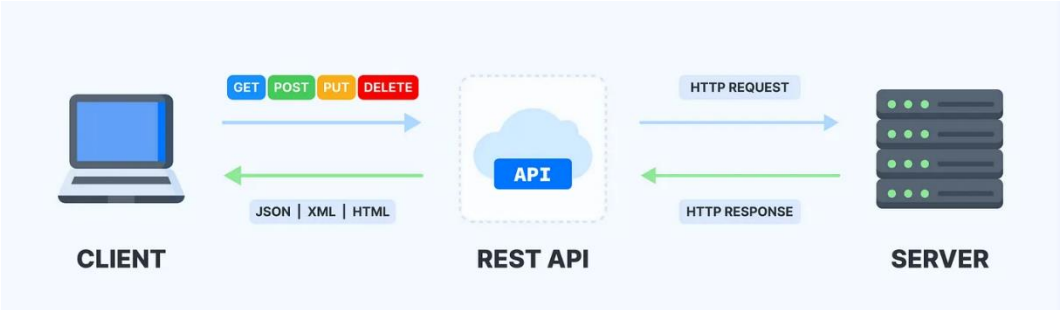
Secara umum aplikasi web memiliki komponen *front-end* (*client-side*) dan *back-end* (*server-side*). Aplikasi *front-end* berisi elemen visual yang dapat berinteraksi langsung dengan pengguna. Adapun bahasa pemrograman yang digunakan adalah HTML, CSS, dan JavaScript. Sementara itu, aplikasi *back-end* memiliki peran penting dalam mengatur logika bisnis, mengelola basis data, dan menyediakan layanan ke *client*. Terdapat banyak bahasa pemrograman yang dapat digunakan untuk pengembangan *back-end*, seperti Java, Python, Node.js, Ruby, Go dan lain-lain.

Selain itu, terdapat juga komponen lainnya yang disebut dengan *Application Programming Interface* (API). Dalam konteks aplikasi web, istilah API biasanya digunakan untuk merujuk *web service* yang menyediakan mekanisme komunikasi antara *client-side* dan *server-side* sebagai sarana pertukaran data. Beberapa

arsitektur API yang umum digunakan adalah REST, SOAP, GraphSQL, atau yang lainnya.

2.3.1. REST API

Representational State Transfer (REST) merupakan seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandarisasi seperti *Hypertext Transfer Protocol* (HTTP). Sementara itu, *Application Programming Interface* (API) adalah antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program (I. Kurniawan dkk., 2020). API yang mengikuti gaya REST disebut dengan RESTful API (Gunawan dan Rahmatullah, 2019). Nama REST diciptakan oleh Roy Fielding dari *University of California*. *Web service* ini sangat sederhana dan ringan jika dibandingkan dengan *Simple Object Access Protocol* (SOAP) (Choirudin dan Adil, 2019). Gambar 1 menunjukkan ilustrasi model REST API.



Gambar 1. Model REST API (www.appmaster.io).

Arsitektur REST memiliki beberapa batasan atau prinsip yang harus diikuti untuk memastikan desain yang konsisten dan efektif (Jain, 2022).

1. *Client-Server Architecture*

   Arsitektur *client-server* mendukung pemisahan tanggung jawab antara *client* dan *server* sehingga bersifat independen satu sama lain. Pengembangan aplikasi *client* dapat dilakukan tanpa mengetahui detail internal dan fungsi-fungsi yang bekerja pada *server*.

2. *Statelessness*

   Server tidak harus mengetahui dan memelihara *state/session* dari *request*. Tugas *server* adalah memberikan *response* tanpa melacak sumbernya dengan suatu *session*. Hal ini dapat membantu meningkatkan kinerja *server* secara keseluruhan.

3. *Cacheability*

   Dengan mekanisme *cache*, *server* tidak perlu menanggapi permintaan yang sama secara berulang. Ini membantu *server* menjadi lebih efisien dan dapat membantu meningkatkan kinerja.

4. *Use of a Layered System*

   Menggunakan sistem berlapis dapat membantu dalam mengatasi beberapa masalah lain seperti otentikasi dan keamanan. Memiliki sistem berlapis bermanfaat untuk debugging akar permasalahan dengan cepat.

5. *Uniform Interface*

   Uniform interface atau keseragaman antarmuka merupakan hal mendasar bagi arsitektur REST. Ini dapat memastikan bahwa sumber daya diidentifikasi berdasarkan *Uniform Resource Identifier* (URI).

6. *Support for Code on Demand*

   Ini memungkinkan *client* dapat mendownload dan mengeksekusi kode yang disediakan *server* (seperti applet Java, skrip JavaScript, dan lainnya).

REST API terdiri dari komponen-komponen berikut ini (Perdana, 2018):

1. *URL Design*

   URL merupakan komponen penting dari sebuah REST API karena menjadi jalur yang digunakan untuk mendapatkan data yang diinginkan. URL biasanya disebut juga dengan *path* atau *endpoint* harus memiliki struktur dan penamaan yang mudah dimengerti. Dokumentasi yang baik dapat mempermudah orang lain dalam memahami fungsionalitas dari setiap URL yang tersedia.

2. *HTTP Verbs*

   Ketika melakukan *request* melalui protokol HTTP, penting untuk mencantumkan *HTTP verbs* guna menunjukkan tindakan yang seharusnya

dilakukan oleh *server*. Metode yang paling sering digunakan adalah GET (mendapatkan data), POST (mengirimkan data), PUT (memperbarui data), dan DELETE (menghapus data).

3. *HTTP Response Code*

   HTTP response code adalah serangkaian kode standar yang digunakan untuk memberikan informasi tentang status permintaan kepada *client*. Secara umum, terdapat tiga kelompok kode yang umumnya digunakan dalam REST API:

   a. Kode 2XX menunjukkan bahwa permintaan yang dilakukan berhasil.
   b. Kode 4XX mengindikasikan bahwa terdapat kesalahan pada sisi klien selama proses permintaan.
   c. Kode 5XX menandakan bahwa terdapat kesalahan pada sisi *server* selama pemrosesan permintaan.

4. Format *Response*

   Response terhadap *request* yang diberikan kepada *client* biasanya berupa data dalam format JSON atau XML. Dengan format ini data kemudian dapat diproses kembali sesuai kebutuhan dengan cara *parsing*.

2.3.2. *Hypertext Transfer Protocol* (HTTP)

Hypertext Transfer Protocol (HTTP) merupakan protokol yang digunakan untuk melakukan permintaan dan memberikan respons antara *client* dan *server*. *Client* HTTP contohnya *web browser* biasanya akan memulai permintaan dengan membuat hubungan TCP/IP ke suatu *port*. *Port* yang umum digunakan pada protokol HTTP adalah *port* 80 (Rombe dkk., 2019).

2.3.3. JavaScript

JavaScript merupakan bahasa pemrograman tingkat tinggi *(High Level Language*) dan dinamis yang pada awalnya dirancang sebagai bahasa pemrograman skrip di sisi *client* (*Client Side Script Programming*). Namun saat ini JavaScript telah mengalami perkembangan yang pesat sehingga dapat digunakan pada *server*, *console*, program desktop, *mobile*, IoT, *game,* dan lain-lain. JavaScript dirancang

oleh Brendan Eich dan dikembangkan oleh Netscape Communication Corporation, Mozilla Foundation JavaScript (Supardi, 2021).

2.3.4. Node.js

Node.js adalah perangkat lunak yang dirancang untuk mengembangkan aplikasi berbasis web dalam sintaks bahasa pemrograman JavaScript. Node.js dapat menjalankan *server* web tanpa menggunakan program *web server* seperti Apache dan Nginx karena memiliki *library server* HTTP bawaan (I. Kurniawan dkk., 2020). Node.js dibuat pada tahun 2009 oleh Ryan Dahl menggunakan mesin Google V82 untuk mengeksekusi kode JavaScript di sisi *server* (Doglio, 2018).

2.3.5. React

Dalam dokumentasi resminya, React didefinisikan sebagai *library* untuk antarmuka pengguna web dan *native*. React merupakan salah satu *library front-end* JavaScript yang bersifat *open-source*. Dibuat oleh Jordan Walke untuk membangun Facebook dan Instagram pada Maret 2013. *Library* ini memungkinkan *software engineers* untuk membuat aplikasi web besar yang dapat memanipulasi data dan memperbarui secara dinamis tanpa perlu memuat ulang halaman. React populer untuk membuat *user interface* di kalangan pengembang saat ini karena sederhana dan efektif.

2.3.6. *Framework* HAPI

HAPI atau Http-API adalah kerangka kerja sumber terbuka untuk membangun aplikasi web dengan Node, yang dibuat oleh tim web seluler di Walmart Labs. Kasus penggunaan HAPI yang paling umum adalah untuk membangun layanan web seperti JSON API, tetapi juga dapat digunakan untuk membangun proksi HTTP dan sebagai komponen *server* dari *website* atau aplikasi *single page* (Harrison, 2016). Tujuan utama HAPI adalah memungkinkan pengembang untuk fokus pada pengkodean logika aplikasi, meninggalkan kode infrastruktur ke kerangka kerja (Doglio, 2018).

Modularitas menjadi aspek keunggulan HAPI. *Framework* ini menggunakan konsep *plugin* yang memungkinkan aplikasi dapat dipecah menjadi bagian-bagian yang lebih kecil. Pendekatan ini memudahkan kolaborasi dalam tim karena setiap

komponen dapat dikerjakan secara independen. Terdapat banyak *plugin* bawaan telah disediakan oleh HAPI sebagai paket npm. Selain itu, pengguna juga dapat membuat *plugin* sendiri sesuai dengan kebutuhan (Harrison, 2016). Beberapa kelebihan *framework* ini menurut Sud (2020) dalam buku Practical Hapi antara lain:

1. HAPI menawarkan ekosistem yang kaya

   Pada *framework* ini tersedia banyak *plugin* yang hampir selalu dapat memenuhi kebutuhan berbagai jenis aplikasi. Adanya *plugin* membantu pengembangan fungsi aplikasi tanpa perlu menulis kode dari awal.
2. Konfigurasi tidak bergantung pada urutan tertentu

   Cara mengatur dan mengkonfigurasi komponen dalam HAPI tidak bergantung pada urutan tertentu sehingga dapat meminimalisir adanya kesalahan karena urutan komponen yang tidak sesuai.
3. *Plugin* dapat saling bergantung dengan aman

   Plugin dalam HAPI dalam saling bergantung satu sama lain tanpa masalah, bahkan jika terdapat *plugin* yang memerlukan *plugin* lain untuk dijalankan terlebih daluhu.
4. *Cache, Plugin, Decorator* dan metode *server* terlindungi

   Elemen-elemen ini memiliki proteksi bawaan yang mencegah perubahan sembarangan atau tidak sengaja. Hal ini dapat menghindarkan pada kesalahan yang biasanya diakibatkan oleh *middleware* yang kompleks.
5. Fleksibel untuk berbagai jenis aplikasi

   Penggunaan *plugin* memberikan fleksibilitas dan kemudahan dalam penambahan atau pengubahan fungsi-fungsi tertentu tanpa memodifikasi inti dari aplikasi.

2.3.7. *JavaScript Object Notation* (JSON)

JSON adalah format untuk berbagi data yang sangat mudah ditransfer antara *server* dan *client*. Sesuai namanya yaitu *JavaScript Object Notation*, format ini merupakan turunan dari bahasa pemrograman JavaScript. Meskipun demikian, JSON juga dapat digunakan pada bahasa pemrograman lain seperti Python, Ruby, PHP, dan Java. Format JSON dapat disimpan dalam file berekstensi .json atau dimasukkan dalam suatu variabel (I. Kurniawan dkk., 2020).

2.3.8. Autentikasi HTTP

Autentikasi merupakan proses validasi ketika pengguna masuk ke dalam sistem. Proses ini dilakukan untuk melindungi data-data yang ada dalam sistem dan juga untuk memastikan bahwa seseorang yang mengakses sistem adalah autentik atau asli. Autentikasi berfungsi sebagai program pengamanan untuk mencegah pihak-pihak yang tidak memiliki otoritas dalam melakukan akses ke sistem. Pengguna harus menyediakan informasi yang diperlukan oleh penyedia layanan untuk membuktikan haknya dalam mendapatkan sumber daya. Sebaliknya, penyedia layanan harus memastikan bahwa akses sumber daya ini tidak dapat dilakukan oleh pihak yang tidak berhak (Khairina, 2011).

Jenis autentikasi HTTP yang paling umum digunakan untuk mengakses API adalah sebagai berikut (Jain, 2022).

1. *Basic Authentication*

   Basic authentication merupakan mekanisme autentikasi yang sangat sederhana untuk mengakses API. *Username* dan *password* yang dikirimkan pengguna hanya akan dikodekan dalam sebuah varian dari Base642 (RFC7617) dan tidak dienkripsikan. Penggunaan tipe autentikasi ini dapat dilakukan dengan melampirkan *username* dan *password* pada *header request* atau dapat juga dijadikan sebagai parameter pada saat mengirim *request* ke API. Pendekatan ini kurang aman dan masih mudah untuk diretas.

2. *Session-Based Authentication*

   Untuk mengakses sumber daya di *server* melalui HTTP, *client* memesan sesi dengan mengirimkan identifikasi ke *server*. Kemudian komunikasi selanjutnya akan dibuat menggunakan identifikasi yang sama. *Server* mengirimkan SESSIONID dalam *response header* ke *client*, dan komunikasi selanjutnya dilakukan berdasarkan SESSIONID di *server*. Umur sesi dapat diatur dengan menetapkan *expiration time* pada *header*. Cara lainnya adalah dengan menulis ulang URL, namun metode ini tidak aman karena jika seseorang berhasil memperoleh SESSIONID maka dapat membahayakan keamanan sistem.

3. *Token/JWT-based Authentication*

   Pada saat masuk ke dalam aplikasi menggunakan kredensial, pengguna akan mendapatkan token untuk mengakses aplikasi. Token ini berlaku untuk jangka waktu tertentu, sehingga ketika masa berlaku token sudah habis maka dibutuhkan token baru yang dibuat oleh *server*. Setiap kali *client* meminta sumber daya, token perlu dilampirkan dalam *header*. *Server* tidak perlu menyimpan *session* atau *state* sehingga tidak membebani kinerja *server*. JSON *Web Token* (JWT) merupakan standar terbuka dalam RFC7519 sebagai pengiriman informasi antara *client* dan *server* melalui objek JSON. JSON menjadi format autentikasi berbasis token yang lebih disukai karena bersifat ringan dan tidak terlalu bertele-tele.

2.3.9. JSON *Web Token* (JWT)

Berdasarkan uraian dari situs resminya, JSON *Web Token* JWT adalah sebuah standar terbuka (RFC 7519) yang mendefinisikan sebuah cara yang ringkas dan mandiri untuk mentransmisikan informasi dengan aman sebagai sebuah objek JSON kepada pihak-pihak yang terlibat. Penggunaan paling umum dari JWT yaitu untuk autentikasi dan otorisasi pada aplikasi web dan *mobile*.

JSON Web Token terdiri dari tiga bagian yang dipisahkan oleh tanda titik. Berikut adalah komponen dari JWT:

1. *Header*

   Header biasanya tersusun oleh dua bagian antara lain tipe token, yaitu JWT, dan algoritma penandatanganan yang digunakan, misalnya HMAC SHA256 atau RSA.

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Gambar 2. JWT *Header* (www.jwt.io).

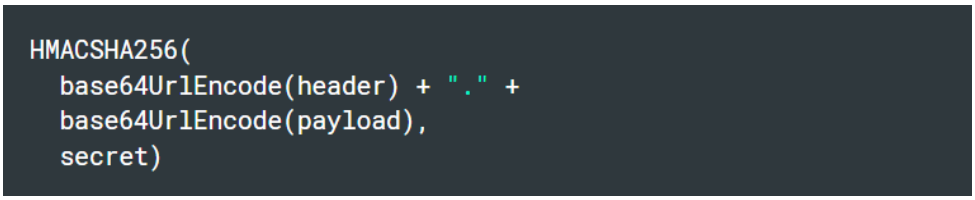
2. *Payload*

   Bagian kedua dari token adalah *payload*, yang mengandung klaim. Klaim merujuk pada informasi yang ingin disampaikan dalam token seperti informasi pengguna, hak akses, waktu kadaluarsa, atau klaim kustom lainnya (Nashikhuddin dkk., 2023). Terdapat tiga tipe klaim, yaitu *reserved*, *public*, dan *private*.

Gambar 3. JWT *Payload* (www.jwt.io).

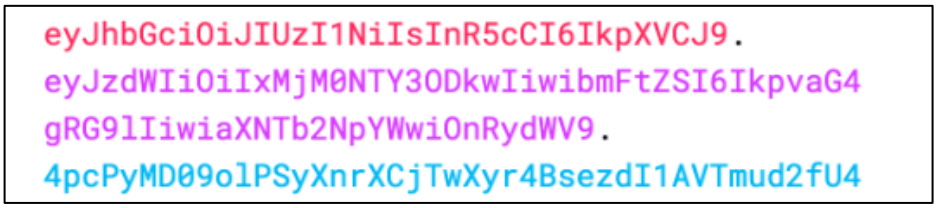
3. *Signature*

   Signature digunakan untuk memverifikasi bahwa pesan tidak diubah selama proses berlangsung. *Signature* juga dapat memverifikasi kebenaran pengirim JWT. Untuk membuat *signature* perlu *header* yang telah dikode, *payload* yang telah dikode, sebuah *secret*, algoritma yang ditentukan dalam *header*, dan menandatanganinya.



Gambar 4. JWT *Signature* (www.jwt.io).

Penggabungan dari tiga bagian tersebut akan menghasilkan sebuah JSON *Web Token* yang aman dan dapat diverifikasi.



Gambar 5. JSON *Web Token* (www.jwt.io).

2.3.10. Axios

Berdasarkan situs resminya, Axios adalah klien HTTP berbasis *promise* yang dapat digunakan di Node.js maupun di *browser*. Axios bersifat isomorfik, artinya dapat berjalan di lingkungan *server* dan klien dengan kode yang sama. Di sisi *server*, Axios memanfaatkan modul HTTP bawaan Node.js, sedangkan di sisi klien (*browser*) menggunakan XMLHttpRequest.

2.4. *Database*

Database atau basis data merupakan sekumpulan data yang saling terhubung dan diorganisir sedemikian rupa sehingga dapat dimanfaatkan kembali dengan cepat dan mudah. Untuk dapat berinteraksi dengan *database*, diperlukan sebuah perangkat lunak yang disebut *Database Management System* (DBMS). Dengan DBMS, pengguna dapat mengelola, memelihara, dan mengakses data dengan praktis dan efisien (Rachmadi dan Kom, 2020). Adapun DBMS yang digunakan untuk mengelola basis data relasional disebut *Relational Database Management System* (RDBMS).

Dalam model basis data relasional (*relational database*), data disimpan dalam bentuk tabel yang berhubungan satu sama lain. Setiap tabel merepresentasikan suatu entitas dan hubungan satu entitas dengan yang lain dijelaskan melalui *key*. Hal ini menawarkan fleksibilitas pengelolaan karena data disajikan dalam bentuk *tables*, *views*, *rows*, dan *columns*. Aturan-aturan juga dapat diterapkan sesuai

dengan kebutuhan sehingga dapat menghindari adanya data yang tidak konsisten, duplikasi data, kehilangan data dan lain sebagainya.

2.4.1. *Object Relational Mapping* (ORM)

ORM merupakan suatu teknik pemetaan objek dalam bahasa pemrograman konvensional ke struktur tabel pada *database* relasional. Dengan menggunakan ORM, manipulasi data dilakukan melalui objek tanpa harus menulis kode SQL secara langsung (Syahputri dan Nasution, 2023). Beberapa ORM yang populer untuk Node.js adalah Sequelize, Mongoose, dan Prisma.

2.4.2. MySQL

MySQL adalah salah satu RDBMS yang paling populer di dunia. Dalam situs Oracle, dikemukakan bahwa MySQL merupakan perangkat lunak yang *open source* sehingga memunginkan siapa saja untuk dapat memodifikasinya. MySQL adalah sistem *client/server* yang terdiri dari *server* SQL *multithread* yang mendukung berbagai *back-end*, beberapa program dan *library* klien yang berbeda, alat administratif, dan beragam antarmuka pemrograman aplikasi (API). MySQL memiliki keunggulan dalam berbagai aspek seperti kemudahan, keandalan, skalabilitas, kinerja, ketersediaan, keamanan, dan fleksibilitas.

2.5. Perancangan Sistem

Perancangan sistem dapat diartikan sebagai proses pendefinisian kebutuhan fungsional, persiapan rancangan untuk implementasi, dan penggambaran bagaimana sistem akan dibentuk, termasuk di dalamnya perencanaan, pembuatan sketsa, dan konfigurasi komponen perangkat lunak dan perangkat keras dari suatu sistem (Nopriandi, 2018).

2.5.1. *Unified Modeling Language* (UML)

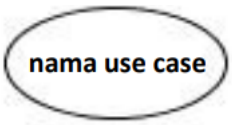
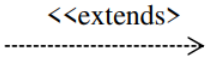
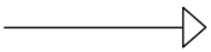
UML adalah bahasa visual menggunakan diagram dan teks-teks pendukung yang digunakan untuk memodelkan dan mengomunikasikan suatu sistem. Penggunaan

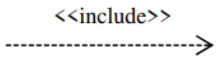
UML yaitu untuk untuk menspesifikasikan, menggambarkan, mambangun dan mendokumentasikan sistem perangkat lunak (Simatupang dan Sianturi, 2019).

2.5.1.1. *Use Case Diagram*

Use case diagram adalah pemodelan untuk menggambarkan perilaku sistem dan interaksi antara satu atau lebih aktor dengan sistem informasi yang sedang dibangun *Use case* digunakan untuk mengidentifikasi fungsi-fungsi yang ada dalam suatu sistem informasi serta menentukan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Simatupang dan Sianturi, 2019). Berikut ini adalah simbol-simbol yang digunakan dalam diagram *use case*, seperti yang ditampilkan pada Tabel 1.

Tabel 1. Simbol *Use Case Diagram*

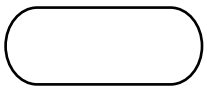
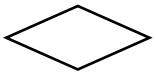
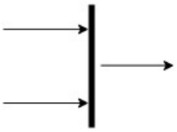
Simbol	Nama	Deskripsi
	*Use case*	Fungsionalitas yang disediakan oleh sistem dan terdiri dari unit-unit yang saling bertukar pesan dengan unit atau aktor lain.
 nama aktor	*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi.
	Association	Komunikasi atau interaksi *use case* dengan *actor*.
	*Extend*	Hubungan dengan *use case* tambahan yang dapat berdiri sendiri.
	*Generalization*	Generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case*.

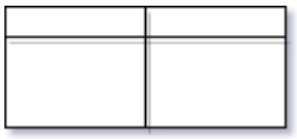
Simbol	Nama	Deskripsi
	*Include*	Hubungan dengan *use case* yang memerlukan *use case* lain untuk menjalankan fungsinya atau sebagai syarat agar dapat dijalankan.

2.5.1.2. *Activity Diagram*

Activity diagram menggambarkan alur kerja atau aktivitas dari suatu sistem, proses bisnis, atau menu dalam perangkat lunak. Fokus dari *activity diagram* adalah menggambarkan aktivitas yang dilakukan oleh sistem, bukan tindakan yang dilakukan oleh aktor (Simatupang dan Sianturi, 2019). Berikut ini adalah simbol-simbol yang digunakan dalam diagram aktivitas, seperti yang ditampilkan pada Tabel 2.

Tabel 2. Simbol *Activity Diagram*

Simbol	Nama	Deskripsi
	*Initial Node*	Status awal dari aktivitas sistem.
	*Activity*	Aktivitas yang dilakukan sistem dan biasanya diawali dengan kata kerja.
	*Decision*	Percabangan asosiasi saat terdapat lebih dari satu pilihan aktivitas.
	*Join*	Penggabungan asosiasi saat terdapat beberapa aktivitas digabungkan menjadi satu

Simbol	Nama	Deskripsi
	*Final Node*	Status akhir yang dicapai oleh sistem.
	*Swimlane*	Memisahkan bagian organisasi bisnis yang bertanggung jawab atas terjadinya aktivitas dijalankan.

2.5.2. Desain *Wireframe*

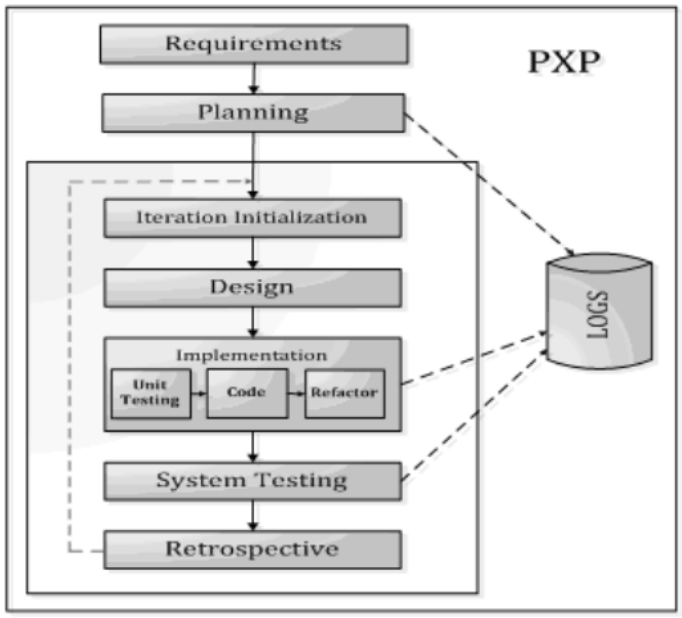
Desain *wireframe* adalah gambaran awal dari suatu desain dan produk sebelum diimplementasikan. *Wireframe* digunakan untuk mengatur konten dan fungsionalitas pada halaman suatu sistem atau aplikasi dengan mempertimbangkan kebutuhan dan pengalaman pengguna (Irwansyah dkk., 2021). Tujuan dari dilakukannya proses desain *wireframe* adalah untuk menghasilkan gambaran visual dasar yang dapat membantu dalam tahap implementasi.

Terdapat banyak alat yang dapat digunakan untuk membuat desain *wireframe*. Salah satunya yaitu Whimsical. Dirujuk dari situs RevoU, Whimsical adalah platform kolaborasi yang dibuat untuk mempercepat proses kreatif dalam proyek, terutama dalam desain dan pengembangan produk. Platform ini memiliki banyak kegunaan lainnya seperti untuk membuat *mind maps*, *flowchart*, dan *sticky notes*.

2.6. Metode *Personal Extreme Programming* (PXP)

Personal Extreme Programming (PXP) merupakan proses pengembangan perangkat lunak yang dirancang untuk *software engineer* yang bekerja secara individu. Metode ini termasuk dalam pendekatan *Agile* dan merupakan turunan dari metode *Extreme Programming*. PXP bertujuan untuk meringankan proses pengembangan perangkat lunak dan membuatnya lebih mudah untuk diikuti, dengan tetap menjaga prinsip dasar PSP (*Personal Software Process*). Proses pengembangan PXP bersifat iteratif dan memungkinkan pengembangan yang lebih

fleksibel dan responsif terhadap perubahan (Dzhurov et al., 2009). Tahapan-tahapan pada metode PXP dapat dilihat pada Gambar 6.



Gambar 6. Tahapan Proses PXP (Dzhurov et al., 2009).

Proses pada PXP bersifat iteratif dan terdiri dari beberapa iterasi dan siklus *inline*. Tahap *requirements* dan *planning* dilakukan pada awal pengembangan karena persyaratan untuk keseluruhan aplikasi biasanya ditetapkan terlebih dahulu. Namun jika terjadi perubahan pada persyaratan, bagian rencana kerja masih dapat direvisi. Berikut ini adalah penjabaran setiap tahapan pada metode PXP (Dzhurov et al., 2009):

1. *Requirements*

   Requirements merupakan tahap awal dalam proses pengembangan PXP. Pada tahap ini dilakukan identifikasi kebutuhan fungsional dan non-fungsional dari sistem yang akan dikembangkan.

2. *Planning*

   Di tahap *planning*, pengembang akan menyusun serangkaian tugas berdasarkan persyaratan yang telah ditentukan dan menetapkan estimasi waktu pengerjaan dari setiap tugas. Pada tahap ini juga dapat dilakukan

perencanaan hal lainnya seperti bahasa pemrograman yang digunakan, *framework* pengembangan, dan sebagainya.

3. *Iteration Initialization*

   Tahap *iteration initialization* merupakan awal dari proses iterasi. Proses dimulai dengan pemilihan tugas yang akan menjadi fokus iterasi.

4. *Design*

   Pada tahap *design* pengembang akan melakukan pemodelan modul dan kelas yang akan diimplementasikan pada iterasi yang sedang berjalan. Desain yang dibuat oleh pengembang hanya ditujukan untuk memenuhi persyaratan *client* saat ini tanpa menebak kebutuhan di masa yang akan datang.

5. *Implementation*

   Implementation merupakan tahap pembuatan dan pengujian kode sesuai dengan rencana pada tahap *design*. Tahap ini terdiri dari 3 subfase yaitu *unit testing, code generation,* dan *code refactoring.*

6. *System Testing*

   Semua fitur yang telah berhasil dibuat hingga tahap ini akan diuji untuk memastikan solusi yang diterapkan sudah sesuai dengan persyaratan. Jika masih terdapat kesalahan maka akan dilakukan perbaikan.

7. *Retrospective*

   Iterasi diakhiri dengan tahap *retrospective*. Pengembang akan melakukan evaluasi terhadap apa yang sudah dikerjakan, termasuk memverifikasi waktu yang diestimasikan dengan waktu yang sebenarnya untuk menemukan alasan penundaan dan keterlambatan. Jika masih terdapat persyaratan yang belum diimplementasikan, proses selanjutnya adalah memulai iterasi baru dan kembali ke tahap *iteration initialization*. Namun jika semua persyaratan telah dipenuhi, proses pengembangan dapat diakhiri pada tahap ini.

2.7. Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan proses mengeksekusi program untuk menemukan kesalahan. Pengujian bertujuan untuk mencari kesalahan (*bug error*) sebanyak mungkin dan menemukan kesalahan sebelumnya yang tidak ditemukan. Selain itu, pengujian juga dapat mengurangi resiko yang ada dalam sistem komputer. Umumnya pengujian dilakukan untuk menentukan perbedaan antara hasil yang diperoleh dengan hasil yang diharapkan (Adi dkk., 2020). Tahap pengujian merupakan hal yang sangat penting dalam pengembangan perangkat lunak dan dapat menentukan kualitas dari perangkat lunak itu sendiri. Perangkat lunak yang telah diuji sebelumnya cenderung memiliki kemungkinan kesalahan yang lebih kecil, sehingga dapat mengurangi risiko kegagalan ketika sistem sudah digunakan.

2.7.1. Pengujian Fungsional

Pengujian fungsional adalah pengujian perangkat lunak yang berfokus pada layanan yang harus disediakan sistem, termasuk cara sistem berinteraksi terhadap input dan bagaimana sistem seharusnya berperilaku pada kondisi atau situasi tertentu (Fatiyah dkk., 2019). Pengujian fungsional bertujuan untuk menguji fungsi atau fitur dalam sistem guna memastikan fungsi atau fitur tersebut berjalan dengan baik dan dapat melaksanakan tugas sesuai spesifikasi yang telah ditentukan. Dalam konteks API, pengujian fungsional dapat digunakan untuk mengecek *response* yang dikembalikan suatu *endpoint* dengan hasil yang diharapkan. Pengujian fungsionalitas API dapat dilakukan dengan berbagai cara, namun yang paling populer di kalangan *developer* saat ini adalah dengan menggunakan platform Postman.

Menurut *website* Postman, platform ini digunakan untuk membangun dan menggunakan API. Postman menyediakan berbagai alat yang membantu dalam pengelolaan API, seperti melakukan permintaan HTTP, dokumentasi API, kolaborasi tim, dan pengujian *endpoint*. Dengan postman, pengujian dapat dilakukan secara manual maupun otomatis dengan menggunakan skrip. Platform

ini memiliki popularitas yang tinggi dalam pengembangan perangkat lunak karena penggunaannya yang mudah.

2.7.2. Pengujian Non Fungsional

Pengujian non fungsional merupakan suatu pengujian yang tidak secara langsung menggambarkan layanan spesifik dalam sistem. Non fungsional sistem umumnya menggambarkan kualitas perangkat lunak pada aspek *development process*, *standard*, dan waktu (Fatiyah dkk., 2019). Pengujian non fungsional yang dilakukan penulias meliputi pengujian keamanan dan kinerja.

2.7.2.1. Pengujiian Keamanan

Pengujian keamanan dilakukan untuk mengidentifikasi ancaman dan potensi ketentanan yang mungkin timbul dan dapat mengakibatkan hilangnya informasi. Pengujian keamanan memeriksa sejauh mana kelemahan pada mekanisme keamanan yang diimplementasikan pada sistem (Siregar, 2020).

Salah satu alat untuk pengujian keamanan adalah *Zed Attack Proxy* (ZAP). Berdasarkan situs www.zaproxy.org, ZAP bersifat *open source* dan gratis untuk menguji aplikasi web. Aplikasi ini dirancang untuk orang-orang dengan berbagai tingkat keahlian sehingga ideal untuk pengembang atau penguji yang baru mengenal pengujian keamanan. Selain itu, untuk menguji keamanan pada mekanisme autentikasi dan otorisasi berbasis token dapat dilakukan dengan Postman. Pengujian ini bertujuan untuk memastikan hanya token yang valid yang dapat mengakses sumber daya.

2.7.2.2. Pengujian Kinerja

Pengujian kinerja dilakukan untuk mengetahui kinerja dari sebuah komputer, jaringan, perangkat lunak, atau perangkat di bawah kecepatan beban kerja, responsivitas, dan stabilitas yang ditentukan (Rafiq dkk., 2020). Pengujian kinerja dapat dilakukan menggunakan alat seperti LoadRunner, Apache, JMeter, WebLOAD, dan lain-lain.

Pada situs resminya, Apache JMeter didefinisikan sebagai perangkat lunak *open-source* berbasis Java yang berfungsi untuk memuat perilaku fungsional pengujian dan mengukur kinerja. Apache JMeter dapat digunakan untuk mensimulasikan beban berat pada *server*, sekelompok *server*, jaringan atau objek. Hal ini dilakukan untuk menguji kekuatan dan menganalisis kinerja aplikasi di bawah jenis beban yang berbeda. Tidak hanya web, alat ini juga dapat menguji berbagai jenis aplikasi, *server*, atau protokol. Pengujian kinerja dilakukan dengan cara mensimulasikan banyak pengguna yang mengakses sistem secara bersamaan dengan rentang waktu yang telah ditentukan. Pada Apache JMeter, hal ini diatur pada konfigurasi *Number of Threads* dan *Ramp-Up Period*. *Number of Threads* mensimulasikan jumlah pengguna atau koneksi ke aplikasi server, sementara *Ramp-Up Period* menentukan berapa lama waktu yang dibutuhkan JMeter untuk menjalankan semua *thread* (Halili, 2008).

2.7.3. *User Acceptance Testing* (UAT)

User Acceptance Testing (UAT) merupakan pengujian yang dilakukan oleh *end-user* yang langsung berinteraksi dengan sistem untuk memverifikasi apakah fungsi yang dibangun telah berjalan sesuai dengan kebutuhan atau fungsinya (Perry, 2006). Pengujian ini bertujuan untuk memastikan perangkat lunak mampu memenuhi kebutuhan pengguna. Tidak hanya sebatas memenuhi spesifikasi sistem, melainkan untuk memvalidasi apakah sistem dapat diterima atau tidak (Hady dkk., 2020). UAT dilakukan pada tahap terakhir dari serangkaian proses pengujian perangkat lunak.

Metode yang digunakan dalam UAT pada penelitian ini adalah skala likert. Skala pengukuran ini dikembangkan oleh Rensis Likert pada tahun 1932. Bentuk kuesioner dengan skala likert biasanya berupa pernyataan yang dilengkapi skala pengukuran terkait pilihan sikap responden terhadap pernyataan yang diberikan. (Suasapha, 2020). Jawaban dari setiap item instrumen memiliki rentang dari sangat positif hingga sangat negatif. Analisis kuesioner dilakukan dengan membandingkan jumlah skor perolehan dengan jumlah skor maksimal yang ditetapkan, sebagaimana tercantum dalam rumus berikut (Y. I. Kurniawan dan Kusuma, 2021).

$$P = \frac{f}{N} \times 100\%$$

Keterangan:

P = Skor presentase yang dicari

f = Perolehan skor validator

N = Skor maksimal

Penilaian hasil pengujian didasarkan pada indikator degradasi kategori yang ditunjukkan dalam Tabel 3.

Tabel 3. Indikator Kategori Penilaian

Nilai P	**Kategori**
0% – 20%	Sangat Buruk
20,01% – 40%	Buruk
40,01% – 60%	Cukup
60,01% – 80%	Baik
80,01% – 100%	Sangat Baik

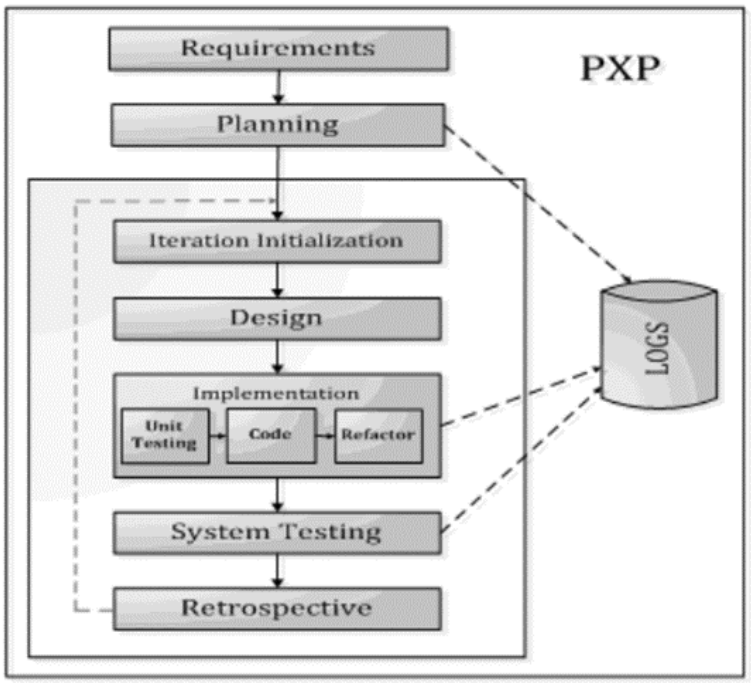
III. METODOLOGI PENELITIAN

3.1. *Executive Summary*

Pada proyek ini akan dikembangkan sebuah aplikasi sebagai platform digital untuk memfasilitasi dan memudahkan proses pengelolaan data dan pemasaran produk usaha mahasiswa FMIPA Universitas Lampung. Pada aplikasi ini mahasiswa wirausaha (*Member*) di FMIPA dapat melakukan interaksi dengan pelanggan (*Customer*) di lingkungan Universitas Lampung dan sekitarnya. Aplikasi berbasis web ini dirancang untuk menjadi sebuah solusi dari masalah yang ditemui pada aktivitas yang berjalan saat ini dengan menyediakan layanan untuk pengelolaan data, pemasaran, pembelian, dan pencatatan dalam satu sistem yang terintegrasi. Dengan aplikasi ini, mahasiswa dapat memperluas jangkauan pemasaran produk, terutama di lingkungan Universitas Lampung dan sekitarnya, serta mempermudah proses pengelolaan bisnis. Platform ini diharapkan dapat meningkatkan efisiensi operasional dan membantu mahasiswa dalam beradaptasi dengan perkembangan bisnis digital yang semakin pesat.

Metodologi yang digunakan adalah *Personal Extreme Programming* karena proyek ini dikerjakan oleh pengembang individu dan mempunyai karakteristik yang fleksibel terhadap perubahan (Dzhurov et al., 2009). Iterasi membantu memecah proyek menjadi bagian-bagian yang lebih kecil sehingga mudah dikelola dan bersifat dinamis atau dapat diulang jika terdapat kekurangan atau kesalahan (Suprapto dkk., 2024). PXP memungkinkan pengembang untuk bekerja dalam siklus iterasi pendek, sehingga proses identifikasi dan perbaikan masalah dapat dilakukan secara cepat serta memungkinkan penyesuaian terhadap kebutuhan pengguna yang mungkin berubah selama proses pengembangan. Selain itu, dengan

adanya perencanaan tugas dengan estimasi waktu tertentu memungkinkan pengembang untuk bekerja secara lebih efisien dan terstruktur.



Gambar 7. Metode *Personal Extreme Programming*.

Proses pada PXP bersifat iteratif dan terdiri dari beberapa iterasi dan siklus *inline*. Tahap *requirements* dan *planning* dilakukan pada awal pengembangan karena persyaratan untuk keseluruhan aplikasi biasanya ditetapkan terlebih dahulu. Namun jika terjadi perubahan pada persyaratan, bagian rencana kerja masih dapat direvisi. Berikut ini adalah penjabaran setiap tahapan pada metode PXP (Dzhurov et al., 2009).

1. ***Requirements***
    - Pada tahap ini, pengembang melakukan identifikasi kebutuhan fungsional serta kebutuhan non fungsional yang harus dipenuhi oleh aplikasi dengan berbagai metode pengumpulan data seperti wawancara dan observasi.
    - **Hasil**: Daftar kebutuhan fungsional dan non fungsional, yang akan menjadi acuan dalam pengembangan aplikasi.

2. ***Planning***
    - Tahap ini melibatkan penyusunan rencana kerja yang terperinci, termasuk pembagian proyek menjadi beberapa iterasi, penentuan tugas yang harus diselesaikan pada setiap iterasi, estimasi waktu yang dibutuhkan untuk masing-masing tugas, dan perangkat yang akan digunakan.
    - **Hasil:** Rincian iterasi, tugas-tugas yang akan dilakukan pada setiap iterasi beserta estimasi waktunya, dan daftar perangkat lunak dan keras.

3. ***Iteration Initialization***
    - Pengembang menentukan tugas yang menjadi fokus utama untuk iterasi yang akan berjalan, misalnya fokus pada pengembangan modul-modul tertentu yang dimulai dari tahap desain hingga pengujian.
    - **Hasil:** Daftar tugas yang akan dikerjakan dalam iterasi,

4. ***Design***
    - Pengembang membuat desain teknis untuk aplikasi, termasuk pembuatan desain *database*, desain API, atau desain *wireframe*, tergantung fokus iterasi yang sedang dijalankan. Desain ini harus memenuhi semua kebutuhan yang telah diidentifikasi pada tahap sebelumnya.
    - **Hasil**: Daftar desain *database*, desain API, dan desain *wireframe*, yang semuanya akan digunakan sebagai acuan dalam tahap implementasi.

5. ***Implementation***
    - Pada tahap ini, pengembang mulai mengubah desain yang telah dibuat menjadi kode sumber yang dapat dieksekusi. Ini meliputi pengembangan fitur-fitur utama yang telah direncanakan, penyiapan *database*, penulisan kode untuk API, dan implementasi antarmuka pengguna.
    - **Hasil:** Kode sumber yang berfungsi sesuai dengan desain dan kebutuhan yang telah ditetapkan.

6. ***System Testing***
    - Setelah implementasi selesai, sistem diuji untuk memastikan bahwa semua fungsi bekerja dengan benar dan sesuai dengan spesifikasi yang telah ditetapkan. Pengujian ini mencakup uji fungsional, uji keamanan, dan uji kinerja, tergantung fokus iterasi yang dijalankan.
    - **Hasil**: Dokumentasi pengujian yang mencakup hasil pengujian untuk setiap fungsi, serta ringkasan hasil pengujuan kinerja dan keamanan sistem.

7. ***Retrospective***
    - Di akhir iterasi, pengembang melakukan evaluasi terhadap proses yang telah dilakukan. Ini mencakup refleksi terhadap apa yang berjalan dengan baik, tantangan yang dihadapi, dan poin yang bisa ditingkatkan untuk iterasi berikutnya.
    - **Hasil:** Laporan evaluasi yang mencakup temuan dari proses retrospektif, serta rekomendasi untuk peningkatan di iterasi atau proyek berikutnya.

3.2. *Project Sponsor*

Project sponsor pada pengembangan aplikasi ini adalah Tim Kewirausahaan FMIPA Universitas Lampung yang beralamat di Jalan Prof. Dr. Ir. Sumantri Brojonegoro, Gedong Meneng, Kecamatan Rajabasa, Kota Bandar Lampung. Sebuah komunitas bagi para mahasiswa wirausaha di FMIPA Universitas Lampung yang dipimpin oleh Ibu Dr. Ilim., M.S. Mereka melihat peluang untuk menciptakan solusi inovatif untuk memfasilitasi dan membantu mahasiswa dalam pemasaran digital dengan membangun sebuah aplikasi pemasaran yang dapat diakses dengan mudah oleh para pelanggan.

3.3. *Business Needs*

Kebutuhan bisnis yang ingin dipenuhi antara lain:

1. Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung memerlukan platform digital terpusat untuk memasarkan dan mengelola produk mahasiswa secara efisien. Saat ini, pemasaran dilakukan secara individual melalui media sosial, yang membatasi jangkauan dan efektivitas.
2. Dibutuhkan sistem yang memungkinkan pengelolaan data dilakukan dalam satu platform yang terintegrasi. Saat ini, data dikelola secara manual di berbagai media yang menyebabkan ketidakefisienan.

3.4. Tahapan Penelitian

Berikut ini adalah tahapan penelitian yang dilakukan sesuai metode *Personal Extreme Programming.*

3.4.1. *Requirements*

Tahap ini terdiri dari beberapa proses yang meliputi identifikasi *Business Requirement, Requirement Definition, Requirement Analysis Strategy,* dan *Functional Model*.

3.4.1.1. *Business Requirement*

Kebutuhan bisnis pengembangan aplikasi ini mencakup kebutuhan fungsional dan non fungsional dari sistem yang akan dibuat. Adapun daftar kebutuhan non fungsional yaitu:

1. Aplikasi harus mampu melindungi data dan informasi dari akses yang tidak sah melalui autentikasi dan otorisasi pengguna berdasarkan kredensial serta aman dari risiko kerentanan yang dapat menyerang sistem.
2. Aplikasi harus bekerja secara stabil dan konsisten dalam menangani banyak permintaan dari pengguna serta mampu memberikan respons yang cepat dengan waktu tidak lebih dari 10 detik.

Sedangkan daftar kebutuhan fungsional adalah sebagai berikut:

1. Aplikasi harus menyediakan fitur registrasi pengguna untuk peran *Member* dan *Customer*.
2. Aplikasi harus dapat menangani proses *login* pengguna sesuai kredensial.
3. Aplikasi harus menyediakan fitur *logout* untuk keluar dari sistem.
4. Admin harus dapat mengelola data *Member*.
5. Admin harus dapat mengelola data *Customer*.
6. Admin harus dapat mengelola data kategori.
7. Member harus dapat mengelola data produk.
8. *Member* harus dapat mengelola data metode pembayaran.
9. *Member* harus dapat mengelola data metode pengiriman.
10. Aplikasi harus dapat menampilkan daftar produk, menampilkan detail produk, serta menyediakan fitur pencarian dan penyortiran.
11. *Customer* harus dapat memasukkan produk ke keranjang, mengubah jumlah item, dan menghapus produk dari keranjang.
12. *Customer* harus dapat melakukan pemesanan produk yang ada di keranjang, melihat detail, membatalkan, dan menyelesaikan pesanan.
13. *Member* harus dapat menerima, melihat detail, dan mengubah status pesanan dan pembayaran yang masuk.
14. Aplikasi harus mampu mencatat riwayat pesanan baik di akun *Member* dan *Customer*.
15. *Customer* yang telah menyelesaikan proses pembelian dapat memberikan ulasan pada produk yang dibeli.
16. *Member* dan *Customer* dapat memperbarui informasi profil.
17. Aplikasi harus menyediakan fitur untuk menangani kasus lupa *password*.

3.4.1.2. *Requirement Definitions*

Definisi kebutuhan non fungsional aplikasi adalah sebagai berikut:

1. Keamanan: Aplikasi harus memiliki fitur autentikasi dan otorisasi pengguna sesuai kredensial yang diberikan untuk memastikan hanya pengguna yang berwenang yang dapat mengakses data dan fitur tertentu. Aplikasi juga harus aman dari risiko kerentanan yang dapat menyerang sistem.

2. Kinerja: Aplikasi harus dapat menangani banyak permintaan dengan stabil dan memiliki performa yang cepat, dengan waktu mendapatkan, menambah, mengedit, dan menghapus data tidak lebih dari 10 detik.

Sedangkan definisi kebutuhan fungsional aplikasi adalah sebagai berikut:

1. Pendaftaran Akun Pengguna: Sistem harus menyediakan fitur registrasi akun sebagai *Member* dan *Customer* yang digunakan untuk *login* ke dalam sistem.
2. Proses *Login:* Sistem harus dapat menangani proses *login* pengguna dengan memverifikasi kredensial yang diberikan, sehingga pengguna dapat mengakses akun mereka sesuai dengan peran (*role*) yang telah ditentukan.
3. Fitur *Logout*: Sistem harus menyediakan fitur *logout* yang memungkinkan pengguna untuk keluar dari akun dengan aman, menjaga privasi dan keamanan informasi mereka.
4. Manajemen Akun *Member* oleh Admin: Admin harus dapat memverifikasi status registrasi *Member* untuk memastikan integritas dan keakuratan data pengguna di sistem. Admin juga dapat mengedit dan menghapus data akun.
5. Manajemen Akun *Customer* oleh Admin: Admin harus dapat mengelola data *Customer*, termasuk mengedit dan menghapus data.
6. Manajemen Kategori oleh Admin: Admin harus dapat mengelola data kategori, termasuk menambah, mengedit, dan menghapus data.
7. Manajemen Produk oleh *Member*: *Member* harus dapat mengelola data produk termasuk menambah, mengedit, mengubah status, dan menghapus produk yang tersedia dalam sistem.
8. Manajemen Metode Pembayaran oleh *Member*: *Member* harus dapat mengelola metode pembayaran termasuk menambah, dan menghapus data.
9. Manajemen Metode Pengiriman oleh *Member*: *Member* harus dapat mengelola metode pengiriman termasuk menambah, mengedit, dan menghapus data.
10. Penyajian Katalog Produk, Pencarian, dan Penyortiran: Aplikasi dapat menampilkan seluruh produk yang tersedia, melakukan pencarian produk berdasarkan nama, menyortir produk berdasarkan kategori yang dipilih, serta menampilkan detail produk.

11. Manajemen Keranjang Belanja: *Customer* harus dapat memasukkan produk ke dalam keranjang belanja, mengubah jumlah item, dan menghapus produk dari keranjang sebelum melakukan pembelian.
12. Pemesanan Produk: *Customer* harus dapat melakukan pemesanan produk yang ada di keranjang, melihat detail pesanan, membatalkan pesanan jika pesanan belum diproses, dan menyelesaikan pesanan ketika pesanan telah selesai diproses.
13. Manajemen Pesanan oleh *Member*: *Member* harus dapat menerima pesanan yang masuk, melihat detail pesanan, serta mengubah status pesanan dan pembayaran yang dilakukan oleh *Customer*.
14. Pencatatan Riwayat Pesanan: Sistem harus mampu mencatat dan menyimpan riwayat pesanan baik di akun *Member* maupun *Customer*, sehingga pengguna dapat mengakses informasi terkait transaksi sebelumnya.
15. Ulasan Produk: *Customer* yang telah menyelesaikan proses pembelian harus dapat memberikan ulasan pada produk yang mereka beli, untuk membantu *Member* meningkatkan kualitas produk dan memberikan informasi kepada *Customer* lainnya.
16. Pembaruan Informasi Profil: Sistem harus dapat memperbarui informasi profil akun *Member* dan *Customer*.
17. Pemulihan Kata Sandi: Sistem harus menyediakan fitur untuk menangani kasus lupa *password*, memungkinkan pengguna untuk memulihkan akses ke akun mereka melalui langkah-langkah yang aman.

3.4.1.3. *Requirement Analysis Strategy*

Strategi atau metode yang digunakan untuk mengidentifikasi kebutuhan terdiri dari wawancara dan observasi yang hasilnya dijabarkan pada poin berikut.

1. **Hasil Kegiatan Wawancara**

   Berdasarkan hasil wawancara, ditemukan bahwa proses bisnis yang dilakukan oleh para mahasiswa wirausaha FMIPA menghadapi beberapa kendala seperti jangkauan pemasaran masih bergantung pada jumlah teman

atau relasi yang dimiliki sehingga target konsumen yang bahkan berada di lingkungan sekitar tidak mengetahui informasi produk yang ditawarkan. Pemasaran *online* yang dilakukan secara individual melalui media sosial seperti WhatsApp, Instagram, dan TikTok belum sepenuhnya optimal karena kurangnya pengetahuan tentang strategi pemasaran yang efektif untuk setiap platform serta tingginya persaingan dengan pelaku usaha yang sudah memiliki nama besar dan telah lama beroperasi. Kendala lainnya adalah mekanisme pengelolaan katalog di Instagram sulit untuk dilakukan. Pemesanan juga masih dilakukan melalui WhatsApp atau DM media sosial sehingga detail dan riwayat pesanan tidak tercatat secara otomatis dan masih dilakukan secara manual di buku, Spreadsheet, atau aplikasi catatan.

2. Hasil Kegiatan Observasi

a. Proses Pencarian dan Pemesanan:

***Customer*:** Sulit menemukan informasi lengkap tentang berbagai produk yang dijual, karena masing-masing *Member* menjual secara terpisah secara individu. Pemesanan dilakukan secara manual melalui WhatsApp atau Direct Message di media sosial. *Customer* harus menanyakan ketersediaan produk terlebih dahulu dan menunjukkan *screenshoot* foto produk yang akan dipesan. Kurangnya fitur pencarian dan *filter* yang efektif memperburuk pengalaman pelanggan, terutama ketika mencari produk di sekitar lingkungan kampus.

***Member*:** Kesulitan dalam pemasaran *online* produk secara efektif, sering kali mengandalkan bantuan teman dan relasi. Ada juga kesulitan dalam mencatat detail dan riwayat pesanan yang diterima secara manual, yang berpotensi menimbulkan kesalahan dalam pengelolaan. Belum adanya sistem terpusat untuk pengelolaan produk menyebabkan proses menjadi kurang efisien dan keterbatasan jangkauan pemasaran.

b. Pembayaran dan Pengiriman

***Customer*:** Mengalami kebingungan dalam mengetahui informasi metode pembayaran dan pengiriman yang disediakan karena terkadang

informasi tidak dicantumkan dan masing-masing penjual memiliki opsi berbeda. Hal ini membuat pelanngan harus harus menanyakannya terlebih dahulu, yang dapat memakan waktu dan menyebabkan ketidakpastian.

***Member*:** Bukti pembayaran yang dikirimkan melalui *chat* rentan hilang atau terhapus. Selain itu, sering terjadi miskomunikasi terkait metode pengiriman, misalnya ketika *Member* tidak menyediakan opsi *delivery*, tetapi *Customer* meminta pesanan untuk diantarkan.

c. Pengelolaan dan Penyajian Produk:

***Customer*:** Kesulitan dalam menemukan produk yang tersedia di sekitar lingkungan kampus terjadi karena kurangnya fitur cari dan filter yang memadai di media sosial. Ketika kata kunci pencarian dimasukkan, hasil yang muncul cenderung menampilkan akun-akun penjual yang sudah populer dan berasal dari berbagai tempat.

***Member*:** Kesulitan dalam mengelola dan memperbarui katalog produk secara *real-time*. Banyak usaha yang masih bergantung pada pengelolaan manual, yang meningkatkan risiko kesalahan. Tidak adanya sistem otomatis untuk pengelolaan katalog produk secara terpusat menyebabkan kendala dalam menjaga informasi produk yang selalu *up-to-date* dan mudah diakses oleh *Customer*.

Berikut ini adalah pemetaan kebutuhan fungsional dengan proses *As-Is* dan proses *To-Be* yang disajikan pada Tabel 4.

Tabel 4. Pemetaan Kebutuhan Fungsional

No	Proses Berjalan (*As-Is*)	Sistem yang Akan Dibuat (*To-Be*)
1	**Pendaftaran Akun Pengguna**: Tidak ada pendaftaran anggota	**Pendaftaran Akun Pengguna**: Aplikasi menyediakan fitur

	(*Member*) dan pelanggan (C*ustomer*). Untuk bergabung dalam tim, mahasiswa mendapat informasi yang dari dosen dan bergabung dalam grup WhatsApp.	pendaftaran otomatis bagi *Member* dan *Customer*. Pendaftaran terbuka bagi siapapun yang memenuhi syarat dan dapat dilakukan dengan mudah langsung melalui sistem.
2	***Login***: Tidak ada sistem *login* dan *logout*. Anggota hanya bergabung dan berpartisipasi dalam kegiatan melalui grup WhatsApp.	***Login***: Aplikasi menyediakan fitur *login* yang aman seusai kredensial yang diberikan untuk mengakses akun.
3	***Logout:*** Tidak ada fitur *logout* karena tidak ada sistem yang terintegrasi. Pengguna hanya meninggalkan grup atau berhenti berpartisipasi.	***Logout:*** Aplikasi juga menyediakan fitur *logout* yang memungkinkan pengguna keluar dari akun untuk menjaga privasi dan keamanan informasi.
4	**Verifikasi dan Manajemen Data *Member***: Tidak ada verifikasi resmi bagi anggota yang bergabung. Pencatatan dilakukan secara manual dengan menulis data pada list di grup WhatsApp secara bergantian.	**Verifikasi dan Manajemen Data *Member***: Admin dapat memverifikasi registrasi *Member* dan mengelola data akun melalui sistem, sehingga proses pencatatan dapat secara otomatis dilakukan dan dapat dimonitor dengan mudah.
5	**Manajemen Data *Customer*:** Tidak ada proses pengelolaan data *Customer*. *Customer* yang belum diketahui identitasnya secara langsung menghubungi penjual terkait.	**Manajemen Data *Customer:*** Aplikasi memungkinkan Admin mengelola data *Customer*, sehingga identitas pelanggan dapat teridentifikasi dengan jelas agar menghindari penipuan dalam pemesanan.

6	**Manajemen Kategori oleh Admin:** Kategori produk tidak dikelola secara resmi. *Member* memasarkan produk mereka sendiri tanpa klasifikasi yang jelas.	**Manajemen Kategori Produk oleh Admin**: Aplikasi memungkinkan Admin mengelola kategori yang tersedia, memudahkan *Customer* dalam mengelompokkan produk berdasarkan kategorinya.
7	**Manajemen Produk oleh *Member***: Pengelolaan produk dilakukan secara manual oleh *Member* melalui komunikasi di media sosial atau WhatsApp.	**Manajemen Produk oleh *Member***: *Member* dapat mengelola produk secara terpusat dalam aplikasi, termasuk menambah, mengedit, dan menghapus data terkait.
8	**Manajemen Metode Pembayaran oleh *Member***: Informasi metode pembayaran seringkali tidak dicantumkan sehingga *Customer* harus menanyakan metode apa saja yang disediakan secara manual.	**Manajemen Metode Pembayaran oleh *Member***: Terdapat informasi metode pembayaran yang disediakan oleh *Member* terkait, sehingga *Customer* dapat langsung memilih metode yang ingin digunakan tanpa harus menanyakannya dahulu.
9	**Manajemen Metode Pengiriman oleh *Member***: Informasi metode pengiriman seringkali tidak dicantumkan, sehingga *Customer* harus menanyakan metode apa saja yang disediakan secara manual.	**Manajemen Metode Pengiriman oleh *Member***: Terdapat informasi metode pengiriman yang *disediakan* oleh *Member* terkait, sehingga *Customer* dapat langsung memilih metode yang ingin digunakan tanpa harus menanyakannya dahulu.
10	**Penyajian, Pencarian, dan Penyortiran Produk**: *Member* menyajikan katalog produk yang dijual menggunakan Instagram	**Pencarian dan Penyortiran Produk**: Aplikasi menyediakan halaman katalog dengan fitur pencarian produk berdasarkan nama

	sehingga menyulitkan *Customer* untuk mencari dan menyortir produk karena penjualan dilakukan secara terpisah oleh setiap *Member*.	dan penyortiran berdasarkan kategori, sehingga memudahkan *Customer* menemukan produk yang diinginkan.
11	**Manajemen Keranjang Belanja**: Tidak ada sistem keranjang belanja. *Customer* harus mencatat produk yang diinginkan secara manual atau melalui *chat*.	**Manajemen Keranjang Belanja**: Aplikasi menyediakan fitur keranjang belanja yang memungkinkan *Customer* dapat menambahkan produk, mengubah jumlah item, dan menghapus produk sebelum melakukan pembelian.
12	**Pemesanan Produk**: Pesanan dilakukan melalui *chat* (WhatsApp/DM), dan *Customer* harus mengirim gambar produk yang ingin dipesan dan menanyakan ketersediaan produk secara manual. Tidak ada mekanisme formal untuk membatalkan atau menyelesaikan pesanan. Semua dilakukan melalui kesepakatan langsung dengan *Member*.	**Pemesanan Produk**: Aplikasi memungkinkan *Customer* melakukan pemesanan produk yang ada di keranjang, dengan pilihan metode pembayaran dan pengiriman yang tersedia. Terdapat status produk yang menginformasikan ketersediaan produk. *Customer* dapat membatalkan pesanan jika belum diproses oleh *Member* dan menyelesaikan pesanan setelah diproses, semuanya melalui aplikasi.
13	**Manajemen Pesanan oleh *Member***: *Member* menerima pesanan dan mengelola statusnya secara manual, seringkali melalui catatan pribadi atau Spreadsheet.	**Manajemen Pesanan oleh *Member***: *Member* dapat menerima pesanan, melihat detailnya, serta mengubah status pesanan dan pembayaran langsung di aplikasi.

14	**Pencatatan Riwayat Pesanan**: Riwayat pesanan dicatat secara manual, seringkali di buku atau Spreadsheet, sehingga sering lupa untuk mencatat dan rentan terhadap kesalahan dan kehilangan data.	**Pencatatan Riwayat Pesanan**: Aplikasi secara otomatis mencatat riwayat pesanan baik untuk *Member* maupun *Customer*, sehingga dapat diakses dengan mudah kapan saja.
15	**Ulasan Produk:** *Customer* memberikan ulasan produk melalui *chat* atau komentar di media sosial, yang tidak terstruktur dan sulit diakses kembali. Dengan begitu *Member* harus mempostingnya kembali sebagai testimoni.	**Ulasan Produk:** *Customer* dapat memberikan ulasan produk melalui aplikasi setelah menyelesaikan pembelian, yang kemudian dapat diakses oleh siapapun.
16	**Pembaruan Informasi Profil**: *Member* yang ingin memperbarui informasi pribadi, seperti nama, alamat, nomor telepon, atau kontak sosial, biasanya hanya memperbarui bio atau deskripsi profil di akun Instagram mereka. *Customer* tidak ada proses pembaruan profil.	**Pembaruan Informasi Profil**: Aplikasi memungkinkan *Member* dan *Customer* untuk dapat mengelola profil secara langsung dari akun yang terdaftar di sistem.
17	**Pemulihan Kata Sandi**: Tidak ada mekanisme ini karena pemulihan kata sandi tergantung media sosial yang digunakan.	**Pemulihan Kata Sandi**: Aplikasi menyediakan fitur pemulihan kata sandi yang memungkinkan pengguna memulihkan akses ke akun mereka secara mandiri melalui langkah-langkah yang aman.

Sementara itu, pemetaan kebutuhan non fungsional dengan proses *As-Is* dan proses *To-Be* disajikan pada Tabel 5.

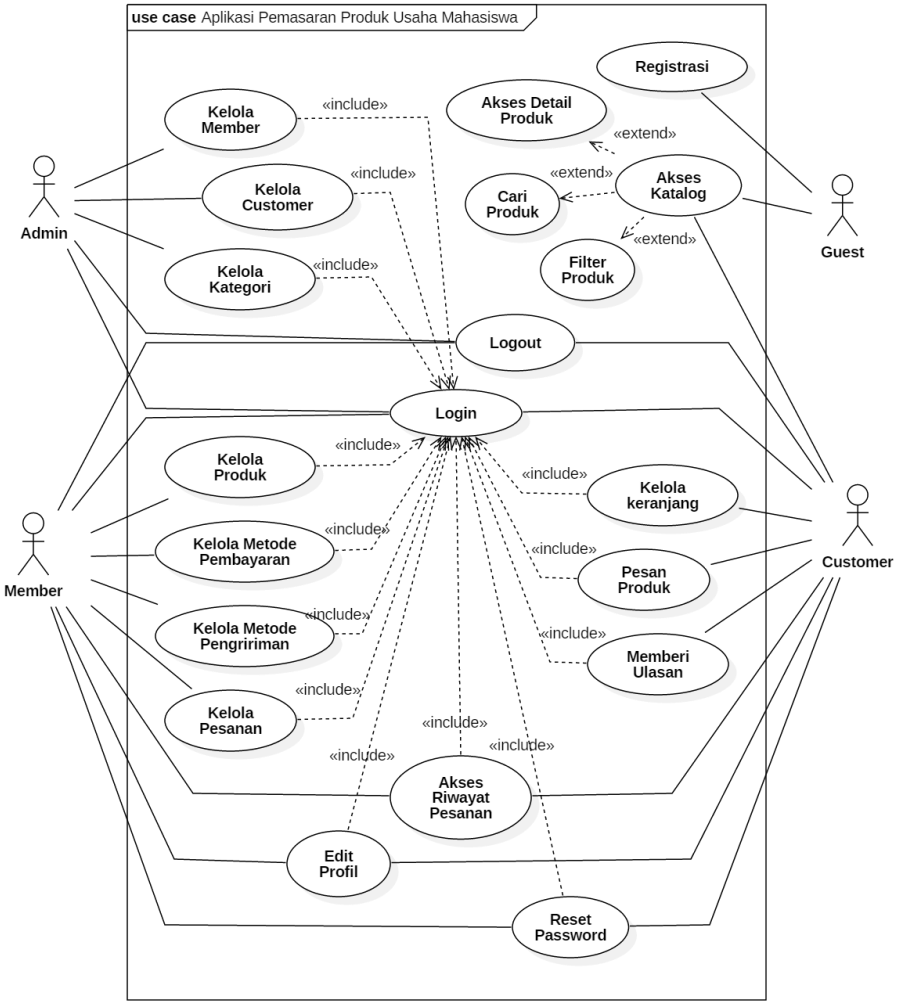
Tabel 5. Pemetaan Kebutuhan Non Fungsional

No.	Proses Berjalan (*As-Is*)	Sistem yang Akan Dibuat (*To-Be*)
1	***Security Requirements*:** Saat ini, tidak ada sistem otorisasi pengguna karena proses bisnis dilakukan secara terpisah satu sama lain dengan tingkat keamanan yang berbeda-beda tergantung individu.	***Security Requirements***: Aplikasi akan memiliki fitur otorisasi pengguna yang memastikan hanya pengguna yang berwenang dapat mengakses data dan fitur tertentu berdasarkan kredensial yang diberikan.
2	***Performance Requirements*:** Saat ini, tidak ada sistem otomatis untuk pengelolaan data, sehingga semua proses dilakukan secara manual, yang seringkali memakan waktu lebih lama dan tidak terukur.	***Performance Requirements***: Aplikasi akan dirancang untuk memiliki performa yang cepat, dengan waktu untuk mendapatkan, menambah, mengedit, dan menghapus data tidak lebih dari 10 detik.

3.4.1.4. *Functional Model*

Pada bagian ini dijelaskan mengenai *use case diagram*, *activity diagram* dan *use case description* dari proyek sistem informasi yang dikembangkan.

A. *Use case Diagram*



Gambar 8. *Use Case Diagram.*

Gambar 8 menunjukkan *use case diagram* dari aplikasi pemasaran produk usaha mahasiswa yang terdiri beberapa aktor sebagai pengguna aplikasi. Admin merupakan pihak pengelola yang bertugas memonitoring jalannya aplikasi. *Member* (Mahasiswa Wirausaha) merupakan mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung yang memasarkan produk pada aplikasi. *Customer* merupakan khalayak umum terutama civitas akademika dan masyarakat di lingkungan Universitas Lampung yang dapat membeli produk yang tersedia. *Guest* adalah pengunjung aplikasi yang tidak mempunyai akun. Setiap *role*

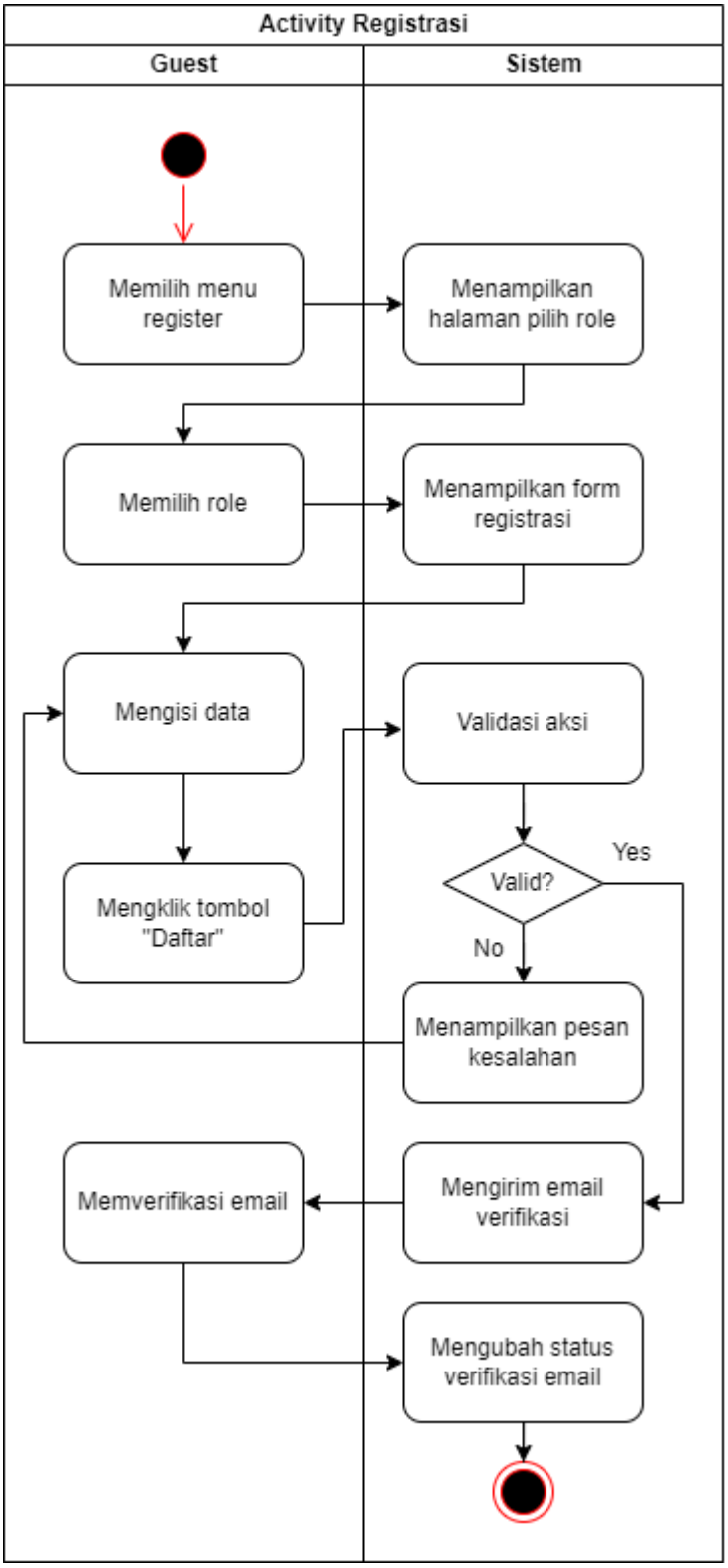
dapat melakukan aksi dan mengakses data tertentu dalam sistem. Adapun deskripsi dari masing-masing *use case* dijelaskan pada pembahasan *use case description*.

B. *Activity Diagram*

Setiap *use case* memiliki *activity diagram* yang menunjukkan alur proses sebuah aktivitas di dalam sistem. Adapun daftar *activity diagram* sistem ini adalah sebagai berikut.

1. *Activity Diagram* Registrasi

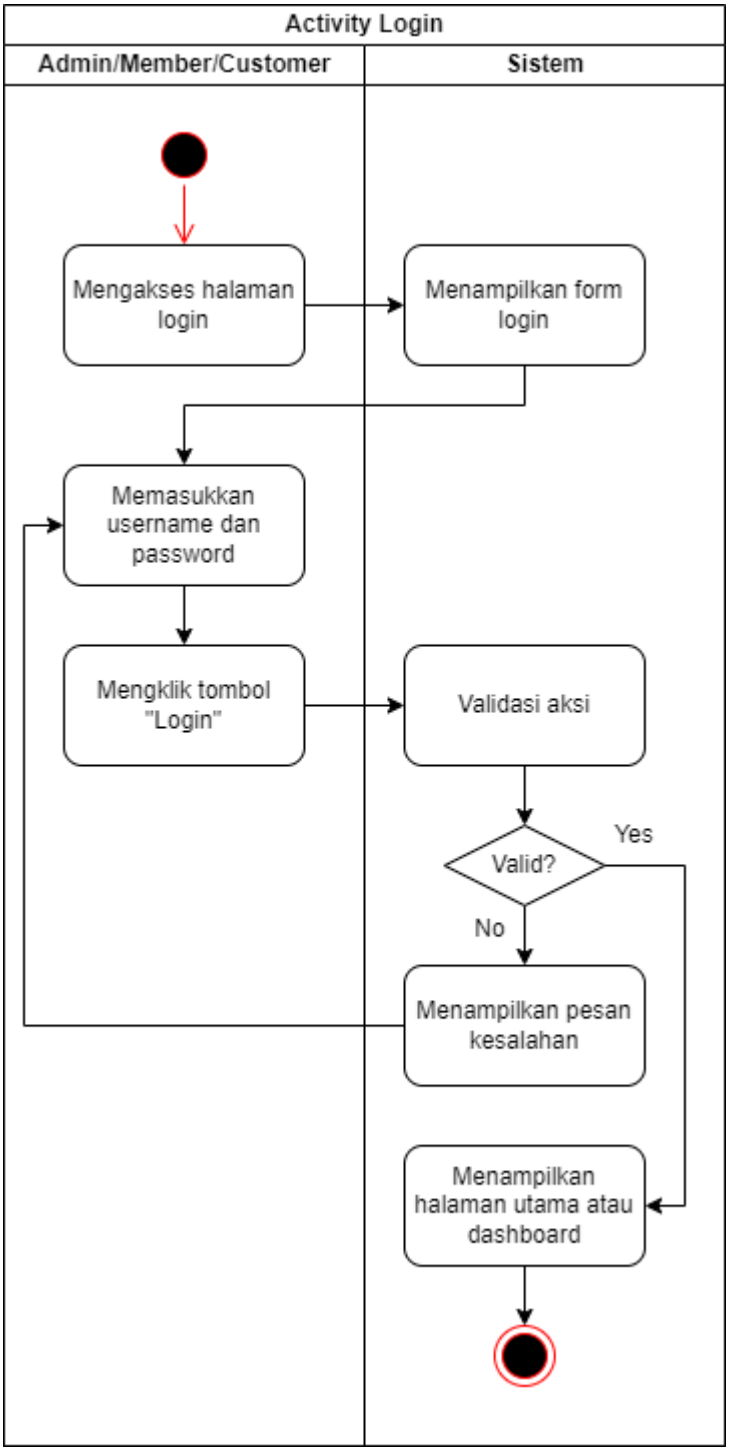
Activity diagram yang ditunjukkan pada Gambar 9 menjelaskan alur proses pengunjung yang ingin melakukan registrasi sebagai pengguna. Pengunjung dapat memilih menu register yang ada pada halaman utama atau halaman *login*. Kemudian sistem akan menampilkan halaman pilih *role* yang berisi opsi peran yaitu *Member* atau *Customer*. Pengunjung dapat memilih peran ini sesuai dengan yang dikehendaki. Sistem akan menampilkan form registrasi dan pengunjung harus mengisi form tersebut dengan lengkap. Setelah itu, pengunjung menyimpan data dan sistem akan melakukan validasi terhadap aksi yang dilakukan. Jika valid maka sistem akan mengirimkan *link* verifikasi ke *email* pengguna. Pengguna kemudian memverifikasi *email* melalui *link* tersebut dan proses registrasi selesai. Namun jika tidak valid maka sistem akan menampilkan pesan kesalahan.



Gambar 9. *Activity Diagram* Registrasi.

2. *Activity Diagram Login*

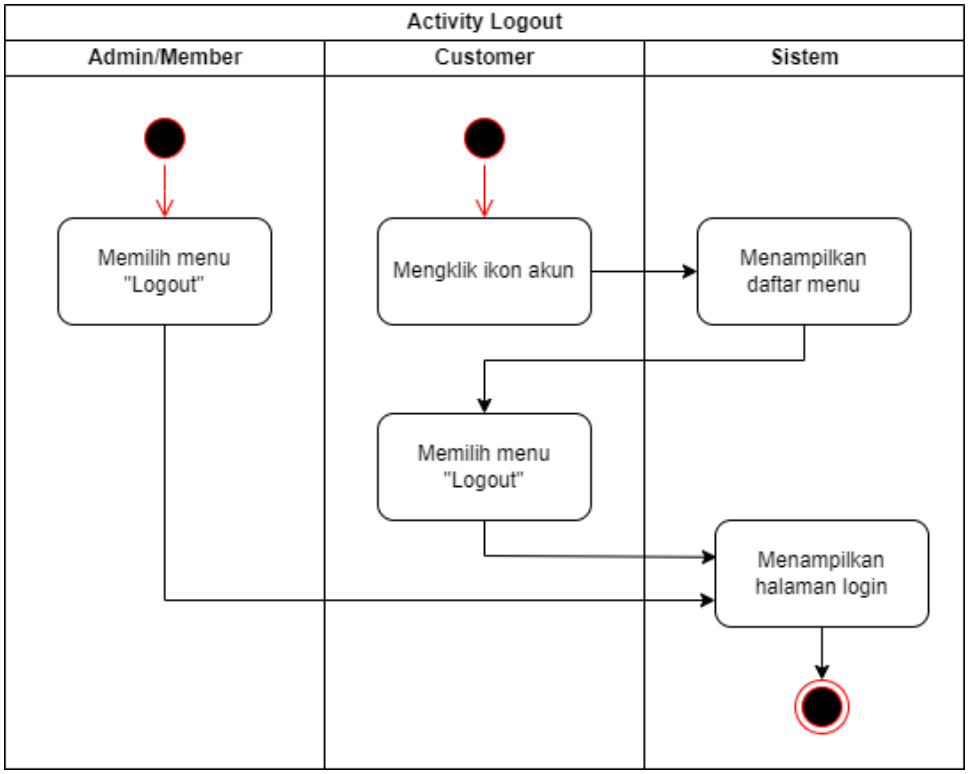
Activity diagram berikut menggambarkan proses *Admin, Member,* atau *Customer* yang *login* menggunakan akun yang dimiliki. Proses diawali dengan *Admin, Member,* atau *Customer* mengakses halaman *login* dan sistem akan menampilkan form *login*. *Admin, Member,* atau *Customer* memasukkan *username* dan *password* lalu mengklik tombol "Login". Sistem akan memvalidasi aksi yang dilakukan, jika valid maka sistem akan menampilkan halaman utama atau dashboard sesuai peran pengguna. Namun jika tidak valid maka pengguna akan tetap pada aktivitas mengisi *username* dan *password*.



Gambar 10. *Activity Diagram Login.*

3. ***Activity Diagram Logout***

Proses *logout* untuk semua peran digambarkan pada Gambar 11. Admin dan *Member* memiliki alur yang sama yaitu dengan memilik menu “Logout”. Sementara *Customer* harus mengklik ikon akun dan sistem akan menampilkan daftar menu, baru kemudian Customer memilih menu “Logout”. Setelah pengguna berhasil *logout*, maka sistem akan menampilkan halaman *login*.

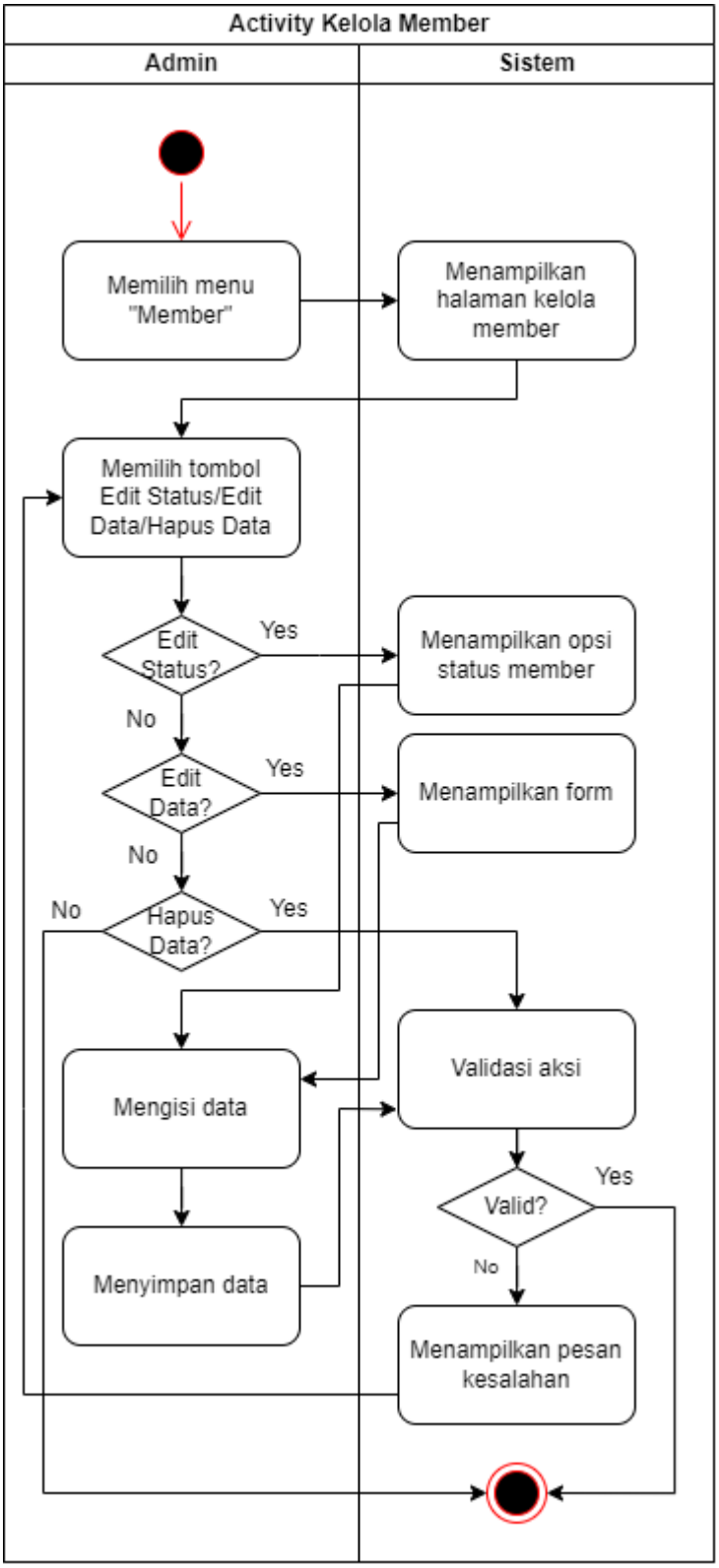


Gambar 11. *Activity Diagram Logout*.

4. ***Activity Diagram* Kelola *Member***

Activity diagram yang ditunjukkan pada Gambar 12 menjelaskan alur proses Admin dalam mengelola data *Member*. Alur proses tersebut dapat dilakukan ketika Admin sudah melakukan *login* dan berada di halaman *dashboard*. Untuk mengelola *Member*, Admin memilih menu “Member” dan sistem akan menampilkan halaman kelola member beserta data yang telah tersimpan. Jika Admin memilih edit status, maka sistem akan menampilkan

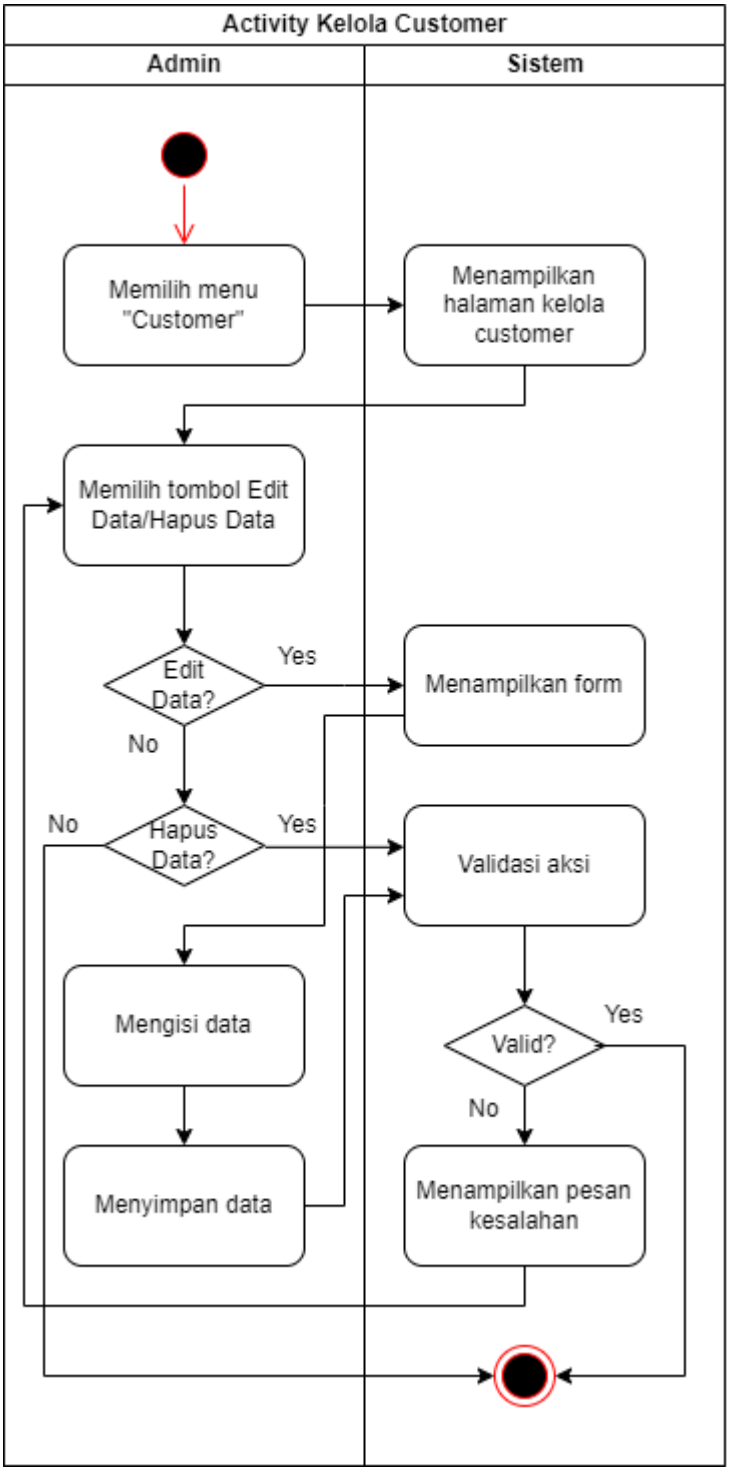
opsi untuk memverifikasi status *Member*. Setelah mengisi data, Admin menyimpan perubahan. Jika Admin memilih edit data, maka sistem menampilkan form data *Member*. Form tersebut kemudian diedit dengan data terbaru dan disimpan. Untuk menghapus akun, Admin dapat memilih tombol hapus. Setiap aksi yang dilakukan akan divalidasi oleh sistem. Jika valid maka proses selesai, jika tidak maka sistem akan menampilkan pesan kesalahan.



Gambar 12. *Activity Diagram* Kelola *Member.*

5. *Activity Diagram* Kelola *Customer*

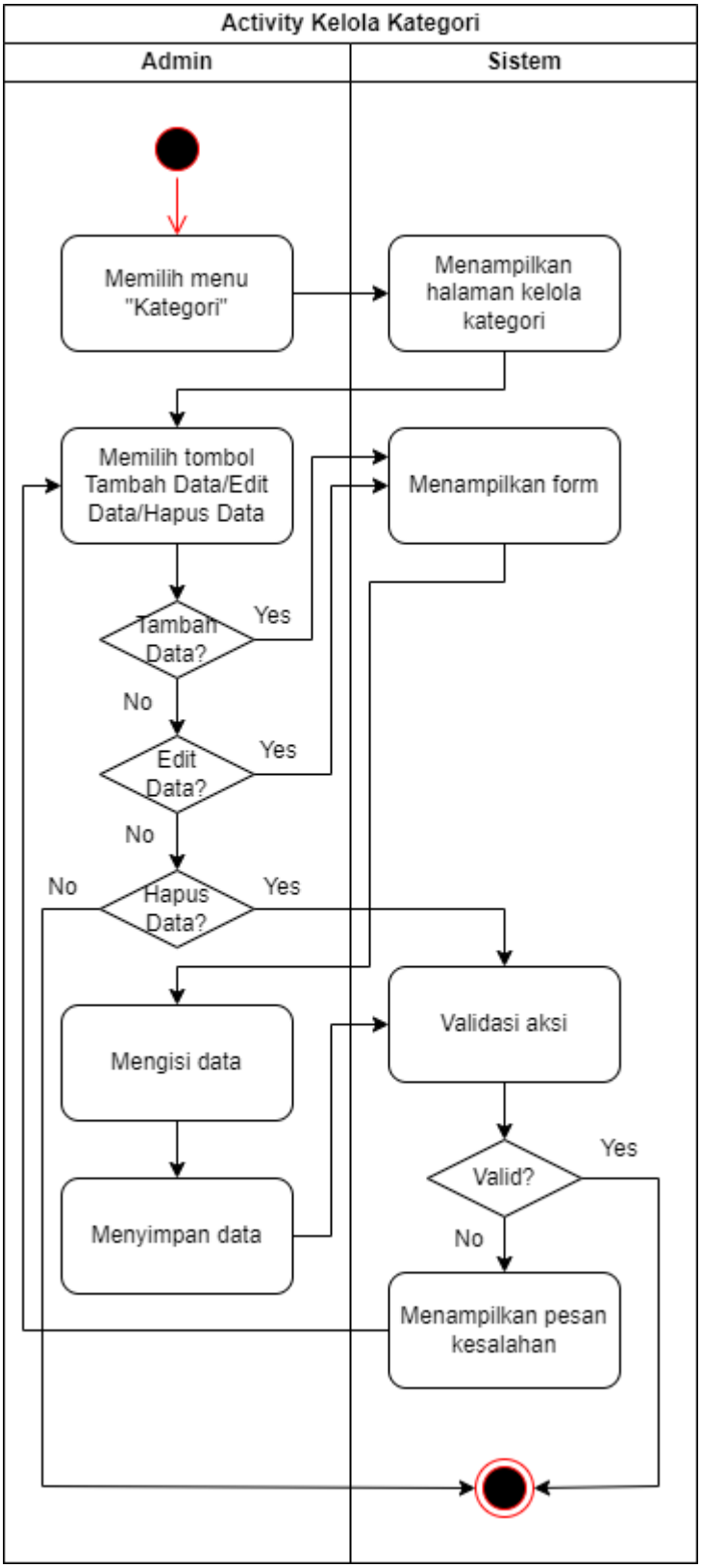
Activity diagram kelola *Customer* oleh Admin dapat dilihat pada Gambar 13. Alur proses tersebut dapat dilakukan ketika Admin sudah melakukan *login* dan berada di halaman *dashboard*. Untuk mengelola *Customer*, Admin memilih menu “Customer” dan sistem akan menampilkan halaman kelola *customer* beserta data yang telah tersimpan. Jika Admin memilih edit data, maka sistem menampilkan form data *customer*. Admin memperbarui data *customer* dan menyimpanya. Jika ingin menghapus akun, Admin dapat memilih tombol hapus. Setiap aksi yang dilakukan akan divalidasi oleh sistem. Jika valid maka proses selesai, jika tidak maka sistem akan menampilkan pesan kesalahan.



Gambar 13. *Activity Diagram* Kelola *Customer.*

6. *Activity Diagram* Kelola Kategori

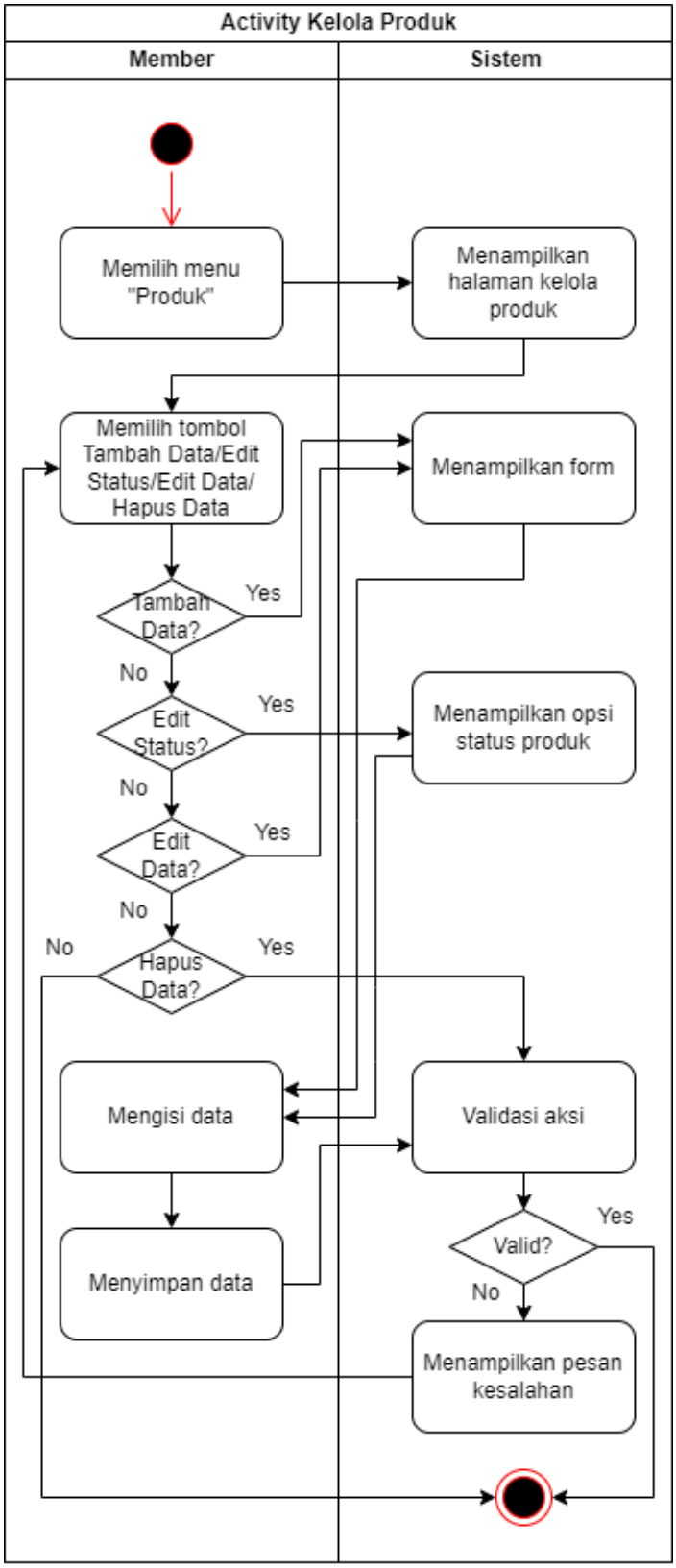
Gambar 14 memperlihatkan *activity diagram* Admin untuk mengelola kategori. Setelah melakukan *login* dan masuk ke *dashboard*, Admin memilih menu “Kategori”. Sistem akan menampilkan halaman kelola kategori beserta data kategori yang sudah tersimpan. Terdapat sejumlah aksi yang dapat dilakukan seperti tambah, edit, dan hapus data. Jika Admin menekan tombol tambah atau edit data maka sistem akan menampilkan form. Admin mengisi form tersebut dengan data yang sesuai, kemudian menyimpannya. Jika ingin menghapus data kategori, Admin dapat mengklik tombol hapus. Selanjutnya, sistem akan melakukan validasi terhadap aksi yang dilakukan. Jika valid maka proses selesai, namun jika tidak sistem akan menampilkan pesan kesalahan.



Gambar 14. *Activity Diagram* Kelola Kategori.

7. *Activity Diagram* Kelola Produk

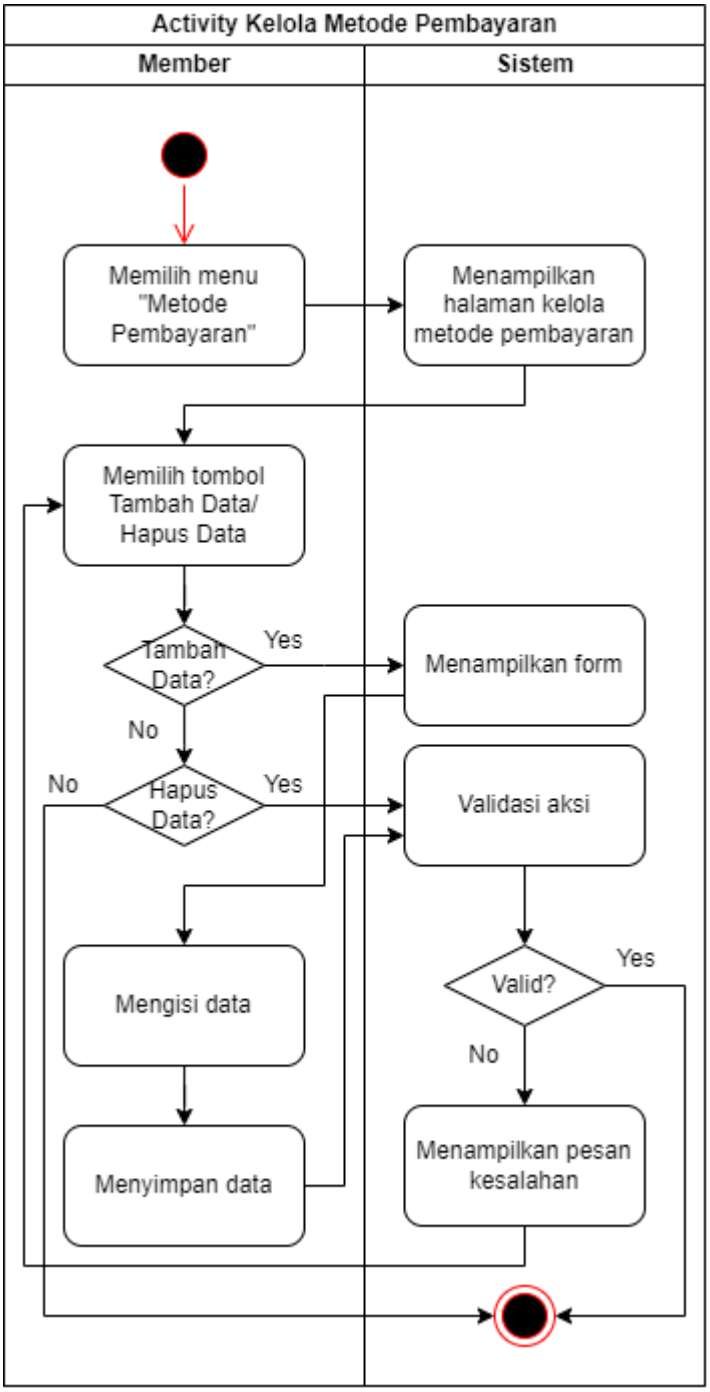
Gambar 15 merupakan *activity diagram* untuk alur proses kelola produk. Ketika sudah *login* dan berada di halaman *dashboard*, *Member* memilih menu “Produk”. Sistem akan menampilkan halaman kelola produk. Jika *Member* mengklik tombol tambah atau edit data, maka sistem akan menampilkan form. *Member* mengisi form tersebut dan menyimpannya. Jika *Member* memilih edit status, maka sistem akan menampilkan opsi untuk mengubah status produk. Setelah memilih, *Member* menyimpan perubahan. Untuk menghapus produk, *Member* dapat memilih tombol hapus. Setiap aksi yang dilakukan akan divalidasi oleh sistem. Jika valid maka proses selesai, jika tidak maka sistem akan menampilkan pesan kesalahan.



Gambar 15. *Activity Diagram* Kelola Produk.

8. *Activity Diagram* Kelola Metode Pembayaran

Alur proses pengelolaan metode pembayaran ditunjukkan pada Gambar 16. Setelah *login* dan berada di halaman *dashboard*, *Member* memilih menu "Metode Pembayaran". Sistem kemudian menampilkan halaman kelola metode pembayaran dengan berbagai tombol yang dapat dipilih oleh *Member* untuk melakukan aksi tertentu. Ketika *Member* mengklik tombol tambah, sistem akan menampilkan formulir untuk menambah metode pembayaran, yang kemudian diisi oleh *Member* dan disimpan. *Member* dapat menghapus metode pembayaran dengan mengklik tombol hapus. Setiap aksi yang dilakukan oleh *Member* akan divalidasi oleh sistem. Jika valid, proses selesai. Jika tidak, sistem akan menampilkan pesan kesalahan**.**

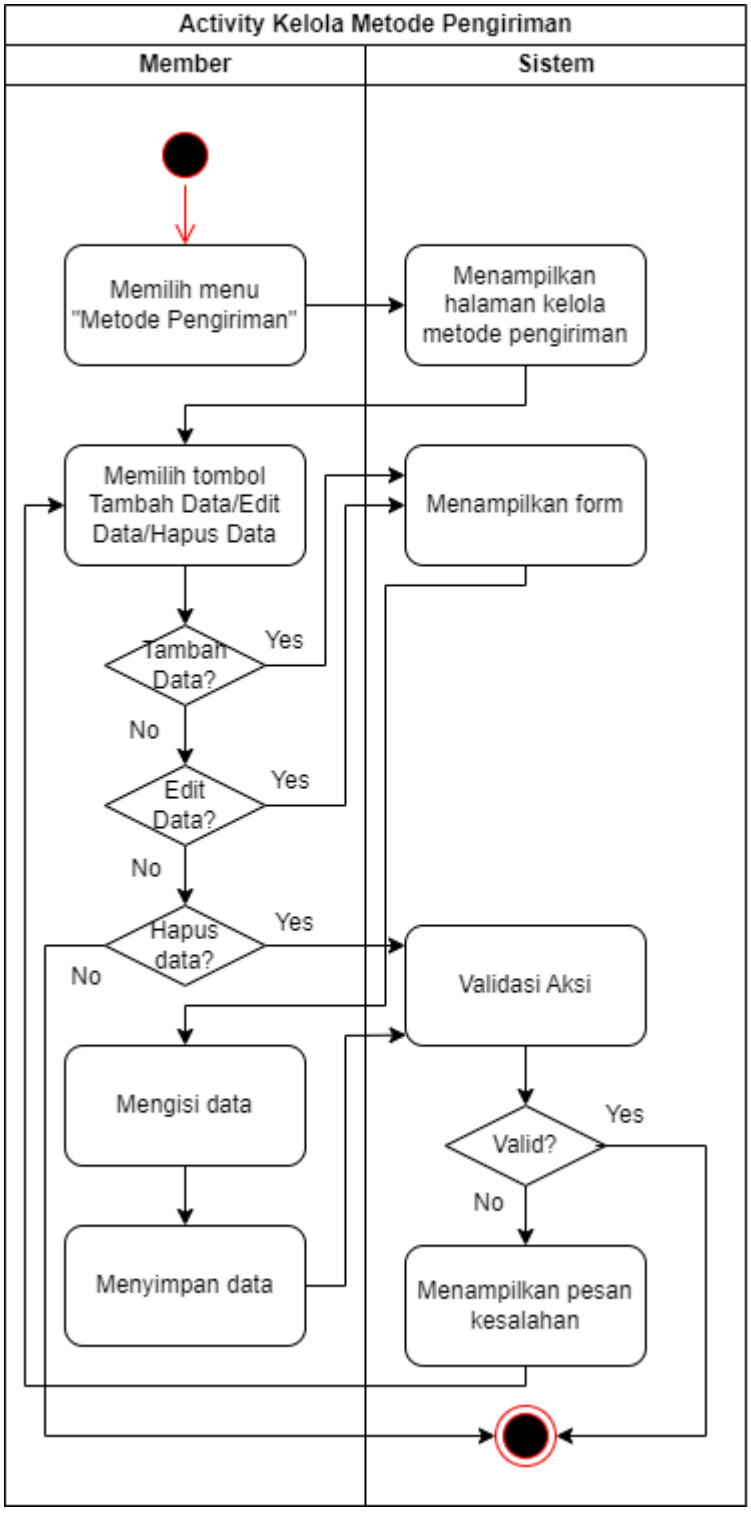


Gambar 16. *Activity Diagram* Kelola Metode Pembayaran.

9. *Activity Diagram* Kelola Metode Pengiriman

Activity diagram mengelola metode pengiriman ditunjukkan pada Gambar 17. Setelah *login* dan berada di halaman *dashboard*, *Member* memilih menu "Metode Pengiriman". Selanjutnya sistem akan menampilkan halaman

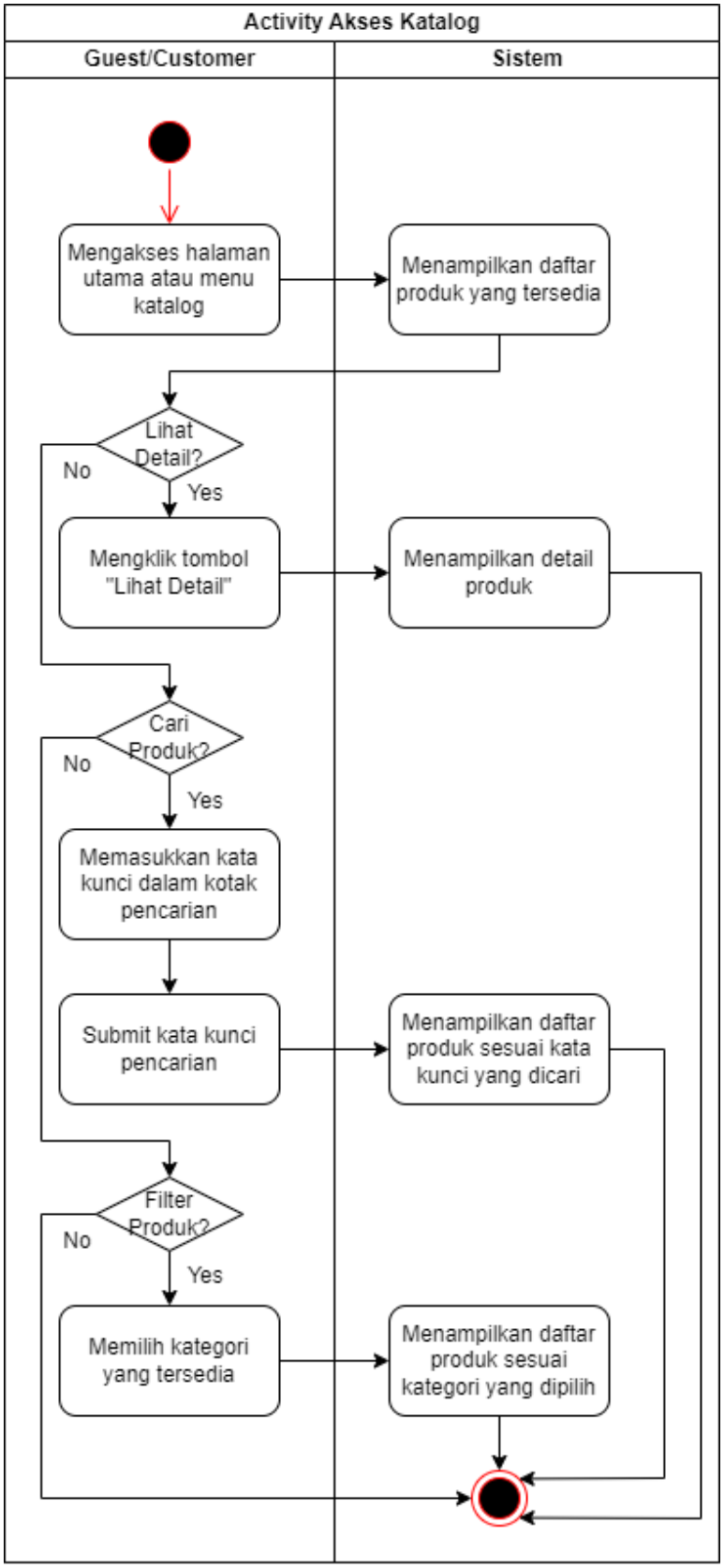
kelola metode pengiriman. Terdapat berbagai tombol yang dapat dipilih oleh *Member* untuk melakukan aksi tambah, edit dan hapus. Ketika *Member* mengklik tombol tambah atau edit data, sistem akan menampilkan form. Form diisi oleh *Member* dan disimpan. Untuk menghapus metode pembayaran, dapat dilakukan dengan mengklik tombol hapus. Setiap aksi yang dilakukan oleh *Member* akan divalidasi oleh sistem. Jika valid, proses selesai. Jika tidak, sistem akan menampilkan pesan kesalahan.



Gambar 17. *Activity Diagram* Kelola Metode Pengiriman.

10. *Activity Diagram* Akses Katalog

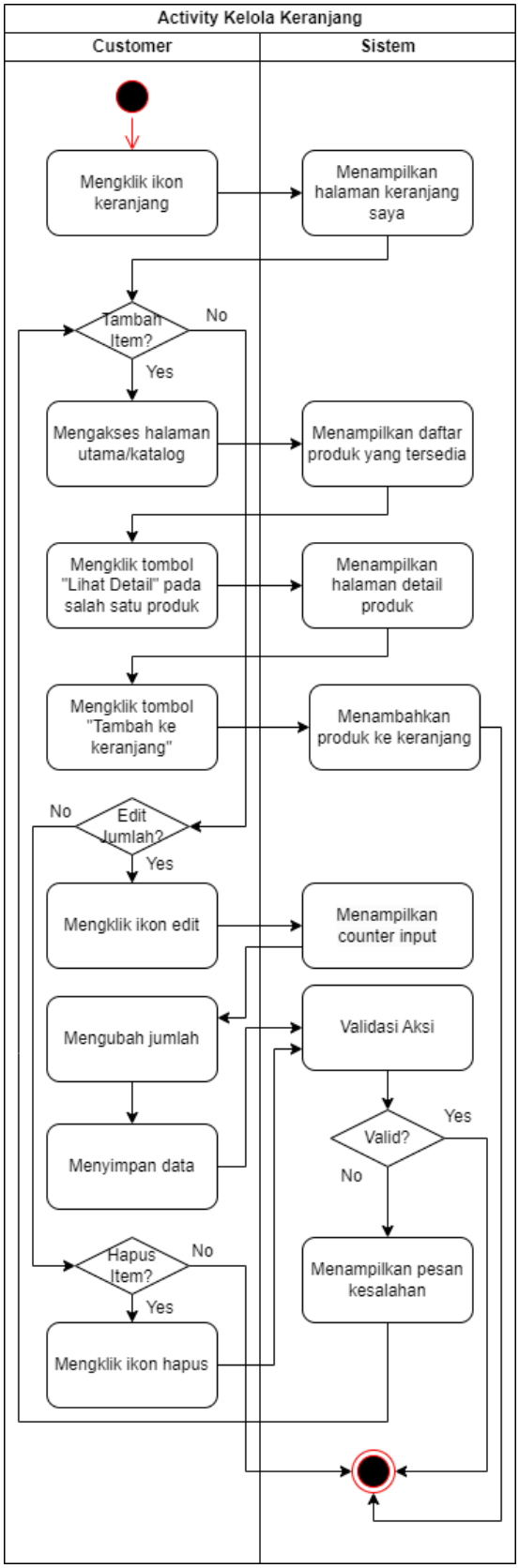
Gambar 18 menunjukkan alur untuk mengakses halaman katalog. *Guest* atau *Customer* dapat mendapatkan informasi mengenai semua produk yang tersedia dengan memilih menu katalog atau dengan mengunjungi halaman utama aplikasi, yaitu halaman yang pertama kali muncul saat url diakses. Sistem kemudian akan menampilkan daftar produk yang dipasarkan dalam aplikasi. *Guest* atau *Customer* dapat melakukan tindakan lanjutan seperti melihat detail, mencari produk, atau menyortir produk. Untuk melihat detail, pengguna mengklik tombol “Lihat Detail” pada salah satu produk dan sistem akan menampilkan detail produk yang dipilih. Jika ingin mencari produk tertentu, pengguna memasukkan kata kunci pada kotak pencarian dan mengirimkannya. Sistem akan melakukan pencarian dan selanjutnya akan menampilkan daftar produk yang namanya mengandung kata kunci yang diberikan. Selain itu, untuk menyortir produk berdasarkan kategori pengguna memilih kategori yang tersedia pada *dropdown*. Sistem akan melakukan penyortiran dan selanjutnya akan menampilkan daftar produk dengan kategori yang sesuai.



Gambar 18. *Activity Diagram* Akses Katalog.

11. *Activity Diagram* Kelola Kelola Keranjang

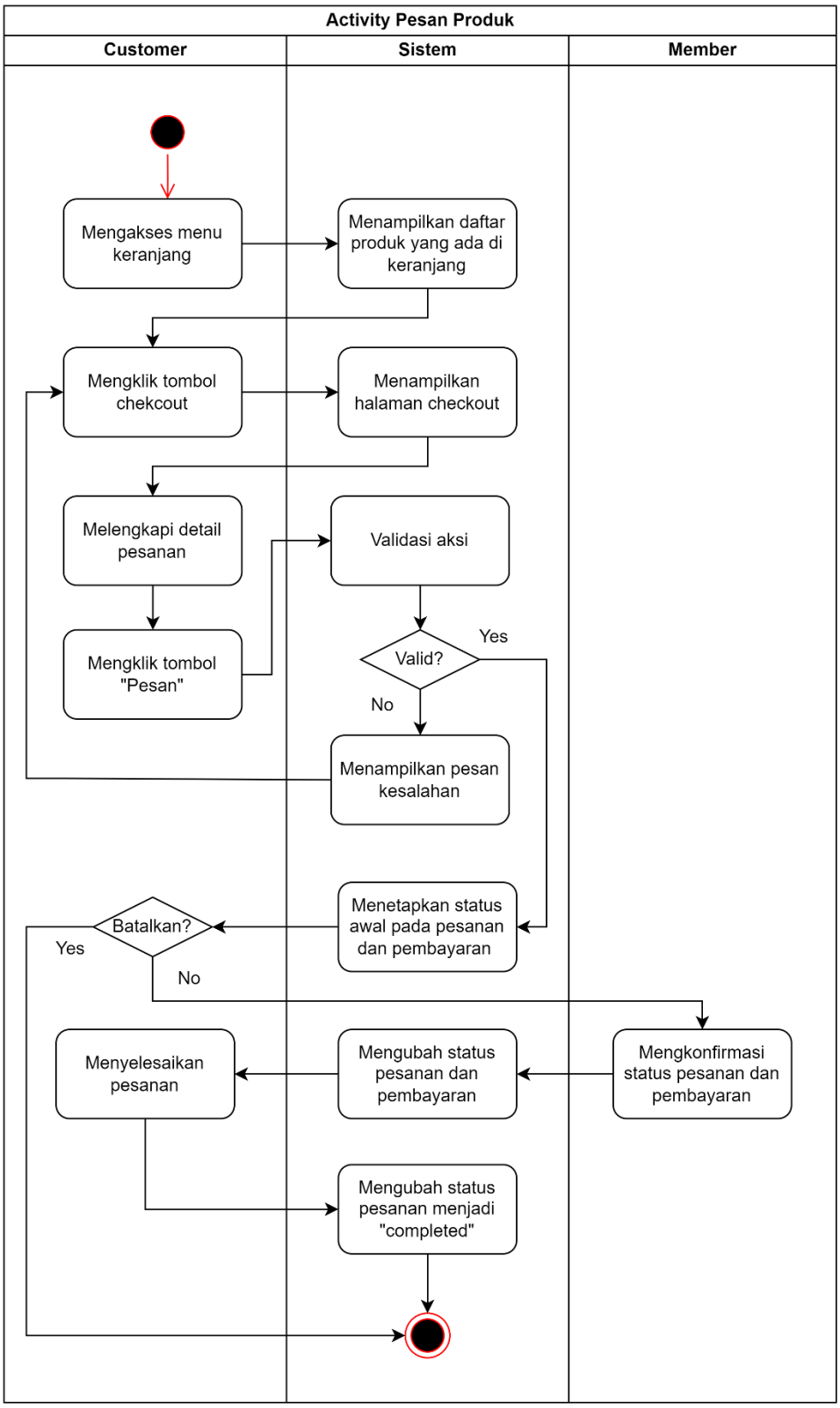
Gambar 19 merupakan *activity diagram* yang menggambarkan alur proses *Customer* untuk mengelola keranjang. Setelah *login* dan berada pada halaman utama aplikasi, *Customer* mengklik ikon keranjang dan sisem akan menampilkan halaman keranjang saya. Untuk menambahkan produk ke keranjang, *Customer* memilih salah satu produk pada halaman katalog lalu melihat detail produk. Pada halaman detail produk, *Customer* mengklik tombol “Tambah ke Keranjang”. Jika *Customer* ingin menambah jumlah item, maka hal ini dapat dilakukan pada halaman keranjang dengan mengklik tombol edit maka sistem akan memunculkan *counter input* untuk mengubah jumlah dan *Customer* menyimpan perubahan. Selain itu, jika ingin menghapus item, maka *Customer* mengklik tombol hapus. Aksi tersebut akan divalidasi sistem dan jika valid proses selesai sebaliknya jika tidak maka sistem memunculkan pesan kesalahan.



Gambar 19. *Activity Diagram* Kelola Kelola Keranjang.

12. *Activity Diagram* Pesan Produk

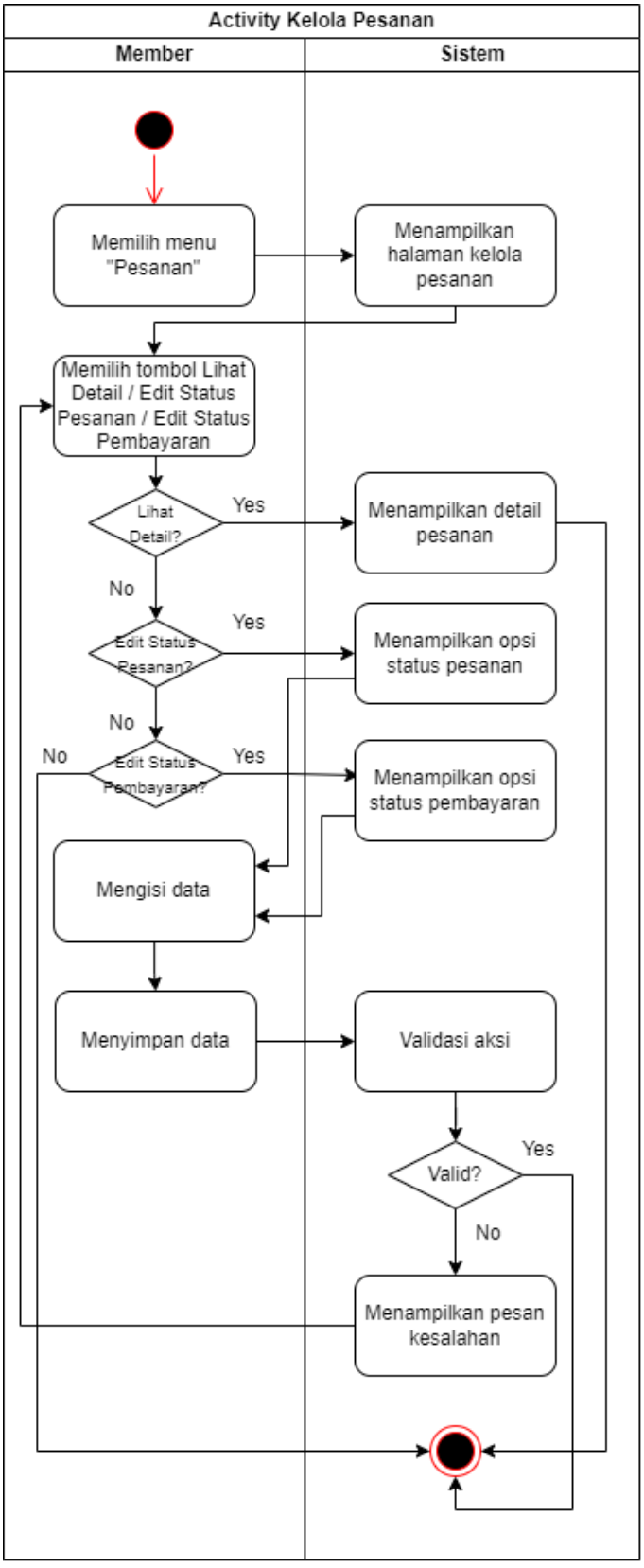
Alur proses pesan produk oleh *Customer* dimodelkan pada Gambar 20. Saat *Customer* sudah *login* dan berada di menu keranjang. Sistem akan menampilkan daftar produk di keranjang. Sebelum melakukan pemesanan, *Customer* mengklik salah satu tombol *checkout* pada keranjang. Setelah itu, sistem akan menampilkan detail pemesanan yang harus *Customer* lengkapi. Setelah data pemesanan sudah diisi, selanjutnya *Customer* mengklik tombol "Pesan". Proses kemudian ini akan divalidasi oleh sistem. Jika valid, maka sistem akan menetapkan status awal pada pesanan pada status. Jika tidak valid, maka akan muncul pesan kesalahan. Saat pesanan masih berstatus pending, *Customer* masih dapat membatalkan pesanan, namun jika ingin tetap melanjutkan maka *Member* akan mengkonfirmasi pesanan yang masuk. Sistem selanjutnya akan mengubah status pesanan dan pembayaran sesuai yang dipilih *Member*. Setelah itu, *Customer* dapat menyelesaikan pesanan dan sistem akan mengubah statusnya menjadi selesai.



Gambar 20. *Activity Diagram* Pesan Produk

13. *Activity Diagram* Kelola Pesanan

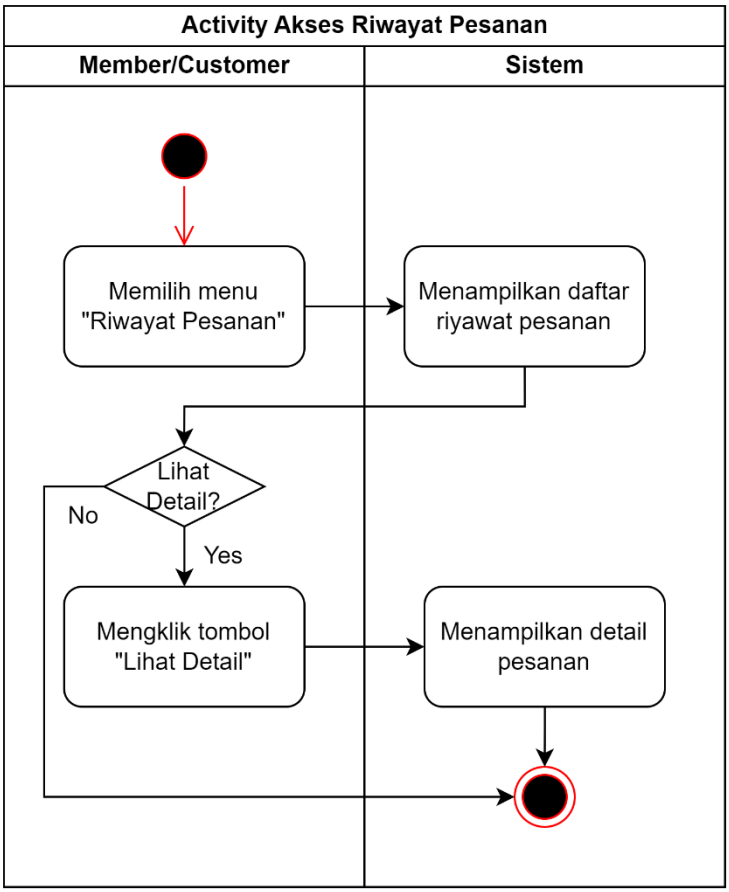
Alur proses untuk mengelola pesanan yang dapat dilakukan oleh *Member* setelah login ke sistem dapat dilihat pada Gambar 21. *Member* memilih menu “Pesanan” dan selanjutnya sistem akan menampilkan halaman kelola pesanan. Jika *Member* ingin melihat detail pesanan, maka dapat mengklik tombol lihat detail dan sistem akan menampilkan detail pesanan yang dipilih. Saat tombol edit status pesanan diklik, sistem akan menampilkan opsi untuk mengubah status pesanan. Begitu juga dengan tombol edit status pembayaran, ketika diklik maka akan muncul opsi untuk mengubah status pembayaran. Setiap aksi yang dilakukan oleh *Member* akan divalidasi oleh sistem. Jika valid, proses selesai. Jika tidak, sistem akan menampilkan pesan kesalahan.



Gambar 21. *Activity Diagram* Kelola Pesanan.

14. *Activity Diagram* Akses Riwayat Pesanan

Gambar 22 adalah alur *Member* atau *Customer* untuk mengakses riwayat pesanan. *Member* atau *Customer* yang sudah *login* memilih menu "Riwayat Pesanan" pada tampilan aplikasi sesuai *role*. Selanjutnya sistem akan menampilkan daftar pesanan yang pernah diterima *Member* dan dilakukan *Customer*. Jika ingin melihat detail, maka *Member* atau *Customer* dapat mengklik tombol "Lihat Detail" dan sistem akan menampilkan detail pesanan.

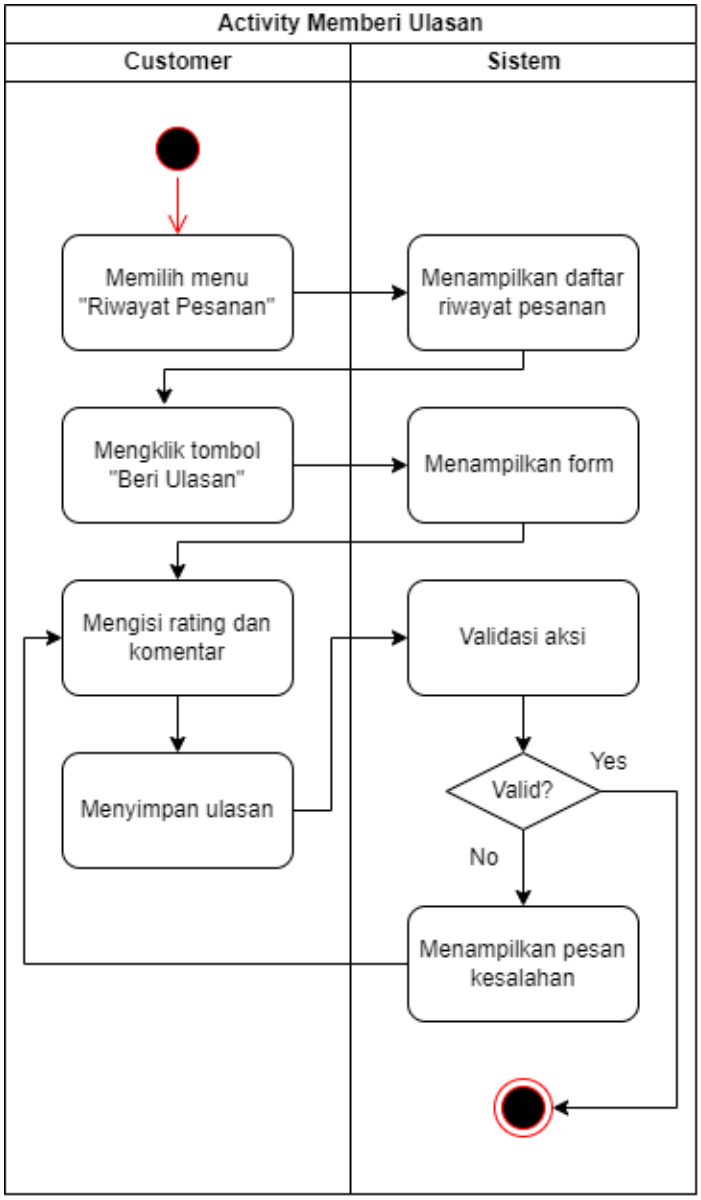


Gambar 22. *Activity Diagram* Akses Riwayat Pesanan.

15. *Activity Diagram* Memberi Ulasan

Activity diagram yang ditunjukkan oleh Gambar 23 memberikan alur proses untuk memberi ulasan pada pesanan yang telah dilakukan. Saat *Customer*

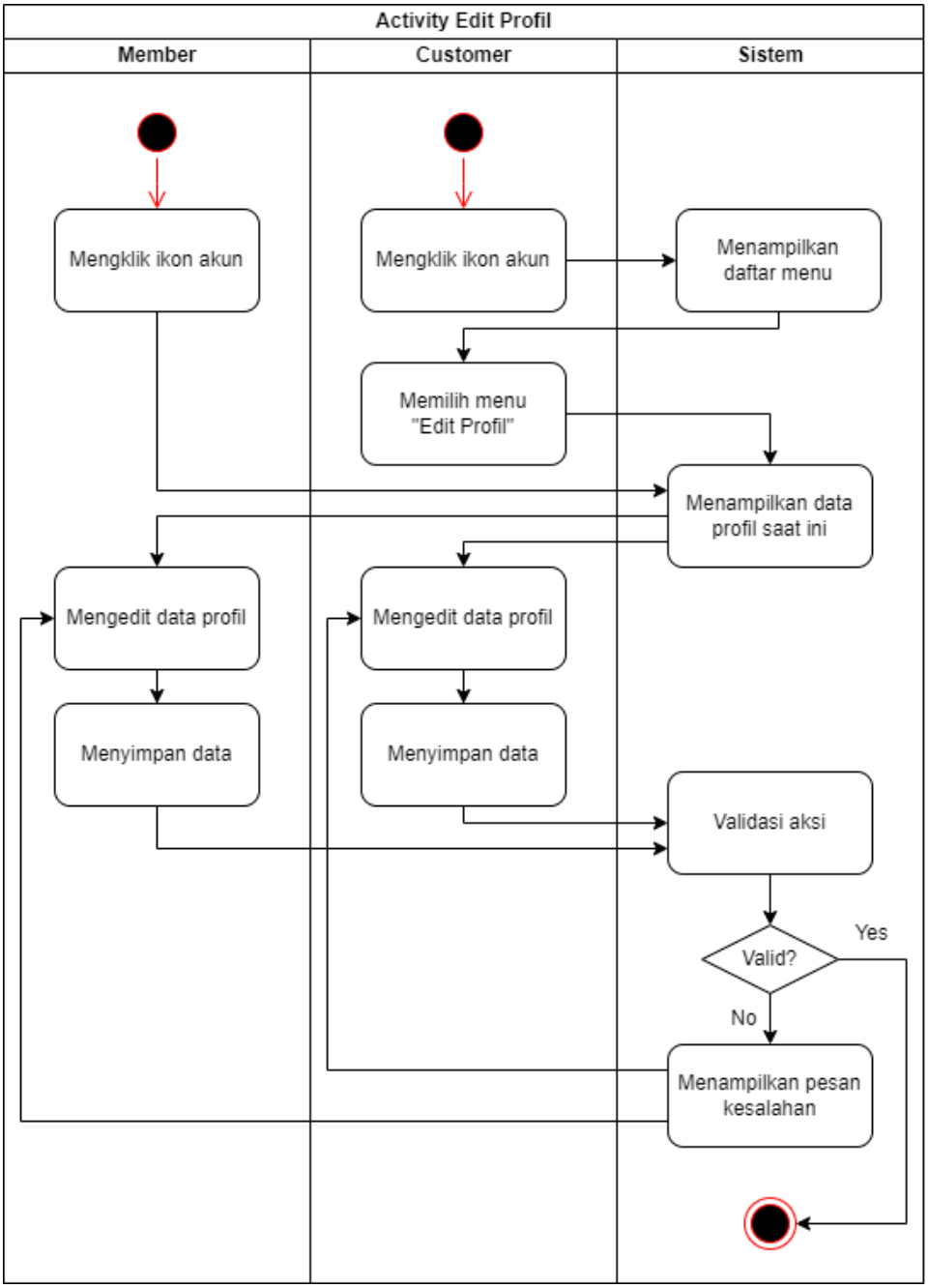
sudah *login* dan berhasil menyelesaikan pesanan, sistem akan menampilkan data pesanan selesai dan tombol beri ulasan pada halaman riwayat pesanan. Customer mengklik tombol ini kemudian sistem menampilkan form ulasan. *Customer* mengisi *rating* dan komentar kemudian menyimpannya. Jika sistem berhasil memvalidasi maka proses selesai dan ulasan akan muncul pada menu detail produk, sedangkan jika data gagal divalidasi maka akan muncul pesan kesalahan.



Gambar 23. *Activity Diagram* Memberi Ulasan

16. *Activity Diagram* Edit Profil

Activity diagram tersebut menggambarkan alur proses edit profil pengguna yang dapat dilakukan oleh *Member* dan *Customer* yang sudah *login* ke dalam sistem. *Member* mengklik ikon akun dan sistem menampilkan data profil saat ini. Selain itu, untuk *Customer* saat mengklik ikon akun maka sistem akan menampilkan daftar menu. Customer akan memilih "Edit Profil" dan sistem menampilkan data profil saat ini. *Member* atau *Customer* mengedit data dan menyimpan perubahan. Jika sistem berhasil memvalidasi data maka proses selesai dan jika gagal maka akan muncul pesan kesalahan.

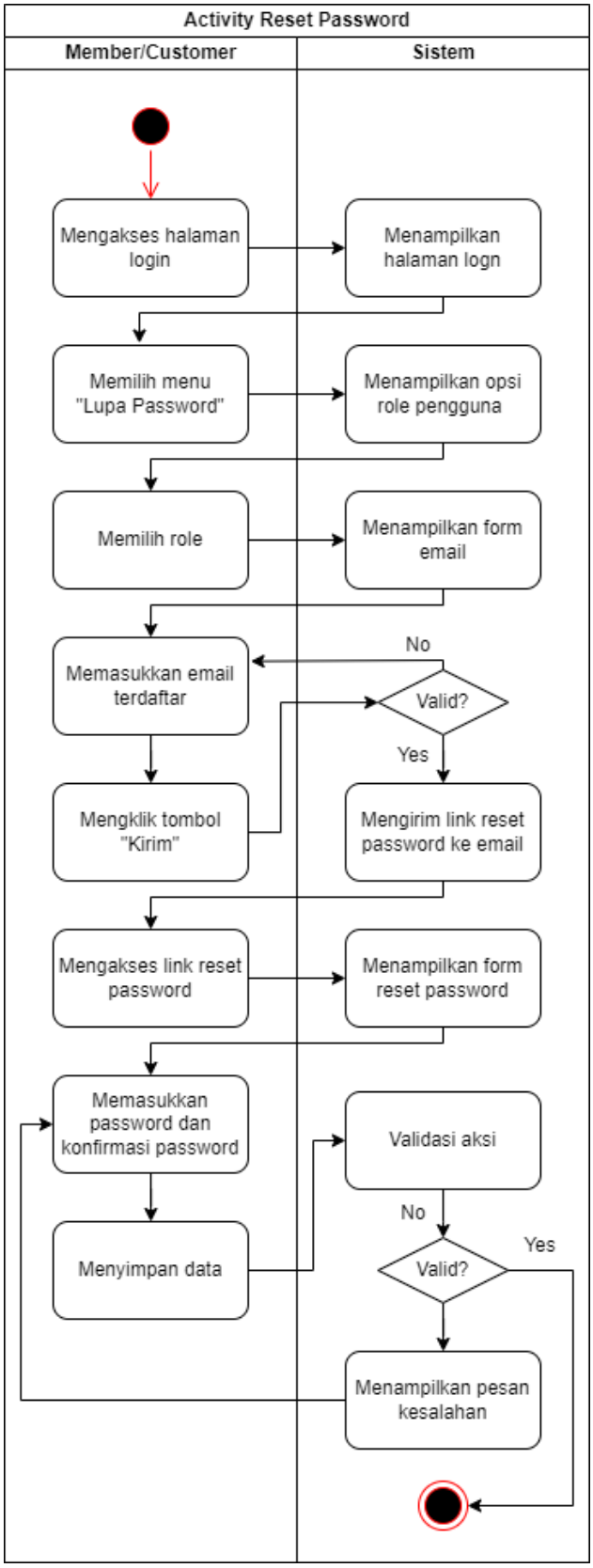


Gambar 24. *Activity Diagram* Edit Profil.

17. *Activity Diagram* Reset *Password*

Activity diagram ini menggambarkan proses reset *password* saat pengguna lupa terhadap *password* akun. Pengguna memulai dengan mengakses halaman login, kemudian memilih opsi "Lupa Password". Setelah itu, pengguna memilih peran (*role*) mereka dan memasukkan alamat *email* yang

terdaftar. Sistem akan memvalidasi *email* tersebut. Jika valid, sistem akan mengirimkan link reset *password* ke *email* pengguna. Pengguna kemudian mengakses link tersebut dan memasukkan *password* baru beserta konfirmasinya. Sistem akan memvalidasi *password* baru, dan jika valid, proses reset *password* selesai. Namun, jika tidak akan muncul pesan kesalahan.



Gambar 25. *Activity Diagram* Reset *Password.*

C. ***Use Case Description***

Deskripsi masing-masing *use case* yang telah dimodelkan pada *use case diagram*, dijelaskan pada tabel-tabel berikut.

Tabel 6. *Use Case Description* Registrasi

Use Case Name**:** Registrasi	**ID**: UC-01	***Priority*****:** Tinggi
Actor**:** *Guest*		
Description**:** Proses pendaftaran akun yang dapat digunakan untuk *login* ke dalam sistem. Terdapat dua jenis registrasi yaitu sebagai *Member* dan *Customer*. Untuk registrasi *Member* dibutuhkan bukti identitas berupa foto KTM untuk memastikan calon *Member* adalah benar mahasiswa FMIPA Unila.		
Trigger**:** *Guest* ingin mendaftar sebagai pengguna baru.		
Preconditions**:** *Guest* berada di halaman *login* atau halaman utama aplikasi dan memilih opsi registrasi.		
Normal Course**:** 1. *Guest* memilih menu register pada halaman utama atau halaman *login*. 2. Sistem menampilkan opsi *role* berupa *Member* dan *Customer*. 3. *Guest* memilih *role*. 4. Sistem menampilkan form registrasi sesuai *role* yang dipilih. 5. *Guest* mengisi form registrasi dengan informasi yang diperlukan (nama lengkap, email, username password, alamat, dan informasi tambahan untuk *Member* seperti deskripsi usaha dan identitas sebagai mahasiswa FMIPA Unila). 6. *Guest* mengirimkan form registrasi dengan menekan tombol "Daftar". 7. Sistem memvalidasi data yang diinputkan. 8. Jika valid, sistem akan mengirimkan *link* verifikasi ke email.		

9. *Guest* memverfikasi *email* melalui *link* tersebut dan status verifikasi akan diubah oleh sistem.
***Postconditions*:** Akun baru berhasil dibuat dan diverifikasi, pengguna dapat melakukan *login* ke sistem.
***Sub Flows*: -**
***Alternate / Exceptional Flows*:** 1. Jika *email* sudah terdaftar, sistem menampilkan pesan kesalahan dan meminta *Guest* untuk menggunakan email lain. 2. Jika *username* sudah terdaftar, sistem menampilkan pesan kesalahan dan meminta *Guest* untuk menggunakan *username* lain. 3. Jika NPM sudah ada, sistem menampilkan pesan kesalahan. 4. Jika validasi data gagal, sistem menampilkan pesan kesalahan dan meminta perbaikan.
Jumlah Transaksi: Bergantung pada jumlah calon pendaftar yang mendaftar akun baru melalui sistem

Tabel 7. *Use Case Description Login*

***Use Case Name*:** *Login*	**ID**: UC-02	***Priority*:** Tinggi
***Actor*:** *Admin, Member, Customer*		
***Description*:** Proses *login* ke dalam sistem agar dapat mengakses data atau melakukan tindakan tertentu.		
***Trigger*:** *Admin, Member,* atau *Customer* ingin *login* ke dalam sistem.		
***Preconditions*:** *Admin, Member,* atau *Customer* berada di halaman *login.*		
***Normal Course*:**		

1. *Admin, Member,* atau *Customer* mengakses halaman *login*. 2. Sistem menampilkan form *login*. 3. *Admin, Member,* atau *Customer* memasukkan *username* dan *password* dan mengklik tombol "Login". 4. Sistem memvalidasi data yang diinputkan dan mengecek kesesuaian kredensial yang diberikan. 5. Jika valid, sistem akan menampilkan halaman utama atau *dashboard*.
***Postconditions*:** Pengguna berhasil masuk ke sistem.
***Sub Flows*: -**
***Alternate / Exceptional Flows*:** 1. Jika *username* salah, maka sistem akan memberikan pesan *username* atau *password* salah. 2. Jika *password* salah, maka sistem akan memberikan pesan *username* atau *password* salah.
Jumlah Transaksi: Bergantung pada jumlah pengguna yang akan login ke dalam sistem.

Tabel 8. *Use Case Description Logout*

***Use Case Name*:** *Logout*	**ID**: UC-03	***Priority*:** Tinggi
***Actor*:** *Member, Customer, Admin*		
***Description*:** Proses *logout* untuk keluar dari sistem.		
***Trigger*:** *Member, Customer, Admin* ingin keluar dari sistem.		
***Preconditions*:** *Member, Customer, Admin* masih berada dalam sistem.		
***Normal Course*:**		

1. Pengguna memilih menu “Logout” 2. Sistem akan menghapus sesi pengguna dan menampilkan halaman *login.*
Postconditions**:** Pengguna keluar dari sistem.
Sub Flows**: -**
Alternate / Exceptional Flows**:** -
Jumlah Transaksi: Bergantung pada jumlah pengguna yang ingin keluar dari sistem.

Tabel 9. *Use Case Description* Kelola Member

Use Case Name**:** Kelola *Member*	**ID**: UC-04	***Priority*****:** Tinggi
Actor**:** Admin		
Description**:** *Use case* ini menggambarkan proses Admin dalam mengelola data *Member*, termasuk verifikasi registrasi, pengeditan informasi, dan penghapusan akun *member*.		
Trigger**:** Admin ingin mengelola data *Member*.		
Preconditions**:** Admin harus sudah *login* dan berada di dashboard Admin.		
Normal Course**:** 1. Admin memilih menu "Member" di *dashboard*. 2. Sistem menampilkan halaman kelola member. 3. Admin memilih *Member* yang ingin dikelola. 4. Admin memilih aksi yang diinginkan: Edit Status, Edit Data, atau Hapus Data. 5. Jika memilih Edit Status, sistem akan menampilkan opsi untuk mengubah status *Member* menjadi disetujui atau ditolak.		

6. Jika memilih Edit Data, maka Admin mengisi form dengan informasi *Member* terbaru dan menyimpan perubahan. 7. Jika memilih Hapus Data, sistem akan menghapus data *Member*.
Postconditions**:** Data *Member* berhasil dikelola sesuai aksi yang dilakukan oleh Admin.
Sub Flows**: -**
Alternate / Exceptional Flows**:** Jika data yang diinputkan tidak valid, sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 10. *Use Case Description* Kelola Customer

Use Case Name**:** Kelola *Customer*	**ID**: UC-05	***Priority*****:** Tinggi
Actor**:** Admin		
Description**:** *Use case* ini menggambarkan proses Admin dalam mengelola data *Customer*, termasuk pengeditan informasi, dan penghapusan akun *Member*.		
Trigger**:** Admin ingin mengelola data *Customer*.		
Preconditions**:** Admin harus sudah *login* dan berada di dashboard Admin.		
Normal Course**:** 1. Admin memilih menu "Customer" di *dashboard*. 2. Sistem menampilkan halaman kelola customer. 3. Admin memilih *Customer* yang ingin dikelola. 4. Admin memilih aksi yang diinginkan: Edit Data atau Hapus Data. 5. Jika memilih Edit Data, maka Admin mengisi form dengan informasi *Customer* terbaru dan menyimpan perubahan.		

6. Jika memilih Hapus, sistem akan menghapus data *Customer*.
Postconditions**:** Data *Customer* berhasil dikelola sesuai aksi yang dilakukan oleh Admin.
Sub Flows**: -**
Alternate / Exceptional Flows**:** Jika data yang diinputkan tidak valid, sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 11. *Use Case Description* Kelola Kategori

Use Case Name**:** Kelola Kategori	**ID**: UC-06	***Priority*****:** Tinggi
Actor**:** Admin		
Description**:** *Use case* ini menggambarkan proses Admin untuk mengelola kategori produk di sistem, termasuk menambahkan, mengedit, dan menghapus kategori.		
Trigger**:** Admin ingin mengelola kategori produk.		
Preconditions**:** Admin harus sudah *login* ke sistem dan berada di halaman dashboard admin.		
Normal Course**:** 1. Admin memilih menu "Kategori" di *dashboard*. 2. Sistem menampilkan halaman kelola kategori. 3. Admin memilih aksi (Tambah Data, Edit Data, atau Hapus Data). 4. Jika Admin memilih Tambah Data, maka akan mengisi form yang berisi nama dan deskripsi. Jika Admin memilih Edit Data, maka akan mengisi		

form perubahan data kategori terbaru. Jika Admin memilih Hapus Data, maka sistem akan melakukan validasi. 5. Admin menyimpan form yang telah diisi. 6. Sistem melakukan validasi data yang dimasukkan pada form 7. Jika valid, sistem menyimpan dan menghapus data kategori dari *database*.
Postconditions**:** Data kategori akan ditampilkan atau dihapus pada form tambah produk, menu *filter*, dan tabel kelola kategori.
Sub Flows**: -**
Alternate / Exceptional Flows**:** Jika data yang diinputkan tidak valid, sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 12. *Use Case Description* Kelola Produk

Use Case Name**:** Kelola Produk	**ID**: UC-07	***Priority*****:** Tinggi
Actor**:** *Member*		
Description**:** *Use case* ini menggambarkan proses *Member* untuk mengelola produk yang dijual, termasuk menambahkan, mengedit, dan menghapus produk.		
Trigger**:** *Member* ingin mengelola produk yang dijual		
Preconditions**:** *Member* harus sudah *login* ke sistem dan berada di halaman dashboard *Member*.		
Normal Course**:** 1. *Member* memilih menu "Produk" di *dashboard*. 2. Sistem menampilkan daftar halaman kelola produk.		

3. *Member* memilih aksi (Tambah Data, Edit Data, atau Hapus Data). 4. Jika *Member* memilih Tambah Data, maka akan mengisi form dengan informasi produk yang lengkap. Jika *Member* memilih Edit Data, maka akan mengisi form perubahan data produk terbaru. Jika *Member* memilih hapus Data maka sistem akan memvalidasi data tersebut. 5. *Member* menyimpan form data produk yang telah diisi. 6. Sistem memvalidasi data yang telah dimasukkan pada form. 7. Jika valid, sistem menyimpan/menghapus data produk dari *database*.
***Postconditions*:** Data produk berhasil dikelola sesuai aksi yang dilakukan oleh Admin.
***Sub Flows*: -**
***Alternate / Exceptional Flows*:** Jika data yang diinputkan tidak valid, sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 13. *Use Case Description* Kelola Metode Pembayaran

***Use Case Name*:** Kelola Metode Pembayaran	**ID**: UC-8	***Priority*:** Tinggi
***Actor*:** *Member*		
***Description*:** *Use case* ini menggambarkan proses *Member* untuk mengelola metode pembayaran yang akan disediakan kepada *Customer*, termasuk menambahkan, menghapus metode pembayaran		
***Trigger*:** *Member* ingin memperbarui atau menyesuaikan metode pembayaran yang tersedia bagi *Customer*		

***Preconditions*: *Member* harus sudah *login* ke sistem dan berada di halaman *dashboard Member*.**
***Normal Course*:** 1. *Member* memilih menu "Metode Pembayaran" di *dashboard*. 2. Sistem menampilkan halaman kelola metode pembayaran. 3. *Member* memilih aksi (Tambah Data, atau Hapus Data). 4. Jika *Member* memilih tambah, maka akan mengirim form dengan informasi metode pembayaran. Jika *Member* memilih hapus produk maka sistem akan memvalidasi data tersebut. 5. *Member* menyimpan form data metode pembayaran yang telah diisi. 6. Sistem memvalidasi data yang telah dimasukkan pada form. 7. Jika valid, sistem menyimpan/menghapus data metode pembayaran dari database.
***Postconditions*: Data metode pembayaran berhasil dikelola sesuai aksi yang dilakukan oleh *Member* dan tersedia bagi *Customer* saat melakukan pemesanan.**
***Sub Flows*: -**
***Alternate / Exceptional Flows*:** Jika data yang diinputkan tidak valid, sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 14. *Use Case Description* Kelola Metode Pengiriman

***Use Case Name*:** Kelola Metode Pengiriman	**ID**: UC-9	***Priority*:** Tinggi
***Actor*:** *Member*		

Description**:** *Use case* ini menggambarkan proses *Member* untuk mengelola metode pengiriman yang akan disediakan kepada *Customer*, termasuk menambahkan, mengedit, dan menghapus metode pengiriman.
Trigger**:** *Member* ingin memperbarui atau menyesuaikan metode pengiriman yang tersedia bagi *Customer*
Preconditions**:** *Member* harus sudah *login* ke sistem dan berada di halaman *dashboard Member*.
Normal Course**:** 1. *Member* memilih menu "Metode Pengiriman" di *dashboard*. 2. Sistem menampilkan halaman kelola metode pengiriman. 3. *Member* memilih aksi yang diinginkan: Tambah Data, Edit Data, atau Hapus Data. 4. Jika *Member* memilih tambah, maka akan mengirim form dengan informasi metode pengiriman. Jika *Member* memilih edit, maka akan mengisi form perubahan data metode pengiriman terbaru. Jika *Member* memilih hapus produk maka sistem akan memvalidasi data tersebut. 5. *Member* menyimpan form data metode pengiriman yang telah diisi. 6. Sistem memvalidasi data yang telah dimasukkan pada form. 7. Jika valid, sistem menyimpan/menghapus data metode pengiriman dari *database*.
Postconditions**:** Data metode pengiriman berhasil dikelola sesuai aksi yang dilakukan oleh *Member* dan tersedia bagi *Customer* saat melakukan pemesanan.
Sub Flows**: -**
Alternate / Exceptional Flows**:** Jika data yang diinputkan tidak valid, sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 15. *Use Case Description* Akses Katalog

Use Case Name**:** Akses Katalog	**ID**: UC-10	***Priority*****:** Tinggi
Actor**:** *Guest*, *Customer*		
Description**:** *Use case* ini menggambarkan proses *Guest* dan *Customer* untuk mengakses dan melihat daftar produk yang tersedia dalam katalog aplikasi.		
Trigger**:** Pengguna ingin melihat produk yang tersedia.		
Preconditions**:** Pengguna berada di halaman utama aplikasi atau telah melakukan *login* sebagai *Customer*.		
Normal Course**:** 1. *Guest* atau *Customer* memilih menu katalog atau halaman utama aplikasi. 2. Sistem menampilkan daftar produk yang tersedia lengkap dengan informasi dasar seperti nama produk, harga, dan gambar produk. 3. Jika ingin melihat detail produk, *Guest* atau *Customer* mengklik tombol “Lihat Detail” dan sistem menampilkan detail produk. 4. Jika ingin mencari produk, *Guest* atau *Customer* mengklik ikon atau kotak pencarian pada aplikasi. 5. Pengguna memasukkan kata kunci terkait nama produk yang dicari dan sistem akan memproses permintaan dan menampilkan daftar produk yang sesuai dengan kata kunci yang dimasukkan. 6. Jika ingin menyortir produk, *Guest* atau *Customer* mengklik menu *filter* "Pilih Kategori" dan sistem akan menampilkan opsi kategori produk yang tersedia. 7. *Guest* atau *Customer* memilih salah satu kategori, selanjutnya sistem memproses permintaan dan menampilkan daftar produk yang sesuai dengan kategori yang dipili		
Postconditions**:** Pengguna berhasil melihat daftar produk yang tersedia dalam katalog.		

***Sub Flows*: -**
***Alternate / Exceptional Flows*:** Jika tidak ada produk yang tersedia, sistem menampilkan pesan bahwa produk belum tersedia.
Jumlah Transaksi: Bergantung pada jumlah pengguna yang melihat daftar produk dan interaksi yang mereka lakukan dengan sistem**.**

Tabel 16. *Use Case Description* Kelola Keranjang

***Use Case Name*:** Kelola Keranjang	**ID**: UC-11	***Priority*:** Tinggi
***Actor*:** *Customer*		
***Description*:** *Use case* ini menggambarkan proses *Customer* untuk mengelola keranjang belanja, termasuk menambah produk, mengubah jumlah item, dan menghapus produk dari keranjang.		
***Trigger*:** *Customer* ingin mengelola keranjang belanja sebelum melakukan pemesanan.		
***Preconditions*:** *Customer* harus sudah *login* dan berada di halaman detail produk.		
***Normal Course*:** 1. *Customer* mengklik ikon keranjang. 2. Sistem menampilkan halaman keranjang saya. 3. Jika ingin menambah item, *Customer* mengakses halaman utama atau katalog dan melihat detail produk yang ingin dimasukkan. 4. *Customer* mengklik tombol “Tambah ke keranjang” yang tersedia pada halaman detail produk. 5. Jika ingin mengedit jumlah item, *Customer* mengklik ikon edit dan sistem akan menampilkan counter input. 6. *Customer* mengedit jumlah dan menyimpan perubahan.		

7. Jika ingin menghapus item, *Customer* mengklik ikon hapus dan sistem akan menghapus item dari keranjang..
Postconditions**:** Keranjang belanja berhasil dikelola dan diperbarui di sistem.
Sub Flows**: -**
Alternate / Exceptional Flows**:** Jika terjadi kesalahan saat menambah, mengubah, atau menghapus data pada keranjang keranjang, sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 17. *Use Case Description* Pesan Produk

Use Case Name**:** Pesan Produk	**ID**: UC-12	***Priority*****:** Tinggi
Actor**:** *Customer*		
Description**:** *Use case* ini menggambarkan proses *Customer* untuk memesan produk yang ada di keranjang.		
Trigger**:** *Customer* ingin melakukan pembelian produk yang sudah dimasukkan ke keranjang.		
Preconditions**:** *Customer* harus sudah login dan memiliki produk di keranjang belanja.		
Normal Course**:** 1. *Customer* membuka menu keranjang dan memilih opsi "*Checkout*". 2. Sistem menampilkan halaman checkout yang berisi ringkasan produk yang akan dipesan, pilihan metode pembayaran dan pengiriman, upload bukti pembayaran, dan catatan pesanan. 3. Melengkapi detail pesanan yang diminta.		

4. *Customer* melakukan pemesanan. 5. Sistem memvalidasi pesanan, jika valid sistem akan menetapkan status awal untuk pesanan dan pembayaran. 6. *Customer* dapat membatalkan pesanan jika statusnya belum dikonfirmasi Member. 7. Jika sudah dikonfirmasi dan status pesanan pada sistem telah berubah, maka *Customer* dapat menyelesaikan pesanan.
***Postconditions*:** Pesanan berhasil tercatat dalam sistem dan diteruskan ke *Member*.
***Sub Flows*: -**
***Alternate / Exceptional Flows*:** Jika terjadi kesalahan saat proses pemesanan produk sistem menampilkan pesan kesalahan.
Jumlah Transaksi: Bergantung pada jumlah pengguna memesan produk.

Tabel 18. *Use Case Description* Kelola Pesanan

***Use Case Name*:** Kelola Pesanan	**ID**: UC-13	***Priority*:** Tinggi
***Actor*:** *Member*		
***Description*:** *Use case* ini menggambarkan proses *Member* untuk mengelola pesanan yang masuk dari *Customer*, termasuk melihat detail pesanan, mengubah status pesanan, dan mengkonfirmasi status pembayaran.		
***Trigger*:** Pesanan baru yang masuk dari *Customer*		
***Preconditions*:** *Member* harus sudah *login* ke sistem dan berada di halaman *dashboard Member*.		

***Normal Course*:** 1. *Member* memilih menu " Pesanan" di *dashboard*. 2. Sistem menampilkan halaman kelola pesanan. 3. Jika ingin melihat detail pesanan, *Member* dapat mengakses menu lihat detail. 4. Jika ingin mengedit status pesanan, *Member* dapat mengklik tombol Edit Status Pesanan dan memilih opsi yang tersedia. 5. Jika ingin mengedit status pembayaran, *Member* dapat mengklik tombol Edit Status Pembayaran dan memilih opsi yang tersedia. 6. Sistem menyimpan pembaruan status.
***Postconditions*:** Status pesanan dan pembayaran diperbarui dan *Customer* mendapatkan informasi terkini mengenai pesanan mereka.
***Sub Flows*: -**
***Alternate / Exceptional Flows*:** Jika sistem gagal menyimpan pembaruan status, sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 19. *Use Case Description* Riwayat Pesanan

***Use Case Name*:** Akes Riwayat Pesanan	**ID**: UC-14	***Priority*:** Tinggi
***Actor*:** *Member*, *Customer*		
***Description*:** *Use case* ini menggambarkan proses di mana *Customer* atau *Member* dapat melihat riwayat pesanan mereka.		
***Trigger*:** *Member* atau *Customer* ingin melihat riwayat transaksi mereka di dalam sistem.		

Preconditions**:** *Member* atau *Customer* harus sudah *login* ke sistem.
Normal Course**:** 1. *Customer* atau *Member* memilih menu "Riwayat Pesanan". 2. Sistem menampilkan daftar pesanan yang telah dilakukan oleh pengguna, termasuk detail setiap pesanan. 3. *Customer* atau *Member* dapat mengklik pesanan tertentu untuk melihat detail lebih lanjut.
Postconditions**:** *Customer* atau *Member* dapat melihat detail riwayat pesanan mereka.
Sub Flows**: -**
Alternate / Exceptional Flows**:** Jika tidak ada pesanan yang ditemukan dalam riwayat, sistem menampilkan pesan bahwa belum ada riwayat.
Jumlah Transaksi: -

Tabel 20. *Use Case Description* Memberi Ulasan

Use Case Name**:** Memberi Ulasan	**ID**: UC-15	***Priority*****:** Menengah
Actor**:** *Customer*		
Description**:** *Use case* ini menggambarkan proses *Customer* untuk memberikan ulasan terhadap produk yang sudah dibeli.		
Trigger**:** *Customer* ingin memberikan ulasan setelah menyelesaikan pesanan.		
Preconditions**:** *Customer* harus sudah login dan memiliki produk di keranjang belanja.		
Normal Course**:**		

1. *Customer* memilih menu riwayat pesanan. 2. Sistem menampilkan daftar pesanan. 3. *Customer* mengklik tombol "Beri Ulasan". 4. Sistem menampilkan form ulaasn. 5. *Customer* mengisi komentar dan memberikan rating. 6. *Customer* mengirim ulasan. 7. Sistem memvalidasi data ulasan. 8. Jika valid, sistem menyimpan data dalam database dan menampilkannya detail pada halaman produk.
***Postconditions*: Ulasan produk berhasil disimpan dan ditampilkan di halaman detail produk.**
***Sub Flows*: -**
***Alternate / Exceptional Flows*:** Jika terjadi kesalahan saat memberi ulasan produk sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 21. *Use Case Description* Edit Profil

***Use Case Name*:** Edit Profil	**ID**: UC-16	***Priority*:** Menengah
***Actor*:** *Member, Customer*		
***Description*:** *Use case* ini menggambarkan proses *Member* atau *Customer* untuk mengedit profil.		
***Trigger*:** *Member* atau *Customer* ingin memperbarui informasi profil mereka.		
***Preconditions*:** *Member* harus sudah *login* sebagai *Member* atau *Customer*		

***Normal Course*:** 1. *Member* mengklik ikon akun. 2. *Customer,* mengklik ikon akun selanjutnya memilih menu profil. 3. Sistem akan menampilkan data profil saat ini. 4. *Member* atau *Customer* mengedit data yang ingin diubah. 5. *Member* atau *Customer* menyimpan perubahan. 6. Sistem memvalidasi data profil terbaru. 7. Jika valid, sistem memperbarui data profil pada *database*.
***Postconditions*:** Profil berhasil diperbarui di sistem.
***Sub Flows*: -**
***Alternate / Exceptional Flows*:** Jika terjadi kesalahan saat memperbarui profil sistem menampilkan pesan kesalahan.
Jumlah Transaksi: -

Tabel 22. *Use Case Description* Reset *Password*

***Use Case Name*:** Reset *Password*	**ID**: UC-17	***Priority*:** Tinggi
***Actor*:** *Member*, *Customer*		
***Description*:** *Use case* ini menggambarkan proses pemulihan akun oleh *Customer* atau *Member* yang lupa *password* mereka dengan menggunakan fitur reset *password*.		
***Trigger*:** *Customer* atau *Member* tidak dapat mengingat *password* dan ingin memulihkan akses ke akun mereka.		
***Preconditions*:** *Customer* atau *Member* harus berada di halaman *login* dan memilih opsi "Lupa Password".		

***Normal Course*:**

1. *Customer* atau *Member* memilih opsi "Lupa *Password*" di halaman *login*.
2. Sistem menampilkan opsi *role* pengguna.
3. *Member* atau *Customer* memilih *role.*
4. Sistem meminta pengguna memasukkan *email* yang terdaftar di akun.
5. Pengguna memasukkan *email* yang terdaftar.
6. Sistem mengirimkan email dengan tautan untuk reset *password* ke email yang terdaftar.
7. *Customer* atau *Member* membuka *email* dan mengklik tautan reset *password*.
8. Sistem menampilkan halaman reset *password* yang baru.
9. *Customer* atau *Member* memasukkan *password* baru dan konfirmasi *password*.
10. Pengguna menyimpan data *password* baru.
11. Sistem memvalidasi data *password.*
12. Jika valid, sistem memperbarui data *password* lama dengan *password* baru di *database*.

***Postconditions*:** *Password* pengguna berhasil diperbarui dan pengguna dapat *login* dengan *password* baru.

***Sub Flows*: -**

***Alternate / Exceptional Flows*:**

1. Jika *email* yang dimasukkan tidak ditemukan dalam sistem, sistem menampilkan pesan kesalahan dan meminta pengguna untuk memeriksa kembali *email* yang dimasukkan.
2. Jika *password* dan konfirmasi *password* tidak cocok, sistem akan menampilkan pesan kesalahan dan meminta pengguna untuk memasukkan data yang cocok.

Jumlah Transaksi: -

3.4.2. *Planning*

Pada tahap *planning* pengembang merencanakan tugas-tugas yang akan dilakukan pada setiap iterasi serta merinci perangkat lunak dan perangkat keras yang dibutuhkan selama pengambangan aplikasi.

3.4.2.1. Daftar Rencana Tugas

Dilakukan perencanaan daftar tugas yang akan dikerjakan beserta estimasi waktu yang dibutuhkan berdasarkan *requirements* yang telah didapatkan. Adapun daftar tugas yang direncanakan dan estimasi waktu pengerjaannya disajikan dalam Tabel 23.

Tabel 23. Daftar Rencana Tugas

Iterasi	Rencana Tugas	Fitur	Estimasi Waktu
1	Pengembangan aplikasi *back-end* dan REST API tahap pertama	Registrasi Login Logout Kelola *Member* Kelola *Customer* Kelola Kategori Kelola Produk Kelola Metode Pembayaran Kelola Metode Pengiriman Katalog Produk	14 hari
2	Pengembangan aplikasi *back-end* dan REST API tahap kedua	Kelola Keranjang Pesan Produk Kelola Pesanan Riwayat Pesanan Memberi Ulasan Edit Profil	21 hari

Iterasi	Rencana Tugas	Fitur	Estimasi Waktu
		Reset Password	
3	Pengembangan aplikasi *front-end* (web)	Membuat aplikasi *front-end* berbasis web yang meliputi tampilan untuk Admin, *Member*, *Customer*, dan tampilan yang bersifat publik.	28 hari

3.4.2.2. Daftar Perangkat Lunak dan Perangkat Keras

Alat yang digunakan dalam penelitian ini meliputi perangkat lunak (*software*) dan perankat keras (*hardware*). Perangkat lunak yang digunakan dalam melakukan penelitian ini antara lain:

1. Sistem Operasi Windows 11 Home Single Language
2. Visual Studio Code versi 1.85.0
3. Node.js versi 16.17.0
4. NPM versi 10.5.0
5. HAPI versi 21.3.3
6. Prisma versi 5.11.0
7. Joi versi 17.12.2
8. JWT versi 3.2.0
9. Whimsical Web
10. React versi 18.2.0
11. Vite versi 5.2.0
12. Axios versi 1.7.1
13. Git versi 2.38.1 dan Github Web
14. phpMyAdmin versi 5.2.1
15. XAMPP versi 8.0.12-0
16. Postman versi 10.21.0
17. ZAP versi 2.14.0
18. Apache Jmater versi 5.6.3

Penelitian ini menggunakan sebuah laptop dengan spesifikasi berikut sebagai perangkat keras yang digunakan dalam pengembangan sistem.

1. Processor 12th Gen Intel(R) Core(TM) i5-1235U 1.30 GHz
2. Memori 8GB RAM.
3. Penyimpanan SSD 512 GB.

3.4.3. *Iteration Initialization*

Tahap ini merupakan awal dari setiap iterasi yang akan dilakukan. Pengembang akan memilih tugas yang akan dikerjakan sebagai fokus iterasi saat ini dan memastikan pengerjaan dilaksanakan sesuai estimasi yang telah ditetapkan. Dalam penelitian ini, iterasi akan dilakukan sebanyak tiga kali dengan tugas dan waktu yang bervariasi.

3.4.4. *Design*

Tahap desain dilakukan pada proses pengembangan aplikasi *back-end* maupun *front-end*. Pada pengembangan aplikasi *back-end* dilakukan perancangan desain basis data beserta relasi antar tabelnya serta rancangan REST API dari fitur yang dikembangkan. Sementara itu, untuk pengembangan *front-end* web, dilakukan proses desain tampilan antarmuka berupa *wireframe* sebagai acuan dalam menerapkan tata letak yang memudahkan pengguna. Desain *wireframe* dibuat menggunakan *tools online* yaitu Whimsichal karena gratis dan mudah digunakan.

3.4.5. *Implementation*

Pada tahapan implementasi dilakukan pembuatan kode sesuai dengan rancangan yang telah dibuat. Proses implementasi aplikasi *back-end* menggunakan Node.js dengan *framework* HAPI yang dimulai dari konfigurasi *database,* pembuatan *server,* hingga pengembangan fungsionalitas API. Sementara itu, proses implementasi aplikasi *front-end* menggunakan *library* React yang meliputi pembuatan *User Interface* hingga integrasi API. *Source code* disimpan dalam *remote repository* yaitu GitHub agar lebih aman serta dapat mempermudah manajemen dan *tracking* kode.

3.4.6. *System Testing*

Terdapat beberapa jenis pengujian yang akan dilakukan pada tahap ini, yaitu pengujian fungsionalitas, pengujian keamanan, dan pengujian kinerja. Pengujian fungsionalitas dilakukan terhadap semua *endpoint* API yang telah dibuat dengan menggunakan *tools* Postman. Dalam pengujian ini fungsionalitas sistem akan diuji untuk memastikan kesesuaian *request* dan *response* yang diharapkan. Untuk memastikan fungsi dapat berjalan dengan baik dan tidak menimbulkan *bug* atau *error*. Postman juga digunakan untuk menguji terkait aspek keamanan autentikasi dan otorisasi dari *endpoint* API. Sementara itu, untuk kerentanan sistem akan diuji menggunakan alat ZAP. Pada proses ini akan diidentifikasi kerentanan sistem terhadap ancaman keamanan yang tidak diinginkan. Sistem juga akan diuji kinerjanya menggunakan alat Apache Jmeter untuk mengetahui kestabilan sistem saat terdapat banyak pengguna yang mengakses dan seberapa cepat waktu respons yang diberikan.

3.4.7. *Retrospective*

Di akhir setiap iterasi, terdapat tahap *retrospective* untuk melakukan evaluasi terhadap iterasi yang baru saja dilakukan. Pada tahap ini, umumnya akan dievaluasi kesesuaian waktu pengerjaan dengan estimasi awal. Jika terdapat keterlambatan, akan ditinjau penyebabnya sebagai bahan perbaikan untuk meningkatkan kualitas proses iterasi berikutnya.

V. SIMPULAN DAN SARAN

5.1. Simpulan

Simpulan dari penelitian yang telah dilakukan adalah penelitian ini berhasil mengembangkan aplikasi pemasaran produk usaha mahasiswa FMIPA Universitas Lampung berbasis web dan dapat memfasilitasi dan memudahkan proses kelola data dan pemasaran produk di lingkungan Universitas Lampung dan sekitarnya.

5.2. Saran

Berdasarkan implementasi sistem yang telah dikembangkan, terdapat beberapa saran untuk pengembangan lebih lanjut agar sistem dapat menjadi lebih baik:

1. Mengembangkan fitur lanjutan seperti notifikasi dan *chat* untuk memudahkan komunikasi dan interaksi pengguna.
2. Mengintegrasikan sistem dengan layanan penyedia metode pembayaran dan pengiriman agar proses transaksi menjadi lebih efisien.
3. Menerapkan desain sistem yang responsif pada semua perangkat agar pengalaman pengguna lebih optimal.
4. Melakukan pengujian yang lebih mendalam terhadap seluruh sistem untuk memastikan kualitas dan keandalannya.

DAFTAR PUSTAKA

Adi, R. P., Koswara, Y., Tashika, J., Devi, Y., & Saifudin, A. (2020). Pengujian Black Box pada Aplikasi Pertokoan Minimarket Menggunakan Metode Equivalence Partitioning. *Jurnal Teknologi Sistem Informasi dan Aplikasi*, *3*(2), 100. https://doi.org/10.32493/jtsi.v3i2.4695

Apache Software Foundation. *Apache Jmeter*. https://jmeter.apache.org diakses pada tanggal 1 Februari 2024.

Ardilla, S., & Hadinata, N. H. (2022). Sistem informasi marketplace produk usaha mikro kecil menengah (umkm). *Jurnal Digital Teknologi Informasi*, *5*(2), 86. https://doi.org/10.32502/digital.v5i2.4986

Ariyanto, A., Bangun, R., Indillah, M. R. M., Trenggana, A. F. M., Sholihah, D. R., Ariyanti, M., Widiati, E., Irawan, P., Ratih, S. D., & Ismail, R. S. (2023). *Manajemen Pemasaran*.

Axios. *What is Axios?*. https://axios-http.com diakses pada tanggal 26 Agustus 2024.

Baenil Huda, & Tukino. (2023). Mendorong Pertumbuhan UMKM Melalui Platform Digital. *Jurnal Buana Pengabdian*, *5*(2), 86–91. https://doi.org/10.36805/jurnalbuanapengabdian.v5i2.5791

Choirudin, R., & Adil, A. (2019). Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa. *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, *18*(2), 284–293. https://doi.org/10.30812/matrik.v18i2.407

Doglio, F. (2018). *REST API Development with Node.js*. Apress. https://doi.org/10.1007/978-1-4842-3715-1

Dzhurov, Y., Krasteva, I., & Ilieva, S. (2009). *Personal Extreme Programming– An Agile Process for Autonomous Developers*.

Fatiyah, A. C., Gumilang, S. F. S., & Witarsyah, D. (2019). Perancangan Website Marketplace Pada Startup Borongajayuk Menggunakan Metode Iterative Dan Incremental. *eProceedings of Engineering*, *6*(2).

Firmansyah, M. A. (2023). *Pemasaran Produk dan Merek: Planning & Strategy*. Penerbit Qiara Media.

Gunawan, R., & Rahmatullah, A. (2019). JSON Web Token (JWT) untuk Authentication pada Interoperabilitas Arsitektur berbasis RESTful Web Service. *JEPIN*, *5*(1).

Hady, E. L., Haryono, K., & Rahayu, N. W. (2020). User Acceptance Testing (UAT) pada Purwarupa Sistem Tabungan Santri (Studi Kasus: Pondok Pesantren Al-Mawaddah). *Jurnal Ilmiah Multimedia dan Komunikasi*, *5*(1).

Halili, E. H. (2008). *Apache JMeter*.

Harrison M. (2016). *hapi.js in Action*. Manning Publications.

Heningtyas, Y., Dandi, L. W., Hijriani, A., & Hermanto, B. (2022). Pengembangan Aplikasi Android untuk Meningkatkan Efisiensi dan Efektivitas Proses di Sentra Inovasi dan Inkubator Bisnis. *Jurnal Pepadun*, *3*(3), 371–379. https://doi.org/10.23960/pepadun.v3i3.135

Herayono, A., & Adri, M. (2021). Pengembangan Student Marketplace Bagi Mahasiswa Wirausaha Unp. *JAVIT : Jurnal Vokasi Informatika*, 38–46. https://doi.org/10.24036/javit.v1i2.23

Irwansyah, M. A., Novriando, H., & Apriandi, R. (2021). Analisis User Experience Aplikasi Bujang Kurir Menggunakan Google Analytics (GA). *JEPIN (Jurnal Edukasi dan Penelitian Informatika)*, *7*(1), 64–69.

Jain, J. (2022). *Learn API Testing*. Apress. https://doi.org/10.1007/978-1-4842-8142-0

JWT. *Introduction to JSON Web Tokens*. https://jwt.io/introduction diakses pada tanggal 9 Desember 2023.

Khairina, D. M. (2011). Analisis Keamanan Sistem Login. *Jurnal Informatika Mulawarman*, *6*(2), 64–67.

Kurniawan, I., Humaira, & Rozi, F. (2020). REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android. *JITSI : Jurnal Ilmiah Teknologi Sistem Informasi*, *1*(4), 127–132. https://doi.org/10.30630/jitsi.1.4.18

Kurniawan, Y. I., & Kusuma, A. F. S. (2021). Aplikasi augmented reality untuk pembelajaran salat bagi siswa sekolah dasar. *J. Teknol. Inf. dan Ilmu Komput*, *8*(1), 7–14.

Mukaromah, S. (2019). Rancang Bangun Sistem Informasi Kewirausahaan Mahasiswa. *ReTII*, 278–284.

Nashikhuddin, A. Y., Karaman, J., & Litanianda, Y. (2023). Implementasi Api Restful Dengan Json Web Token (Jwt) Pada Aplikasi E-Commerce Thrifty Shop Untuk Otentikasi Dan Otorisasi Pengguna. *METHOMIKA Jurnal Manajemen Informatika dan Komputerisasi Akuntansi*, *7*(2), 239–246. https://doi.org/10.46880/jmika.Vol7No2.pp239-246

Nopriandi, H. (2018). Perancangan Sistem Informasi Registrasi Mahasiswa. *Jurnal Teknologi Dan Open Source*, *1*(1), 73–79. https://doi.org/10.36378/jtos.v1i1.1

Perdana, M. A. K. (2018). Pengembangan REST API Layanan Penyimpanan menggunakan Metode Rapid Application Development (Studi kasus PT. XYZ). *InfoTekJar (Jurnal Nasional Informatika dan Teknologi Jaringan)*, *3*(1), 100–104. https://doi.org/10.30743/infotekjar.v3i1.563

Perry, W. E. (2007). *Effective Methods for Software Testing, CafeScribe: Includes Complete Guidelines, Checklists, and Templates*. John Wiley & Sons.

Postman. *What is Postman?*. https://www.postman.com/product/what-ispostman diakses pada tanggal 9 Desember 2023.

Rachmadi, T., & Kom, S. (2020). *Sistem Basis Data* (Vol. 1). Tiga Ebook.

Rafiq, M., Ashraf, R., & Abid, H. (2020). Automated VS. Manual Testing: A Scenario Based Approach Towards Application Development. *Gyancity Journal of Electronics and Computer Science*, *5*(1), 47–55.

React. *React: The library for web and native user interfaces*. https://react.dev/ diakses pada 1 Maret 2024.

RevoU. *Apa itu Whimsical?*. https://revou.co/kosakata/whimsical diakses pada 26 Agustus 2024.

Rombe, A. N., Aksara, L. F., & Surimi, L. (2019). Analisis perbandingan real time streaming protocol (RTSP) dan hypertext transfer protocol (HTTP) pada layanan live video streaming. *semanTIK*, *5*(1), 149–156.

Saputra, A. (2020). *CAMI: Aplikasi Uji Validitas dan Reliabilitas Instrumen Penelitian Berbasis Web*. Yayasan Ahmar Cendekia Indonesia.

Simatupang, J., & Sianturi, S. (2019). Perancangan sistem informasi pemesanan tiket bus pada po. Handoyo berbasis online. *Jurnal Intra-Tech*, *3*(2), 11–25.

Siregar, L. (2020). Review Pengujian Keamanan Perangkat Lunak dalam Software Development Life Cycle (SDLC). *Journal of Applied Sciences, Electrical Engineering and Computer Technology*, *1*(3), 1–11. https://doi.org/10.30871/aseect.v1i3.2380

Sri Susilawati, P., Hilal, F., Aulia Azzahra, N., & Luthfiah Nurlaeli, S. (2022). Perbandingan Strategi Pemasaran Online Dan Offline di Era Pandemi Covid-19. *Al-Kharaj : Jurnal Ekonomi, Keuangan & Bisnis Syariah*, *5*(3), 1080–1083. https://doi.org/10.47467/alkharaj.v5i3.1480

Suasapha, A. H. (2020). *Skala Likert untuk penelitian pariwisata; beberapa catatan untuk menyusunnya dengan baik. JURNAL KEPARIWISATAAN, 19 (1), 26–37*.

Sud, K. (2020). *Practical hapi*. Apress. https://doi.org/10.1007/978-1-4842-5805-7

Supardi, Y. (2021). *Semua Bisa Menjadi Programmer JavaScript & Node.js*. Elex Media Komputindo.

Suprapto, F. R., Marthasari, G. I., & Nuryasin, I. (2024). Sistem Informasi Penjualan dan Pelelangan Berbasis Web pada Ricardo Corner MLG Menggunakan Metode Personal eXtreme Programming (PXP). *Jurnal Repositor*, *2*(11). https://doi.org/10.22219/repositor.v2i11.30972

Syahputri, K., & Nasution, M. I. P. (2023). Peran Database Dalam Sistem Informasi Manajemen. *Jurnal Akuntansi Keuangan Dan Bisnis*, *1*(2), 54–58.

ZAP Dev Team. *Intro to ZAP*. https://www.zaproxy.org/getting-started/ diakses pada 31 Januari 2024.

Zulfa, L., & Hidayati, R. (2018). Analisis pengaruh persepsi risiko, kualitas situs web, dan kepercayaan konsumen terhadap keputusan pembelian konsumen e-commerce Shopee Di Kota Semarang. *Diponegoro Journal of Management*, *7*(3), 1–11.