BAB II
LANDASAN TEORI

2.1. Tanaman Alpukat

Alpukat (Perseaamericana mill) merupakan tanaman yang dapat tumbuh subur di daerah tropis seperti Indonesia dan merupakan salah satu jenis buah yang digemari masyarakat karena selain rasanya yang enak juga kandungan antioksidannya yang tinggi [2]. Buah alpukat merupakan buah yang sering kita jumpai. Buah serbaguna ini memiliki banyak manfaat dan khasiat bagi manusia. Ada banyak zat yang kaya manfaat yang terdapat di buah ini.. Rasanya yang nikmat membuat banyak orang menyukainya. Seperti buah pada umumnya, alpukat memiliki tingkat kematangan tersendiri. Kematangan buah alpukat dapat dilihat dari warna kulit maupun warna daging buahnya. Kemampuan akan membedakan tingkat kematangan buah alpukat tentu saja dibutuhkan pengamatan yang akurat. Hal ini disebabkan karena alpukat memiliki warna yang hampir mirip antara alpukat matang dengan setengah matang (Hanafi, Fadillah and Insan, 2019).

2.2. Sistem Pakar

Sistem pakar adalah suatu program komputer atau sistem informasi yang mengandung beberapa pengetahuan dari satu atau lebih pakar manusia terkait suatu bidang yang cenderung spesifik.Pakar yang dimaksudkan merupakan seseorang yang memiliki keahlian khusus di bidangnya (Sucipto *et al.*, 2019).

Sistem pakar adalah suatu program komputer atau sistem informasi yang mengandung beberapa pengetahuan dari satu atau lebih pakar manusia terkait suatu bidang yang cenderung spesifik. Pakar yang dimaksudkan merupakan seseorang yang memiliki keahlian khusus di bidangnya masing-masing, contohnya dokter, psikolog, mekanik, dan lain sebagainya. Perangkat lunak ini pertama kali dikembangkan oleh periset program kecerdasan buatan (AI) sekitar

tahun 1960-an dan 1970-an, serta baru diterapkan pada tahun 1980-an (Noviani, Prambudi and Mulyadi, 2020)

2.3. Metode *Forward Chaining*

Forward chaining: Pencocokan fakta atau pernyataan dimulai dari bagian sebelah kiri dulu (IF dulu). Dengan kata lain penalaran dimulai dari fakta terlebih dahulu untuk menguji kebenaran hipotesis. Kadang disebut:data-driven karena inference engine menggunakan informasi yang ditentukan oleh user untuk memindahkan ke seluruh jaringan dari logika „AND" dan „OR" sampai sebuah terminal ditentukan sebagai objek. Bila inference engine tidak dapat menentukan objek maka akan meminta informasi lain. Aturan (*Rule*) di mana menentukan objek, membentuk path (lintasan) yang mengarah ke objek. Oleh karena itu, hanya satu cara untuk mencapai satu objek adalah memenuhi semua aturan (Permana and Saputra, 2019).

Langkah – langkah dalam membuat sistem pakar dengan menggunakan metode *forward chaining* yaitu Menurut (Riskadewi dan Hendrik, 2005) :

a. Pendefenisan masalah dimulai dengan pemilihan domain masalah dan akuisi pengatahuan.
b. Pendefenesian data input untuk memulai inferensi karena diperlukan oleh sistem forward chaining.
c. Pendefenisian struktur pengendalian data untuk membantu mengendalikan pengaktifan suatu aturan.
d. Penulisan kode awal dalam domain pengatahuan.
e. Pengujian sistem agar dapat mengatahui sejauh sistem berjalan mana
f. Perancangan antarmuka dengan basis pengatahuan
g. Pengembangan system h. Evaluasi system

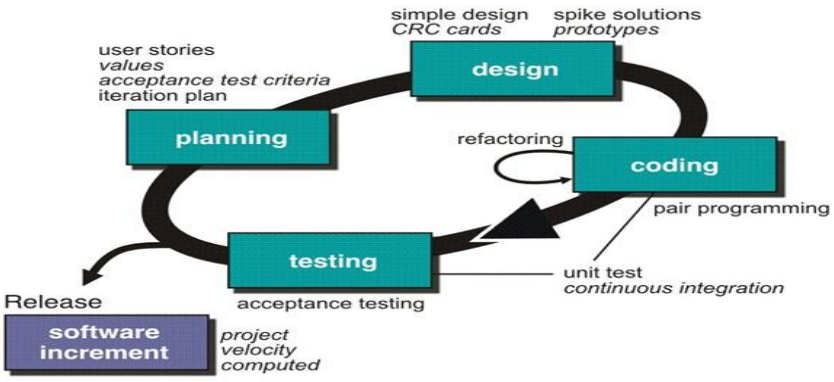
2.4. Metode Pengembangan Sistem

XP adalah metodologi pengembangan perangkat lunak yang ditujukan untuk meningkatkan kualitas perangkat lunak dan tanggap terhadap perubahan kebutuhan pelanggan. Jenis pengembangan perangkat lunak semacam ini

dimaksudkan untuk meningkatkan produktivitas dan memperkenalkan pos pemeriksaan dimana persyaratan pelanggan baru dapat diadopsi (Pressman, 2015).

Tahapan-tahapan dari *Extreme Programming* terdiri dari *planning* seperti memahami kriteria pengguna dan perencanaan pengembangan, *designing* seperti perancangan *prototype* dan tampilan, *coding* termasuk pengintegrasian dan yang terakhir adalah testing. Unsur-unsur lain dari *Extreme Programming* meliputi *paired programming* pada tahapan *coding*, unit testing pada semua kode, penghindaran pemrograman fitur kecuali benar-benar diperlukan, struktur manajemen yang datar, kode yang sederhana dan jelas, dan seringnya terjadi komunikasi antara programmer dan pelanggan ketika terjadi perubahan kebutuhan pelanggan seiring berlalunya waktu berlalu.

Metode ini membawa unsur-unsur yang menguntungkan dari praktek rekayasa perangkat lunak tradisional ke tingkat "ekstrem", sehingga metode ini dinamai *Extreme Programming*. Unsur-unsur yang menjadi karakteristik metodologi adalah kesederhanaan, komunikasi, umpan balik, dan keberanian. Gambar tahapan XP dapat dilihat pada gambar 2.1:



Gambar 2. 1 Tahapan Extreme Programming

Dibawah ini adalah penjelasan tahapan *Extreme Programming* yaitu :

1. *Planning* (Perencanaan)

Kegiatan Perencanaan disebut juga *planning game* biasanya dimulai dengan mendengarkan suatu kegiatan yang bertujuan mengumpulkan kebutuhan-kebutuhan untuk memahami konteks bisnis dan perlunya keluaran-keluaran (*output*), fungsi utama dan fungsionalitas.

Pada perencanaan terdapat *user stories values* yaitu story dengan value tertinggi akan dipindahkan dari jadwal dan diimplementasikan pertama, *acceptance test criteria interation plan* melakukan perhitungan kecepatan *project* selama *development*, kustomer dapat menambah story, merubah value, membagi story atau menghapusnya.

2. *Design* (Perancangan)

Perancangan yang menarik, dan sederhana selalu memberikan hasil yang lebih disukai daripada gambaran-gambaran yang lebih kompleks. Perancangan XP memberikan panduan implementasi untuk suatu cerita ketika ditulis, tidak kurang, tidak lebih.

Terdapat *simple design CRC Casrds* untuk mengenali dan mengatur *object oriented class* sesuai dengan *software increment* dan *spike solutions prototypes* melakukan spesifikasi solusi dari *object oriented class.*

3. *Coding* (Pengkodean)

Pengkodean ini dilanjutkan setelah cerita yang telah dikembangkan dan rancangan yang telah dilakukan oleh tim perangkat lunak. Pengkodean ini tidak langsung mengarah ke kode-kode program. Tim akan mengembangkan serangkaian unit pengujian lalu beralih ke pengkodean.

Pada tahapan *pair programming* melakukan kerja sama untuk membuat code dari satu story. Dan *refactoring* adalah proses restrukturisasi kode program computer yang ada tanpa mengubah perilaku eksternalnya.

4. *Testing* (Pengujian)

Unit pengujian yang harus dibuat dan kemudian dijalankan menggunakan kerangka kerja yang memungkinkan mereka untuk diotomatisasi sehingga dapat dijalankan dengan mudah dan berulang kali.

Pada tahapan pengujian yaitu *unit test continous integration* yaitu tahapan pengujian code yang diintegrasikan dengan kerja lainnya dengan pengujian yang dilakukan oleh customer dan fokus pada keseluruhan dan fungsional sistem dan *acceptance testing* yaitu pengujian yang dilakukan *customer stories* yang akan diimplementasikan sebagai bagian dari *software realease.*

Selanjutnya terdapat tahapan *software increment project velocity computed* yaitu tahapan yang telah diimplementasikan dari *software realease* yang nantinya akan diterapkan dalam suatu sistem.

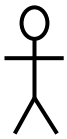
2.5. Bahasa Pemodelan Pengembangan Sistem (*UML*)

Bahasa Pemodelan Pengembangan Sistem (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement* (kebutuhan), membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa & Shalahuddin, 2018).

2.5.1. *Use Case Diagram*

Use case diagram atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat (Rosa & Shalahuddin, 2018). *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat menjelaskan simbol-simbol yang ada pada diagram *use case* dapat dilihat pada gambar 2.1 di bawah ini:

Tabel 2. 1 Simbol Diagram *Use Case*

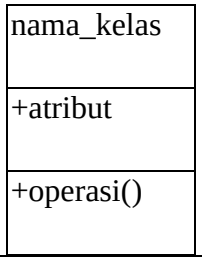
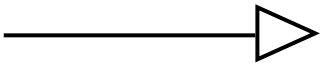
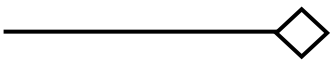
Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*
Aktor/*actor* 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan

	menggunakan kata benda di awal frase nama *actor*
Asosiasi/*association* 	Komunikasi antara aktor dan *use case* yang berpatisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor*
Ekstensi/*extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan
<<*extend*>> 	dapat berdiri sendiri walau tanpa *use case* tambahan itu mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek biasanya *use case* tambahan memiliki nama depan
Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan/*Include/uses* <<*include*>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini

2.5.2. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi (Rosa and Shalahudin, 2018). menjelaskan simbol-simbol yang ada pada diagram kelas pada tabel *class diagram* 2.2.

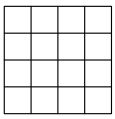
Tabel 2. 2 Simbol *Class Diagram*

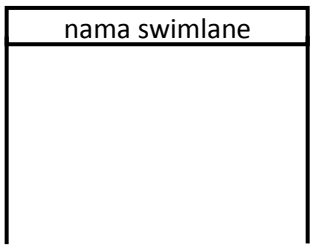
Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antarmuka/*Interface*  nama_*interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi/*asociation* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah/*directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi(umum khusus)
Kebergantungan/*dependecy* 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi/*agregation* 	Relasi antar kelas dengan makna semua bagian (*whole-part*)

2.5.3.*Activity Diagram*

Activity diagram atau Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Rosa & Shalahuddin, 2018), menjelaskan Simbol-simbol yang ada pada *activity diagram*dapat dilihat pada tabel 2.3 di bawah ini :

Tabel 2. 3 Simbol *Activity Diagram*

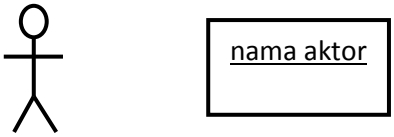
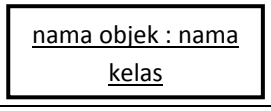
Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem,sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan/*join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Tabel 	Suatu file komputer dari mana data bisa dibaca atau direkam selama kejadian bisnis
Dokumen 	Menunjukan dokumen sumber atau laporan
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

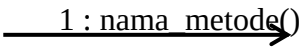
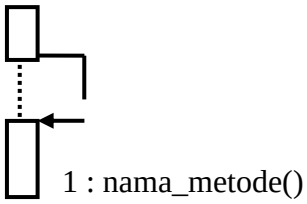
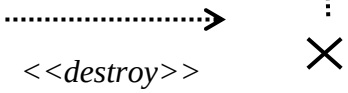
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

2.5.4. *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram *sequen*ce maka harus diketahui objek-objek yang terlibat dalam sebuah *use case.* simbol-simbol yang ada pada *sequence* digram dapat dilihat pada tabel 2.4 di bawah ini :

Tabel 2. 4 Simbol *Sequence Diagram*

Simbol	Deskripsi
Aktor 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri,
Atau nama aktor tanpa waktu aktif	jadi walaupun simbol dari aktor gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda
Garis hidup/*lifeline* 	Menyatakan kehidupan suatu objek
Objek 	Menyatakan objek yangberinteraksi pesan

Simbol	Deskripsi
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
Pesan tipe *create* 	Menyatakan suatu objek membuat objek lain, arah panah objek yang dibuat
Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri 
Pesan tipe *send* 	Menyatakanbahwa suatu objek mengirimkandata/masukkan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

2.6. Alat Implementasi

2.6.1. Xampp

Menurut MADCOMS (2016) Xampp adalah sebuah paket kumpulan software yang terdiri dari Apache, MySQL, PhpMyAdmin, PHP, Perl, Filezilla, dan lain.

Xampp berfungsi untuk memudahkan instalasi lingkungan PHP, di mana biasanya lingkungan pengembangan web memerlukan PHP, Apache, MySQL dan PhpMyAdmin (MADCOM, 2016).

2.6.2. *MySQL*

Menurut MADCOM (2016) MySQL adalah sistem manajemen Database SQL yang bersifat *Open Source* dan paling popular saat ini. Sistem Database MySQL mendukung beberapa fitur seperti multithreaded, multiuser dan SQL Database managemen system (DBMS).

2.7. PHP

Menurut Arif (2011), PHP (*Personal Home Page*) adalah bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah web dan bisa digunakan pada HTML. PHP merupakan singkatan dari "PHP: *Hypertext Prepocessor*", dan merupakan bahasa yang disertakan dalam dokumen HTML sekaligus bekerja di sisi *server* (*server-side* HTML-*embedded scripting*). Artinya sintaks dan perintah yang diberikan akan sepenuhnya dijalankan di *server* tetapi disertakan pada halaman HTML biasa, sehingga script-nya tak tampak di sisi *client* (Destiningrum and Adrian, 2017)..

PHP dirancang untuk dapat bekerja sama dengan database server dan dibuat sedimikian rupa sehingga pembuatan dokumen HTML yang dapat mengakses database menjadi begitu mudah. Tujuan dari bahasa scripting ini adalah untuk membuat aplikasi dimana aplikasi tersebut yang dibangun oleh PHP pada umumnya akan memberikan hasil pada web *browser*, tetapi prosesnya secara keseluruhan dijalankan di server.

2.8. Pengujian Sistem *Black Box*

Pengujian *black-box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program (Howden, 2019). Pengujian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut :

1. Fungsi – fungsi yang tidak benar atau hilang,
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

Kelebihan *Black Box* adalah *black box testing* dapat menguji keseluruhan fungsionalitas perangkat lunak. *Black box testing* dapat memilih *subset test* yang secara efektif dan efisien dapat menemukan cacat. Dengan cara ini *black box testing* dapat membantu memaksimalkan *testing investment* (Howden, 2019).

2.9. Tinjauan Pustaka

Berikut ini adalah beberapa literatur yang digunakan dalam penelitian, dapat dilihat pada berikut :

1. Permana and Saputra (2019) meneliti tentang Implementasi Sistem Pakar Hama Pada Tanaman Alpukat Mentega Berbasis Web Menggunakan Metode Forward Chaining Di Cv.Romaco Jaya. Gejala ama dan penyakit Sangat banyak ditemukan pada alpukat mentega, tetapi masalahnya adalah apakah hama atau penyakit tersebut menimbulkan kerugian yang berarti atau tidak. Untuk mengetahui jenis serangan apa yang terjadi pada alpukat mentega terkadang petani minim pengetahuan di bidang informasi hama penyakit sehingga ada kalanya untuk pengobatan tidak sesuai dengan jenis serangan sehingga penyakit tetap merebak ke tanaman lain. Kemajuan sistem pakar dapat mengatasi permasalahan ini yaitu dengan merancang sebuah sistem komputer berbasis web yang terintegrasi dengan database dan bahasa pemrograman seperti PHP-MySql sehingga dapat membantu petani dalam mendiagnosa penyakit yang terserang. Aplikasi sistem pakar dalam

pengambilan keputusannya menggunakan mesin inferensi forward chaining (penalaran maju) dimana kasus ditelusuri pada stat awal sehingga akan ditemuan goal untuk hasil akhir diagnosa penyakit alpukat mentega. Hasil dari implementasi sistem yaitu sistem memberikan pertanyaan berupa gejala-gejala yang harus dijawab oleh petani dan hasil dari proses tersebut sistem akan memberikan informasi penyakit apa yang menyerang tanaman serta solusi untuk penaganannya.

2. Tyar and Wahyuddin (2022) meneliti tentang Sistem Pakar Menggunakan Metode Naïve Bayes dan Certainty Factor untuk Mendeteksi Hama pada Tanaman Alpukat Berbasis Web.Di Indonesia sendiri, Alpukat banyak di temui karena iklim yang tropis, lalu masyarakat juga banyak yang menggemari buah alpukat karna bisa di jadikan berbagai macam bahan makanan dan minuman. Kurangnya perhatian untuk budidaya dan pengembang biak an buah alpukat menjadikan hasil panen jadi kurang baik, dan tidak optimal. maka penulis melakukan peneletian ini dengan harapan agar hasil panen dan pengembangbiak an tanaman Alpukat dapat lebih maksimal. Sistem pakar tersebut didirikan dengan memakai pemrograman berbasis Web. Perbandingan Metode yang diterapkan yaitu Naïve Bayes dan Certainty Factor (CF) untuk lebih spesifik teknik yang memanfaatkan wawasan dan kemungkinan dalam meramalkan bukan tanaman alpukat memiliki gangguan pertumbuhan berdasarkan adanya hama (ulat) yang ada di tanaman Alpukat. Dengan adanya fasilitas ini para petani dapat lebih mudah mendeteksi hama pada tanaman alpukat.
3. Sholikhah, Kurniadi and Riansyah (2021) meneliti tentang Sistem Pakar Menggunakan Metode Forward Chaining untuk Diagnosa Hama dan Penyakit Tanaman Padi. Salah satu kendala atau penghambat dalam pertanian padi adalah hama dan penyakit yang menyerang tanaman padi. Namun pengetahuan para petani padi tentang hama dan penyakit serta pengendaliannya masih kurang. Di sisi lain jumlah pakar tentang hama dan penyakit padi terbatas. Salah satu ilmu dalam bidang teknologi yang dapat menjadi solusi pada masalah tersebut yaitu ilmu sistem pakar. Dalam pembuatan sistem ini dibutuhkan pakar yang ahli di bidang hama dan

penyakit tanaman padi untuk mendapatkan data-data yang akurat. Metode yang akan digunakan dalam penelitian ini adalah metode Forward Chaining untuk menentukan jenis-jenis hama dan penyakit tanaman padi. Input yang dibutuhkan adalah gejala atau ciri-ciri yang muncul pada tanaman padi. Basis pengetahuan dibangun dengan menggunakan kaidah produksi (IF_THEN). Nilai akan diperoleh dari aturan (rule) untuk gejala atau ciri-ciri yang digabungkan. Hasil dari penggabungan ini merupakan output solusi hama dan penyakit tanaman padi. Sistem pakar ini akan mempermudah petani untuk mengetahui jenis hama dan penyakit tanaman padi serta cara pengendaliannya.