BAB 2 TINJAUAN PUSTAKA

2.1 Deskripsi Masalah

Pertanian hortikultura skala kecil menghadapi tantangan besar akibat perubahan iklim, keterbatasan sumber daya, serta ancaman hama dan penyakit tanaman. Perubahan iklim meningkatkan ketidakpastian pola cuaca, yang berdampak pada ketersediaan air, produktivitas tanaman, dan kerentanan terhadap serangan hama [10]. Selain itu, keterbatasan akses teknologi modern bagi petani skala kecil memperburuk kesenjangan produktivitas pertanian [11]. Untuk mengatasi hal ini, teknologi *Internet of Things* dipandang sebagai solusi potensial. IoT memungkinkan pengumpulan data *real time* dari sensor suhu, kelembaban, maupun tegangan, yang dapat membantu petani dalam monitoring dan pengambilan keputusan berbasis data [12].

Namun, implementasi IoT dalam rumah kaca tidak lepas dari kendala konektivitas internet. Banyak wilayah pedesaan tidak memiliki akses internet stabil, sehingga sistem monitoring berbasis *cloud* saja tidak memadai. Oleh karena itu, diperlukan pendekatan hybrid, di mana perangkat seperti ESP32 dapat berfungsi sebagai hotspot lokal untuk akses offline, sekaligus mendukung sinkronisasi ke *cloud* bila koneksi tersedia [5].

Selain monitoring, *Decision Support System* berbasis AI dibutuhkan untuk membantu petani menafsirkan data sensor dan memberikan *descriptive feedback*, misalnya kapan harus menyalakan pompa atau menambahkan nutrisi. Beberapa penelitian terbaru menunjukkan efektivitas integrasi AI, IoT, dan *smart greenhouse* dalam meningkatkan ketahanan pertanian [4].

Dengan demikian, penelitian ini berfokus pada pembuatan antarmuka *embedded web* yang mampu mengintegrasikan data monitoring dengan *descriptive decision support system* berbasis AI yang tetap dapat diakses meskipun tanpa internet, untuk mendukung petani hortikultura skala kecil dalam pengelolaan rumah kaca

2.2 Metode Pendekatan Masalah

Pemilihan metodologi penelitian ini berfokus pada bagaimana merancang dan mengembangkan antarmuka sistem rumah kaca berbasis ESP32 yang dapat diakses secara efektif dan mudah digunakan. Karena sistem ini akan digunakan oleh petani hortikultura skala kecil dan harus beradaptasi dengan kondisi

lingkungan yang dinamis, maka pendekatan pengembangan yang dipilih tidak bisa statis. Proses pemilihan metodologi dilakukan melalui perbandingan beberapa pendekatan yang umum digunakan, mencakup metode pengembangan perangkat lunak serta metodologi perancangan antarmuka pengguna. Metodologi pengembangan perangkat lunak menentukan cara bekerja dalam merencanakan, mengimplementasikan, menguji, dan merilis sistem. Di antara metode yang dipertimbangkan adalah *Waterfall*, *Prototype*, dan *Agile Development*.

Waterfall adalah pendekatan tradisional yang membagi proses ke dalam tahapan berurutan antara lain: analisis kebutuhan, desain, implementasi, pengujian, pemeliharaan. Kelebihannya adalah kejelasan urutan langkah dan dokumentasi yang kuat, yang memudahkan pengelolaan proyek dengan ruang lingkup yang stabil. Kekurangannya muncul ketika kebutuhan berubah pada saat proses *development* berlangsung, revisi akan memakan waktu karena harus kembali ke langkah awal. Hal ini sangat krusial bagi sistem rumah kaca, di mana kondisi lingkungan dan feedback pengguna bisa berubah seiring waktu.

Prototype berfokus pada pembuatan model *prototype* yang dapat diuji oleh pengguna sebelum sistem final dikembangkan. Kelebihannya adalah pengguna dapat melihat bentuk awal sistem, memberikan umpan balik, dan memperbaiki desain sebelum implementasi penuh dilakukan. Hal ini mengurangi risiko kesalahpahaman kebutuhan. Kekurangannya, jika tidak dikontrol, proses iterasi bisa menjadi terlalu panjang dan menunda rilis produk akhir. Selain itu, pengguna kadang menganggap *prototype* sebagai produk akhir sehingga menimbulkan ekspektasi yang salah.

Metode *Agile* membagi proses pengembangan menjadi beberapa siklus *development* yang disebut *sprint*. Setiap *sprint* menghasilkan versi awal sistem yang sudah bisa diuji. Masukan dari pengguna dikumpulkan di setiap tahap, lalu digunakan untuk memperbaiki sistem di sprint berikutnya. Kelebihan metode ini adalah kemampuannya beradaptasi dengan perubahan kebutuhan secara cepat dan bertahap. Berdasarkan perbandingan tersebut, Metodologi *Agile* dipilih karena paling sesuai dengan sifat penelitian ini. Sistem rumah kaca berbasis IoT memerlukan kemampuan beradaptasi dengan variabel lingkungan, dan *Agile* memungkinkan untuk diuji langsung di lapangan serta disesuaikan berdasarkan masukan pengguna dengan cepat. Lalu dilanjutkan dengan perancangan antarmuka pengguna yang menjadi aspek penting karena target pengguna adalah

petani skala kecil yang mungkin memiliki keterbatasan literasi digital. Agile juga mendukung kolaborasi antara pengembang sistem, dan pengguna [13]

Perancangan antarmuka (UI/UX) adalah langkah penting dalam penelitian ini karena sistem akan digunakan langsung oleh petani hortikultura, yang mungkin tidak terbiasa dengan teknologi canggih. Tujuannya bukan hanya membuat sistem yang berfungsi, tetapi juga yang nyaman, mudah dipelajari, dan membantu pekerjaan mereka. Untuk itu, beberapa pendekatan desain dipertimbangkan tiga metode antara lain *Design Thinking, Goal Directed Design*, dan *User Centered Design* (UCD).

Metode *Design Thinking* dimulai dengan memahami masalah secara mendalam, mengumpulkan ide sebanyak mungkin, membuat prototipe, lalu mengujinya. Kelebihannya adalah mendorong ide ide kreatif dan solusi inovatif. Namun, pendekatan ini bisa menghasilkan terlalu banyak ide yang sulit dipilih atau diimplementasikan jika waktu penelitian terbatas. Di sisi lain metode *Goal Directed Design* berfokus pada tujuan yang ingin dicapai pengguna. Prosesnya dimulai dengan membuat *user persona* lalu merancang sistem yang benar-benar membantu mereka mencapai tujuannya. Metode ini bagus untuk menghasilkan desain yang sangat terarah, tetapi membutuhkan data pengguna yang detail. Jika data tidak lengkap, hasil desain bisa tidak sesuai dengan kenyataan di lapangan. Metode UCD menempatkan pengguna sebagai pusat dari seluruh proses desain. Prosesnya dimulai dengan mengidentifikasi siapa saja pengguna utama dalam kasus ini, petani skala kecil. Lalu dilakukan analisis kebutuhan, seperti Sistem harus bisa diakses walaupun koneksi internet lemah. UI harus sederhana dan mudah dipahami. *Feedback* sistem harus jelas, misalnya menunjukkan kondisi tanaman atau peringatan dalam bahasa yang sederhana. Berdasarkan perbandingan ini, *User Centered Design* dipilih karena paling sesuai dengan konteks penelitian. Sistem rumah kaca ini akan dipakai oleh petani dengan berbagai latar belakang, sehingga keterlibatan mereka sangat penting. Dengan UCD, desain antarmuka tidak hanya terlihat menarik, tetapi juga benar-benar membantu pengguna memahami data rumah kaca dan mengambil keputusan dengan cepat. *Agile* memastikan sistem berkembang secara iteratif dan responsif terhadap kebutuhan baru, sementara UCD menjamin bahwa setiap versi prototipe benar benar sesuai dengan pengalaman dan kebutuhan pengguna [14], [15].

2.3 Teknologi dan Tools Pendukung Pembuatan Solusi

Pengembangan sistem rumah kaca berbasis ESP32, Preact, PicoCSS, *cloud computing*, dan AI enhanced DSS membutuhkan integrasi beberapa teknologi pendukung. Pemilihan teknologi ini didasarkan pada pertimbangan efisiensi, keterbatasan perangkat keras, kemudahan implementasi, serta kesesuaian dengan kebutuhan pengguna akhir. ESP32 sebagai Pusat Kendali IoT merupakan mikrokontroler yang mendukung Wi-Fi dan Menyediakan akses offline melalui *hotspot* lokal, mengirim data ke *cloud* saat koneksi internet tersedia. Dengan konsumsi daya rendah serta memori yang terbatas (4Mb), namun cukup untuk menjalankan *server web* kecil. Beberapa studi menunjukkan bahwa ESP32 mampu mengintegrasikan AI sederhana dan IoT pada perangkat dengan sumber daya terbatas, sehingga cocok untuk aplikasi pertanian serta dengan dukungan *Cloud Computing* untuk Penyimpanan dan Analitik digunakan untuk penyimpanan data historis, analisis tren, serta akses jarak jauh. Integrasi IoT dan Cloud memungkinkan monitoring dan kontrol secara *real time* meskipun lokasi rumah kaca berada di daerah terpencil [1], [16].

PostgreSQL sebagai sistem manajemen basis data utama karena kemampuannya yang kuat dalam menangani data terstruktur, analisis kompleks, serta integrasi dengan sistem *cloud* dan IoT. PostgreSQL juga mendukung terhadap data *time series* melalui ekstensi seperti TimescaleDB, yang sangat berguna untuk menyimpan data sensor dari ESP32. Dengan pendekatan ini, data yang dikirim dari sensor secara periodik dapat disimpan, dan di *query* secara efisien berdasarkan rentang waktu. Hal ini memudahkan proses analisis, prediksi kondisi lingkungan, serta penyusunan laporan historis. PostgreSQL juga mendukung JSONB, memungkinkan sistem menyimpan data semi terstruktur dari berbagai sensor IoT tanpa kehilangan fleksibilitas. Dengan fitur ini, sistem tetap efisien meskipun format data sensor mengalami perubahan atau pembaruan *firmware* pada perangkat ESP32.

Penelitian ini juga mempertimbangkan pemilihan framework *front end* yang ringan, cepat, dan sesuai dengan keterbatasan perangkat ESP32. ESP32 memiliki memori terbatas, teknologi yang dipilih harus mampu memberikan tampilan antarmuka yang responsif tanpa membuat sistem menjadi lambat. *Framework* JavaScript digunakan untuk membuat antarmuka yang interaktif. Tiga framework yang dibandingkan adalah React, Vue, dan Preact. React adalah framework paling populer untuk membangun antarmuka berbasis komponen. Beberapa keuntungan

nya antara lain banyak dokumentasi dan tutorial sehingga mudah dipelajari, Ekosistem besar, banyak *library* pendukung, Kompatibel dengan berbagai tools modern. Namun, React memiliki ukuran file yang cukup besar untuk dijalankan di perangkat IoT. Bundle React rata rata mencapai puluhan kilobyte, sehingga dapat memperlambat waktu build aplikasi. Vue menawarkan pendekatan yang lebih sederhana dibanding React. Keunggulannya adalah ukurannya lebih kecil dibanding React, dan Dokumentasi cukup jelas. Kelemahannya ekosistem Vue sedikit lebih kecil dibanding React, dan masih relatif berat jika dijalankan di perangkat dengan memori sangat terbatas. Preact adalah versi mini dari React. Ukurannya sangat kecil sekitar 3 KB , tetapi API nya hampir sama dengan React. Keunggulannya antara lain sangat ringan, cocok untuk IoT dan perangkat terbatas yang tidak memerlukan banyak konfigurasi tambahan, bisa memanfaatkan pengetahuan React yang sudah ada. Kekurangannya adalah beberapa fitur React seperti *Context* API lanjutan atau *Suspense* mungkin tidak sepenuhnya tersedia, sehingga kadang perlu penyesuaian.

Selain framework Javascript Framework CSS membantu mempercepat pembuatan tampilan yang konsisten dan menarik. Beberapa framework yang dibandingkan adalah Bootstrap, Tailwind CSS, dan PicoCSS. Bootstrap terkenal dengan komponen siap pakai seperti tombol, form, dan *grid* sistem. Keuntungannya cepat dalam membuat prototipe, banyak dokumentasi dan komunitas, desain responsif, namun, ukuran file CSS nya relatif besar. Disisi lain Tailwind menggunakan pendekatan utility first, yang memberi fleksibilitas tinggi dalam mendesain. Keunggulannya sangat bisa dikustomisasi, diberikan kebebasan dalam mendesain sesuatu, mendukung penghapusan class yang tidak dipakai untuk mengecilkan ukuran file. Namun, untuk proyek kecil, setup Tailwind bisa memakan waktu. Selain itu, kode HTML bisa terlihat ramai karena banyaknya class yang dipakai di setiap elemen. PicoCSS adalah framework CSS yang sangat minimalis. Kelebihannya ukuran file sangat kecil, sehingga cepat dimuat, serta tidak memerlukan build process atau konfigurasi rumit. kekurangannya, opsi kustomisasi tidak sebanyak Tailwind, tetapi untuk aplikasi yang sederhana seperti sistem rumah kaca, PicoCSS sudah cukup untuk menghasilkan tampilan yang profesional.

Untuk pemilihan model AI yang digunakan dalam penelitian ini, saya mempertimbangkan dua kandidat utama yaitu ChatGPT dari OpenAI dan Gemini AI dari Google. ChatGPT, yang ditenagai oleh arsitektur *Generative Pre-trained*

Transformer (GPT), telah menjadi standar industri dalam banyak hal, terutama dalam hal kefasihan dan kreativitas. Model model OpenAI seperti GPT 3.5 dan GPT 4/4o dilatih pada set data teks yang masif. Hasilnya, mereka menunjukkan kemampuan yang luar biasa dalam menghasilkan tulisan yang natural. ChatGPT sering kali unggul dalam tugas tugas yang membutuhkan *brainstorming* imajinatif, *storytelling*, dan bantuan coding yang kompleks. Kelemahan utama, terutama pada versi gratisnya adalah batas pengetahuan. Model ini tidak memiliki pengetahuan, data, atau publikasi yang terjadi setelah tanggal pelatihannya. Untuk penelitian yang membutuhkan data terkini, ini merupakan hambatan signifikan.

Disisi lain Gemini dibangun di atas pondasi infrastruktur Google yang berpusat pada informasi. Keunggulan utama Gemini adalah integrasi bawaan dengan Google *Search*. Yang berarti model tersebut dapat mengakses, memproses informasi real time dari web. Untuk tugas tugas seperti analisis tren pasar terbaru, tinjauan literatur atas publikasi ilmiah terkini, atau pengumpulan data statistik terbaru [17], [18]. Versi gratis Gemini saat ini ditenagai oleh model Gemini Pro yang modern dan, secara krusial, sudah mencakup akses *real time* ke internet serta kemampuan multimodal/memahami gambar. Ini memberikan fungsionalitas yang lebih relevan untuk penelitian dibandingkan dengan penawaran gratis dari kompetitornya, yang seringkali membatasi pengguna pada model yang lebih tua dan tanpa akses internet. Komponen terpenting dalam penelitian ini adalah integrasi AI API sebagai inti dari *descriptive decision support system*. Gemini dipilih karena mendukung *multimodal input*, reasoning tingkat lanjut, serta window konteks yang lebih besar, sehingga memungkinkan prompt yang lebih kompleks dan kaya konteks. Keunggulan lain dari Gemini API adalah kemudahan integrasi API berbasis JSON, dukungan *embeddings* untuk pemahaman konteks bahasa yang lebih baik, serta tersedianya *free tier* yang memadai untuk skala penelitian kecil. Hal ini membuat Gemini jauh lebih efisien dibandingkan membangun model machine learning dari nol yang membutuhkan dataset besar, training berulang, serta sumber daya komputasi tinggi [4]. Selain aspek teknis, penerapan AI dalam penelitian ini juga memperhatikan prinsip *Responsible* AI dan *Explainable* AI agar solusi yang dibangun benar benar bermanfaat bagi pengguna. *Responsible* AI menekankan pada keadilan, transparansi, akuntabilitas, keamanan, dan inklusivitas dalam penggunaan AI [19]. Dalam konteks penelitian ini, penerapan *Responsible* AI berarti Gemini AI yang digunakan harus memberikan rekomendasi yang dapat dipertanggungjawabkan,

tidak menyesatkan petani, serta menjaga privasi data sensor. Di sisi lain, *Explainable* AI berfokus pada kemampuan sistem untuk menjelaskan mengapa rekomendasi tertentu diberikan. Penerapannya dalam penelitian ini berarti setiap deskripsi atau rekomendasi dari Gemini AI akan disertai alasan berbasis data sensor. Misalnya, jika AI menyarankan menghidupkan *blower*, sistem juga harus menjelaskan bahwa hal tersebut disebabkan oleh suhu rumah kaca yang terdeteksi melebihi batas optimal. Dengan pendekatan ini, pengguna tidak hanya menerima instruksi, tetapi juga memahami logika di balik rekomendasi, sehingga meningkatkan kepercayaan dan adopsi teknologi.

Bentuk prompt deskriptif yang diberikan ke Gemini AI API, output DSS berupa rekomendasi yang deskriptif, adaptif. Dengan pendekatan ini, DSS lebih fleksibel, mudah diperluas, dan *user friendly*. Implementasi AI pada smart *greenhouse* terbukti mampu meningkatkan efisiensi penggunaan air, nutrisi, dan energi [20]. Dengan demikian, kombinasi Preact, PicoCSS, dan Gemini API membentuk kerangka utama penelitian ini. Preact dan PicoCSS menjamin UI yang ringan dan ramah pengguna, Gemini AI API menghadirkan lapisan kecerdasan yang memungkinkan antarmuka tidak hanya menampilkan data, tetapi juga memberikan *descriptive decision support* yang etis, transparan, dan dapat dijelaskan kepada pengguna

2.4 Metode Evaluasi dan Validasi Solusi

Tahap evaluasi dan validasi merupakan komponen penting untuk memastikan bahwa solusi rumah kaca berbasis ESP32 dengan integrasi AI DSS benar-benar mampu memenuhi kebutuhan pengguna, bekerja secara optimal, serta aman digunakan. Antarmuka pengguna berbasis Preact dan PicoCSS diuji untuk memastikan kemudahan akses baik di mode *offline* maupun *online*. Modul AI DSS diuji dengan skenario nyata, misalnya prediksi kebutuhan irigasi. Uji coba dan evaluasi dilakukan bersama pengguna lapangan untuk menguji pemahaman, kecepatan akses, dan kemudahan kontrol. Iterasi perbaikan desain berdasarkan masukan pengguna [17], [18].

Evaluasi Performa Pengujian performa dilakukan untuk menilai kecepatan respons sistem pada mode offline dan online dalam menampilkan data, bagaimana user interface tetap responsif meskipun diakses melalui mobile ataupun web. Evaluasi Pengguna dilakukan dengan pendekatan *User Centered Design*, di mana petani sebagai pengguna utama dilibatkan dalam proses uji coba. Pengguna diminta memberikan *feedback* terkait *usability*, dan kemudahan

memahami rekomendasi DSS. Metode ini mengacu pada praktik evaluasi IoT dengan antarmuka cerdas yang juga diterapkan pada sistem keamanan rumah berbasis AI [9].

Validasi data pada penelitian ini bertujuan untuk memastikan bahwa data sensor yang dikumpulkan oleh ESP32 akurat, dan rekomendasi yang dihasilkan Gemini AI melalui prompt benar, konsisten, dan bermanfaat bagi pengguna. Karena DSS berbasis prompt ke Gemini AI, maka validasi difokuskan pada kualitas rekomendasi deskriptif. Pendekatan yang digunakan adalah validity oleh petani yang mencoba sistem, lalu memberikan umpan balik terkait apakah rekomendasi mudah dipahami, apakah dapat membantu mengambil keputusan , dan pakah rekomendasi relevan dengan kondisi nyata di lapangan. Data sensor yang sama dikirim beberapa kali, lalu diperiksa apakah Gemini memberikan rekomendasi yang konsisten. Jika berbeda, dicek apakah variasi jawaban masih logis. Rekomendasi Gemini dibandingkan dengan petani, jika hasilnya sesuai, maka rekomendasi dianggap valid

Dengan kombinasi evaluasi teknis, uji pengguna, dan validasi data (akurasi & reliabilitas), penelitian ini memastikan bahwa sistem rumah kaca cerdas yang dibangun tidak hanya berfungsi secara teknis, tetapi juga relevan, dapat dipercaya, dan bermanfaat nyata bagi pengguna lapangan

2.5 *State of The Art*

Penelitian ini mengintegrasikan beberapa komponen utama dalam satu kesatuan sistem rumah kaca berskala kecil. ESP32 tidak hanya berfungsi sebagai pengumpul data sensor, tetapi juga sebagai hotspot lokal, sehingga sistem dapat tetap diakses meskipun tidak tersedia koneksi internet. Fitur ini memastikan bahwa pemantauan kondisi rumah kaca tetap dapat dilakukan kapan saja. Kombinasi *Agile Development* dan *User Centered Design* adalah pendekatan paling tepat untuk penelitian ini. *Agile* menjamin pengembangan iteratif yang responsif terhadap perubahan kebutuhan, sementara UCD memastikan bahwa setiap iterasi menghasilkan desain yang sesuai dengan pengalaman pengguna. Dengan dukungan literatur terbaru seperti Guerrero-Ulloa et al. (2023) [21], pendekatan ini memiliki dasar ilmiah yang kuat untuk diterapkan pada sistem rumah kaca. Pemilihan framework Preact dan PicoCSS memungkinkan antarmuka yang ringan, cepat, serta *mobile friendly*. Hal ini mendukung akses yang nyaman melalui perangkat seluler maupun desktop, baik dalam mode lokal maupun saat terhubung dengan *cloud*. Sistem pendukung keputusan dalam penelitian ini tidak berbasis

algoritma *machine learning* tradisional, melainkan berbasis prompt pada Gemini AI. Dengan pendekatan ini, data sensor dapat langsung diterjemahkan menjadi descriptive feedback yang mudah dipahami oleh pengguna, serta dapat dikembangkan untuk interaksi berbasis tanya jawab. Dengan kombinasi tersebut, penelitian ini menghadirkan solusi praktis, adaptif, dan ramah pengguna untuk mendukung pengelolaan rumah kaca skala kecil. Pendekatan ini mengintegrasikan beberapa instrumen berupa IoT, *lightweight* UI, serta *Decision Support System* berbasis AI.